Neural Processing Letters (2020) 51:905–946
https://doi.org/10.1007/s11063-019-10120-x

Training a Neural Network for Cyberattack Classification Applications Using Hybridization of an Artificial Bee Colony and Monarch Butterfly Optimization

Waheed A. H. M. Ghanem[1,2,3] · Aman Jantan[1]

Published online: 1 October 2019

Abstract
Arguably the most recurring issue concerning network security is building an approach that is capable of detecting intrusions into network systems. This issue has been addressed in numerous works using various approaches, of which the most popular one is to consider intrusions as anomalies with respect to the normal traffic in the network and classify network packets as either normal or abnormal. Improving the accuracy and efficiency of this classification is still an open problem to be solved. The study carried out in this article is based on a new approach for intrusion detection that is mainly implemented using the Hybrid Artificial Bee Colony algorithm (ABC) and Monarch Butterfly optimization (MBO). This approach is implemented for preparing an artificial neural system (ANN) in order to increase the precision degree of classification for malicious and non-malicious traffic in systems. The suggestion taken into consideration was to place side-by-side nine other metaheuristic algorithms that are used to evaluate the proposed approach alongside the related works. In the beginning the system is prepared in such a way that it selects the suitable biases and weights utilizing a hybrid (ABC) and (MBO). Subsequently the artificial neural network is retrained by using the information gained from the ideal weights and biases which are obtained from the hybrid algorithm (HAM) to get the intrusion detection approach able to identify new attacks. Three types of intrusion detection evaluation datasets namely KDD Cup 99, ISCX 2012, and UNSW-NB15 were used to compare and evaluate the proposed technique against the other algorithms. The experiment clearly demonstrated that the proposed technique provided significant enhancement compared to the other nine classification algorithms, and that it is more efficient with regards to network intrusion detection.

Keywords Intrusion detection system (IDS) · Neural network training · A multilayer perceptron (MLP) · Swarm Intelligence (SI) · Artificial Bee Colony Algorithm (ABC) · Monarch Butterfly Optimization (MBO)

✉ Waheed A. H. M. Ghanem
Waheed.ghanem@gmail.com

[1] School of Computer Science, Universiti Sains Malaysia, Penang, Malaysia

[2] Faculty of Engineering, University of Aden, Aden, Yemen

[3] Faculty of Education-Saber, University of Aden, Aden, Yemen

1 Introduction

Today system assaults are inescapably pervasive and growing in number, which automatically results in the unprecedented demand for systems which can detect intrusion. IDSs were created for the first time in 1980 by Anderson [1] and were later perfected by Denning [2]. This system was either manifested as a device or a software tool and has been constantly improving since its inception. The main purpose of the intrusion detection system (IDS) is to seek out and react to intrusive and harmful activities aimed at the system's resource and facilities. This can be achieved by paying close attention to the activities of the system and analyzing networks [3]. IDSs can be categorized into those that detect anomalies and those that detect misuse: what makes the two different is the means of detection. To detect misuse the IDS searches for the attack fingerprint in quite a large database that has all attack signatures stored. Conversely, to detect anomalies, the IDS can notice how the system behaves differently from normative behavior. The use of both methods creates another method known as the 'hybrid technique'. Naturally, it produces better results than using a singular separate method [4].

Recently data mining and neural networks have occupied a crucial position in increasing the quality in the performance of IDSs. This has facilitated the process of categorizing the kinds of attacks needed to calculate the effectiveness of the IDS. The main purpose of the data mining procedure is to obtain simulating information from big knowledge warehouses and convert it into a data structure that can be understood. The most used methods in data mining are: information preprocessing, clustering, recognizing patterns and classifying the information. The most significant technique is the classification as it is of the utmost importance to determine accurately the aimed class for each situation in the information. The categorization implies finding the hidden pattern in information and can be considered a usual issue in data mining, and in learning how to operate a machine [5].

Quite a few suggestions regarding data mining have been utilized by creating systems to detect anomalies and some examples could be the artificial neural networks (ANNs), radial basis function (RBF) [6, 7], multi-layer perceptron (MLP) [8–10], fuzzy neural network (FNN) [9], self-organizing map (SOM) [11, 12], support vector machines (SVMs) [13–16] and SVM with modified versions [17, 18].

Recently biology and natural systems have been used by researchers, and innovative systems such as 'Swarm Intelligence' have been obtained thereby; these depicting the conduct of animals and insects [19]. This conduct includes activities such as finding food resources, creating their nests, and moving the nests from one place to another-this is then analyzed thoroughly (this also aided the improvement of the IDS performance). By being better able to trace the source of the attack it was possible to differentiate between malicious and non-malicious behavior, and also to offer elucidation pertaining to some complicated issues [20].

Artificial neural networks (ANNs) can be defined as the most significant parts of the Artificial intelligence. They are categorized as either Supervised Learning Neural Networks or Unsupervised Learning Neural Networks; the difference between the two being that the supervised network has to learn under supervision of another person, whereas the unsupervised kind of network does not need this (as indicated by its name). To be successful in its activity the system depends upon the following factors [21]: the architecture of the system, the training algorithm, and the features utilized in the training. These factors make designing an optimal neural network difficult [22]. Furthermore, each of these factors should be chosen correctly so that the training algorithm does not fall into a local minimum. In [23] there have been reports on methods based on the heuristic algorithms in order to obtain a good ANNs model.

Artificial Neural Networks have certain qualities that permit them to resolve a range of issues such as pattern classification, regression, and forecasting. Some of the most notable traits are the ability to learn from examples, adaptability to generalize and to be able to solve problems such as the classification of patterns, and to approximate functions and optimization [24, 25].

ANNs have been created in different kinds of multilayer Neural Networks. The majority of applications utilize a feed-forward type of NNs that implies the use of the typical back propagation (BP) learning method. This type of training usually concludes by utilizing the back-propagation (BP) gradient descent (GD) method. When using this algorithm some issues may be confronted as the algorithm is based on the gradient. One of the most notable problems is the propensity to get stuck in a local minimum as a result of the quite-low speed of convergence [26, 27].

Moreover, the BP method has to be able to determine a few essential learning parameters such as the rate of learning, the momentum, and the prearranged structure. The BP method has a prefixed NNs structure, meaning it trains only its weights in its structure. As a result there is no solution as for how to design a nearly perfect NNs arrangement for an application. Global optimum search methods are capable of avoiding local minima and are usually utilized to regulate weights of NNs, such as the artificial bee colony (ABC) [21, 28], particle swarm optimization (PSO) [29, 30], evolutionary algorithms (EA), simulated annealing (SA) and ant colony optimization (ACO), meaning they can also be used in order to dispose of the problems in standard and typical algorithms.

This research focuses on dealing with the fixed structure of the ANN. ANNs are considered some of the most widely used machine-learning techniques. Many complex and practical problems have been successfully solved, which are conversely difficult to solve through other methods. Despite this, the general architecture of ANNs still suffers from the local optima issue and low convergence speed [31–33]. There are three major drawbacks to an ANN-based intrusion detection system.

- The error function of an ANN is a multimodal function that is frequently trapped into local minima.
- This type of ANN-based IDS demonstrates a slow convergence.
- Over-fitting usually creates an overly complex model.

To overcome the shortcomings attributed to the BP training algorithm and avoid falling into a local minimum, we use our previously proposed HAM algorithm to train ANN [34, 35]. Our new proposal may provide an effective and suitable alternative solution for the problem of Multilayer Perceptron Neural Network training algorithm and the global and local optima in a multimodal search space. It may be evident from our previous work that HAM algorithm could guarantee to find a global optimal solution, whereas the BP algorithm could only guarantee finding the initial point at the end of the slope of the search space (*local optimum*).

In this article we propose the new HAMMLP technique which improves the intrusion detection rate as well as reduces the false alarm rate. The main idea of the new approach is to solve the problem of the training algorithm in ANN to identify new attacks of IDS, and to evaluate three IDS datasets, two of which are new, namely UNSW-NB15 and ISCX 2012, as well the antique dataset KDD Cup 99. In this work we demonstrate that we managed to solve the dataset-related problems whilst managing to enhance the detection of intrusions. The main contributions in this research are summarized, as follows:

1. Building a new hybrid HAM algorithm that optimizes the MLP neural network and achieves its effectiveness in addressing the shortcomings of artificial neural networks in the field of network intrusion detection.
2. Assessing the performance, reliability and validity of the new technique in detecting a new attack by using two new datasets (ISCX 2012 and UNSW-NB15) and compare them with the (KDD Cup 99) dataset.
3. Comparing our new proposed approach with other evolutionary and swarm intelligence algorithms, using the four of IDS datasets to confirm the applicability of our approach.
4. Comparing our work with other related works of literature by using the datasets to evaluate the performance of our proposal.

The advantages of our proposal as following:

- A high accuracy rate in detecting the attacks on the network.
- The possibility of detecting an unknown attack.
- Reduce the false alarm rate.

Section 2 reviews the materials and methods which include: describing the methodology then outlining the mathematical overview of the neural network and HAM algorithm; then discussing how the HAM algorithm could be deployed to train the MLP. Section 3 describes the experimental setup which includes: the validation of the IDS, algorithms and parameters, and performance measuring. Section 4 presents the experimental results. Section 5 summarizes the conclusions and provides directions for future research.

2 Materials and Methods

2.1 Hybrid Algorithm Based on Artificial Bee Colony and Monarch Butterfly Optimization

The most important factors in metaheuristic algorithms are the exploitation and exploration search mechanisms. A good metaheuristic algorithm has the ability to strike a balance between these two mechanisms, thereby enhancing the solving of low and high-dimensional optimization problems. The exploitation mechanism is based on the present knowledge which is to seek better solutions; while the exploration mechanism is based on fully searching the problem space for an optimal solution [34, 35].

In general by analyzing the standard MBO algorithm we notice that it has the ability to explore the search space very effectively; it also has the ability to find the global optimum in a fast speed; however, it has a poor ability to exploit the local search space due to the occasional use of Levy flight by the updating operators, which leads to large steps (or moves). On the other side we notice that the ABC algorithm has the ability to explore the search space relatively well, but has better ability in finding local optima through the two phases of employee and onlooker bees, which are considered local search processes. ABC is mostly based on selecting the solutions that improve the local search. There is one fundamental difference between those two phases: an onlooker bee relies on the probability value of the solution in order to select it where it chooses solutions that have high fitness value, while solutions with low fitness values are used to produce the trial solutions. Global search, on the other hand, is implemented in the ABC algorithm by the scout phase, resulting in the reducing of the convergence speed during the search process.

As presented in our previous work [34, 35], the main idea of the HAM algorithm is based on two improvements; first, to modify the butterfly adjusting operator in the MBO algorithm in order to improve the exploitation versus exploration balance, by increasing the search diversity and counterbalancing the shortfall of ABC algorithm in global search efficacy. The modified version of the operator is shown in algorithm 1. The second improvement is to integrate the modified butterfly adjusting operator from MBO in place of the first phase in the standard ABC algorithm (the employee phase). The improved operator is named as "*employee bee adjusting operator*".

Algorithm 1: *Employee bee adjusting phase*

```
Begin
    For i = 1 to SN do
      Calculate the walk step dx by Eq. $dx_k = levy(x_j^t)$;
      Calculate the weighting factor by Eq. $\propto = S_{max}/t^2$;
      For j =1 to D do
      If rand ≥ limit1 then
            Generate the j^th element by Eq. $x_{i,j}^{t+1} = x_{best,j}^t$;
            $x_{i,j}^{t+1} = x_{best,j}^t$;
      Else
            Randomly select a food Source (r) by Eq. $r = round((SN * rand) + 0.5)$;
            If rand<limit2 then
               Generate the j^th element by Eq.
               $x_{i,j}^{t+1} = x_{r,j}^t + 0.5 * rand * (x_{worst,j}^t - x_{r,j}^t - x_{best,j}^t)$;
            Else
               Generate the j^th element by Eq.
               $x_{i,j}^{t+1} = x_{r,j}^t + 0.5 * rand * (x_{best,j}^t - x_{r,j}^t - x_{worst,j}^t)$;
               If rand < BAR, then
                 $x_{i,j}^{t+1} = x_{i,j}^{t+1} + \propto \times (dx_k - 0.5)$;
               End if
            End if
         End if
      End for j
      Evaluate the fitness value of the candidate solution $x_i$.
      Apply a greedy selection process between $x_i$ and $x_{best}$
      If solution $x_i$ does not improve, $trial_i = trial_i + 1$, otherwise $trial_i = 0$.
    End for i
End.
```

The proposed Hybrid ABC and MBO (HAM) algorithm is shown in Fig. 1. This algorithm includes four phases: *Initialization, Employee bee adjusting phase*, *Onlooker bee phase* and *Scout bee phase* (the onlooker and scout phases are the same phases inherited from the standard ABC algorithm). Thus, the new HAM algorithm is essentially

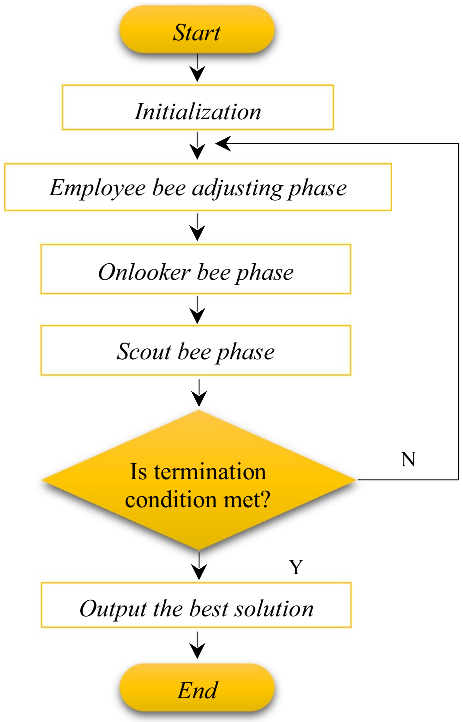


Fig. 1 Flowchart of the HAM algorithm

an integration of the effective local search phase (*onlooker bee*) and two global search phases (*Employee bee adjusting* and *Scout bee*) for effective global optimization.

In the *Initialization* phase we need to define all the variables that would be defined in the standard ABC algorithm and assign them their respective suitable values. The HAM algorithm adopts all parameters from the original ABC algorithm then adds three new control parameters: *limit1*, *limit2* and the maximum walk step parameter; these three parameters are used in the *employee bee adjusting* phase.

In the *employee bee adjusting* phase, each employee bee is assigned to its food source and in turn generates a new one either by using *Levy flight* or through mutation operators, which are based on the two control parameters (*limit1* and *limit2*). These parameters are used to fine-tune the exploitation versus exportation by improving the global search diversity. The *employee bee adjusting* phase is very simple and is used to update all solutions in the bee population, where each solution is a *D*-dimensional vector.

The first step in this phase is to calculate a walk step "*dx*" for the *i*th bee using the *levy flight* in Eq. 1, and calculate a weighting factor "∝"using Eq. 2, where S_{max} represents the max walk step that a bee individual can move in one step, and *t* is the current generation. Then, for each element *j* of the *D* dimensions, if ($rand \geq limit1$), the algorithm uses Eq. 4 to update the solution element:

$$dx_k = levy(x_j^t) \tag{1}$$

$$\propto = S_{max}/t^2 \tag{2}$$

$$x_{i,j}^{t+1} = x_{i,j}^{t+1} + \propto \times (dx_k - 0.5) \tag{3}$$

$$x_{i,j}^{t+1} = x_{best,j}^{t}; \tag{4}$$

where $x_{i,j}^{t+1}$ is the *j*th element of solution x_i at generation *t+1*, which represents the location of the solution *i*, while $x_{best,j}^{t}$ is the *j*th element of x_{best} at generation *t*, which represents the best location among the food sources so far with respect to the *i*th bee. In contrast, if (*rand* <. *limit1*) then another set of updates are performed. First, a random food source (equivalent to a random solution or bee) is selected from the current population using Eq. 5. Then, depending on whether a randomly generated value is smaller than *limit2*, Eq. 6 is used to update the solution elements, as follows:

$$r = round((SN * rand) + 0.5) \tag{5}$$

$$x_{i,j}^{t+1} = x_{r,j}^{t} + 0.5 * rand * (x_{worst,j}^{t} - x_{r2,j}^{t} - x_{best,j}^{t}) \tag{6}$$

where $x_{i,j}^{t+1}$ is the *j*th element of solution x_i at generation *t+1*, which represents the location of the solution *i*,$x_{best,j}^{t}$ is the *j*th element of x_{best} at generation *t*, which represents the best location among the food sources so far, $x_{worst,j}^{t}$ is the *j*th element of x_{worst} at generation *t*, which represents the worst location among the food sources so far, and $x_{r,j}^{t}$ is the *j*th element of x_r at generation *t*, which represents the location of the solution *r* calculated by Eq. 5. The *t* in Eq. 6 is the current generation number.

On the other hand, if the randomly generated value was bigger than *limit2,* the solution elements are updated by Eq. 7, where $x_{i,j}^{t+1}$ is the *j*th element of solution x_i at generation *t+1*, which represents the location of the solution *i*, $x_{best,j}^{t}$ is the *j*th element of x_{best} at generation *t*, which represents the best location among the food sources so far, $x_{worst,j}^{t}$ is the *j*th element of x_{worst} at generation *t*, which represents the worst location among the food sources so far, while $x_{r,j}^{t}$ is the *j*th element of x_r at generation *t*, which represents the location of the solution *r* calculated by Eq. 17.

$$x_{i,j}^{t+1} = x_{r,j}^{t} + 0.5 * rand * \left(x_{best,j}^{t} - x_{r3,j}^{t} - x_{worst,j}^{t}\right) \tag{7}$$

The *levy flight* step from the MBO algorithm is adopted here with a smaller probability of execution to reduce its impact on the exploitation process. Assuming the execution path passed the test of *limit1* and *limit2* control parameters, yet another random check against the *BAR* parameter is performed, right after the update by Eq. 7 to further change the value of $x_{i,j}^{t+1}$ occasionally by the amount $\propto \times \left(\mathrm{dx_k} - 0.5\right)$, as per Eq. 3.

Finally, the *employee bee adjusting* phase tests the boundary for the new solution to make sure the newly generated solution is within the allowed boundaries for the optimization problem at hand. It then evaluates the fitness value of the new solution in order to apply a 'greedy' selection process between the new and the best solutions in order to select the better one. If the solution does not improve then a trial counter is increased by one. As for the onlooker-bee and scout phases, the algorithm adopts their implementation from the original ABC algorithm without any change.

2.2 The HAM Adaptation Process

The adapting process is an important step for the optimization of the ANN by using evolutionary and swarm intelligence for the training of ANNS. In all evolutionary-based and swarm-based methods, the training process translates into using suitable MLP weights representation, fitness function and termination condition(s). Consequently, in order to adapt the HAM algorithm as an MLP training method these three issues must be adapted to suit the enforcement of the HAM algorithm, and to satisfy the requirements assigned by the MLP training process in general. This new training method is called the HAMMLP algorithm.

The methodology of using a metaheuristic algorithm for training neural networks is based on three methods. Firstly, algorithms are used to find a combination of weights and biases that provide a minimum of error for any type of neural network, and to reduce the MSE (*Mean Square Error*) which represents the cost function of the ANN. Secondly, the algorithms are utilized to find a suitable structure for any type of neural network in a particular problem. Lastly, the metaheuristic algorithm is used to tune the parameters of a gradient-based learning algorithm, such as the learning rate and momentum. In the first method the structure of neural network model (MLP) does not change during the learning process. The training algorithm seeks after finding the suitable values for all connection weights and biases for minimizing the overall error of the MLP. While in the second method the architecture of the MLP model varies. The training algorithm determines the best architecture of the MLP model for solving a specific problem. Changing the architecture can be accomplished by manipulating the connections between neurons, the number of hidden layers, and the number of hidden nodes in each layer.

For example, Yu et al. [36] used the PSO algorithm for defining the architecture of the MLP model to solve two real problems. And, Leung et al. used the EA algorithm to tune the parameters of an FNN by applying the last method. There are some studies that employed a combination of methods simultaneously: for instance, Mizuta et al. [37] and Leung et al. [38] used the genetic algorithm and the improved genetic algorithm to define the architecture of neural network model FNN. In this work, the HAM algorithm is applied to an MLP using the first method. To design a HAM trainer for MLPs, the following main stages should be completed:

2.2.1 Representing the Weights and Biases for MLP Using HAMMLP

This section shows the details of the HAMMLP method in order to improve the weights and biases of MLP model; the aim being to reduce overall error. Moreover, any MLP model will rely on the number of hidden layers, the number of nodes in each hidden layer, which in turn relate to weights, while the biases relate to each node in hidden and output layers.

In fact, the methods of representing weights and biases can be one of these three methods: matrix, binary, and vector. This was presented in detail in our previous work [39, 40]. In matrix encoding every solution is encoded as a matrix. To train the MLP, each solution is representative of all the weights and biases. While in the binary representation solutions are encoded as the strings of binary bits. And in vector representation every agent is encoded as a vector. Each of these three methods has its own advantages and disadvantages that can be useful in a particular application [41].

In the vector method the encoding process is much easier than the decoding process. It is often used for simple neural network models. In the matrix method the decoding process is easy but the encoding is difficult for neural network models with complex architectures; it is very appropriate for learning algorithms from generalized neural network toolboxes. But naturally the binary method needs to represent variables in the binary form. In this case the length of each solution will be increased when the architecture starts to become more complex; this leads to the process of decoding and encoding also becoming very intricate.

This study does not use any generic toolboxes because the run time is much less for our hand-coded MLPs. As an example of this encoding process the final vector of the MLP is shown in Fig. 2.

This study proposes a new algorithm (HAMMLP) for training the MLPs. In the HAMMLP solutions are represented by two vectors:

1. The first vector represents the solution structure which contains the number of inputs, number of hidden layers, and the number of nodes in each hidden layer in the MLP Model.
2. The second vector does represent the weights and biases of the solution, which corresponds to the weights and biases in the MLP model. For the purpose of this study the structure is fixed before training MLPs. The objective is to reach a training algorithm that excels in finding suitable values for all connection weights and biases that ultimately contribute to minimizing the MLPs overall error.

Each of these two vectors have different representations; the cell of the structure vector contains 0 or 1, while the cell of the vector which represents the weights and biases of the solution contains a real number in the range of [0, 1]. The solution representation in HAMMLP is provided in Fig. 2. The solution structure vector divides into three groups; the first group contains a set of cells representing the number of nodes in the input layer, which is considered a feature of the datasets. The length of the weights and biases solution vector is based on the number of the weights, plus the number of biases in the neural network model; it is deduced by Eq. 8. The number of weights and biases is dependent on the number of hidden layers and the number of nodes in each hidden layer participating in the solution structure; they are calculated by Eqs. 9 and 10.

$$\text{Length of weights and biases solution vector} = W + B \tag{8}$$

$$W = (I \times N) + ((N \times N) \times (H - 1)) + (N \times O) \tag{9}$$

$$B = H \times N + O \tag{10}$$

where W represents the number of weights, B represents the number of biases; I denotes the number of nodes in the input layer, N represents the number of nodes in each hidden layer, H represents the number of hidden layers, and O denotes the number of nodes in the output layer.

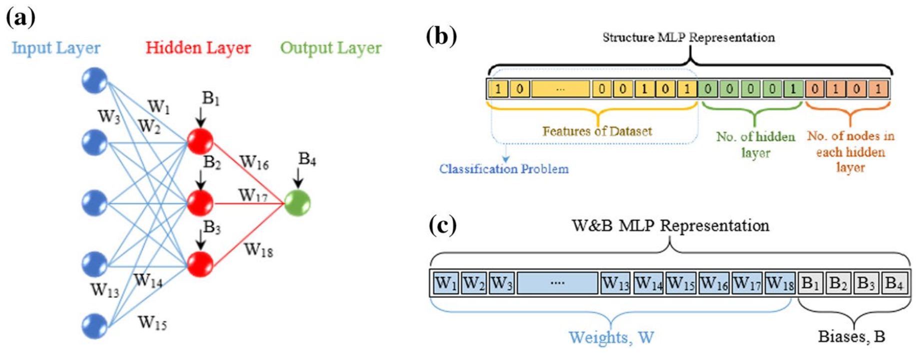


Fig. 2 Solution representation of HAMMLP algorithm

In the case of classification (as the feature selection is vital in the classification process) several cells are added in the solution structure for the selection of inputs for the neural network as a feature selection part. Undoubtedly, the number of cells in the features part is equal to the number of the conditional features in the dataset. If one is an element of the features part, then the particular feature from the conditional feature set is contained in the subset. If zero, the subset does not contain the specified feature. All the experiments in this study employ the complete set of features in the dataset.

2.2.2 Adapting the HAMMLP Quality Measure (Fitness Function)

The HAMMLP algorithm can decide on the success of the improvised solution. In fact it uses a HAMMLP quality measure, i.e. fitness function. The HAM algorithm is very similar to other optimization algorithms in that it is translated to maximize or minimize a measure obtained by the above-mentioned fitness function. The goal of such a fitness function should be similar to its functionality in optimization algorithms, besides that it is similar to those training methods in [41, 42]; they reduce the overall error. Thus, such a fitness function could utilize any of the MLPs error calculation formulas, or derive a new fitness function based on these formulas where the goal is to minimize this error.

In this work, MSE is used as the principal quality measure of the proposed HAM training algorithm. The training goal is to minimize the MSE until the maximum number of iterations has been reached.

The MSE is one of the most commonly used fitness functions; it is chosen as the main quality measure for the proposed MLP training algorithm (as this work is one that considers classification problems). The MSE, as the main fitness function, is the measure by which the food source vectors are to be sorted on from best to worst, with the best being the solution with the least MSE value. Thus in order to find an optimal solution, i.e. MLPs with acceptable weights & biases vector, its MSE value must be smaller than the worst one in the current food source memory vectors (FSMV). The fitness function is responsible for evaluating the quality of the solution in successive iterations. With the use of the fitness function a solution is picked that optimizes the quality of the solution.

In order to compute MSE, forward pass calculations must be performed first on the given MLP structure. This is a repetitive process that involves loading the entire training data set. This would require a process by which the network weights & bases (represented by the solution vector) are to be loaded into the MLP structure to implement such a computation. The MLP structure must be therefore flexible to allow loading different weight & bias vectors during the HAMMLP algorithm initialization and improvisation processes-such forward pass computation process is shown in Fig. 3.

The objective in training an MLP is to reach the highest classification, approximation, or prediction accuracy for both training and testing samples. A common metric for the evaluation of an MLP is the mean square error (MSE). In this work we apply the same method in [43–45] in order to calculate the fitness function. From Fig. 2(a) we note that MLPs with three layers contain one input, one hidden, and one output layer. The number of input nodes is equal to (n), the number of hidden nodes is equal to (h), and the number of output nodes is (m). The output of the ith hidden node is calculated as follows:

$$f\left(S_j\right) = 1/\left(1 + exp\left(-\left(\sum_{i=1}^{n} \mathcal{W}_{ij}.\mathcal{X}_i - \theta_j\right)\right)\right), j = 1, 2, \ldots, h \tag{11}$$

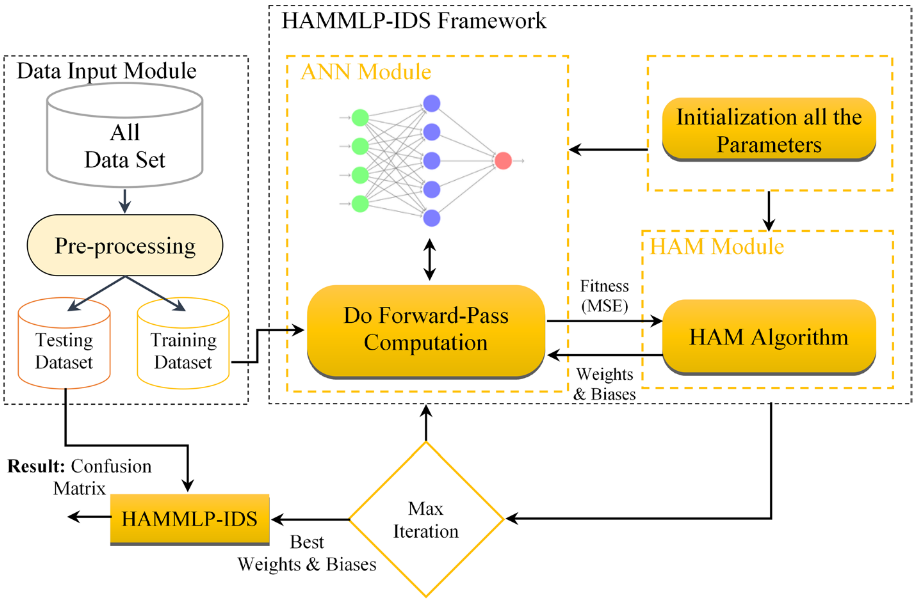


Fig. 3 The HAMMLP- IDS framework

where $S_j = \sum_{i=1}^{n} \mathcal{W}_{ij}.\mathcal{X}_i - \theta_j$, n is the number of the input nodes, $\mathcal{W}_{ij}$ is the connection weight from the ith node in the input layer to the jth node in the hidden layer,θ_j is the bias (threshold) of the jth hidden node, and $\mathcal{X}_i$ is the ith input. After calculating outputs of the hidden nodes, the final output can be defined as follows:

$$\mathcal{O}_k = \sum_{i=1}^{n} \mathcal{W}_{kj}.f\left(S_j\right) - \theta_k, k = 1, 2, \dots, m, \tag{12}$$

where $\mathcal{W}_{kj}$ is the connection weight from the jth hidden node to the kth output node and θ_k is the bias (threshold) of the kth output node. Finally, the learning error E (fitness function) is calculated as follows:

$$E_k = \sum_{i=1}^{m} \left(\mathcal{O}_i^k - d_i^k\right)^2 \tag{13}$$

$$\mathrm{E} = \sum_{\mathrm{k}=1}^{\mathrm{q}} \frac{\mathrm{E_k}}{\mathrm{q}} \tag{14}$$

where q is the number of training samples, d_i^k is the desired output of the jth input unit when the kth training sample is used, and $\mathcal{O}_i^k$ is the actual output of the ith input unit when the kth training sample is used. Therefore, the fitness function of the ith training sample can be defined as follows:

$$\text{Fitness}\left(x_{\mathrm{i}}\right) = \mathrm{E}\left(x_{\mathrm{i}}\right) \tag{15}$$

2.2.3 Training of MLP with the HAM algorithm

Figure 3 shows the flowchart of a generic HAM based MLP training approach for intrusion detection that utilizes the HAM algorithm introduced above. It illustrates the framework for our proposed HAMMLP-IDS and shows that it can be dissected into four main modules: parameter-initialization stage, data-input stage, neural network-training stage, and the HAM module.

The first stage of our proposal framework is the initialization of the parameters of the HAM algorithm and neural network module. The HAM algorithm has many variables, including population size (*SN*) representing the number of food sources or solutions in the population (*Solution Number*). Every solution in *SN* (*i* = *1, 2…, N*) represents a D-dimensional vector, *N* is the number of decision variables. The ranges' *lower* and *upper* limits are specified by two vectors x^L and x^U, both having the same length *SN*. The *limit* is used to diversify the search, to determine the number of allowable iterations for which each non-improved solution is to be abandoned, and additionally there is three control parameters: *limit1*, *limit2* which are used to adjust the mutation operators in the HAM algorithm and the maximum walk step parameter S_{max}. The Food Source Memory (*FSM*) is a matrix of the best solution vectors achieved so far. It is an augmented matrix of size $SN \times N$ comprised in each row as in Eq. (16). The *FSM* size is set prior to the running of the algorithm. Each Source vector is also associated with a source quality value (*fitness*) based on an objective function *f(x)*. The algorithm in Fig. 1 is similar to the optimization algorithm where it begins by initializing the HAM with random food source memory vectors representing candidate MLP weight vector values.

$$FSM = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ x_{31} & x_{32} & \cdots & x_{3N} \\ \cdots & \cdots & \ddots & \cdots \\ x_{SN1} & x_{SN2} & \cdots & x_{SNN} \end{bmatrix} \begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{SN}) \end{bmatrix} \tag{16}$$

On the other hand, the number of neurons in the layers of the neural network is determined by knowing the number of features for each dataset. An example of this: if the construction of a neural network based on the KDD Cup 99 dataset which has 41 of features, the number of neurons in the input layer is equal to 41, which also refer to the number of food sources or solutions vector in the HAM algorithm. The number of neurons in the hidden layer is calculated by using Kolmogorov's Theorem: one hidden layer and *2 N* + *1*, *N* representing the number of neurons in the input layer. The neurons in the output layer are equal to 1 for each IDS dataset used in this work, as all datasets used in our work are represented by binary data [0 or 1].

The second stage is an important one as it uses the data input module. This module is based on processing, filtering, and extracting the features from the raw data. One of the most important steps in this module is to divide the raw data into a training and testing set, which will then be used in the next neuron module as input data. Before we send the data into NN module, we must map the incoming inputs to turn into zero to one [0, 1] to make this data usable for the next module.

In the third stage, the ANN module begins to function after receiving training attributes for the input data from the previous module. This module is designed as a multilayer perceptron (MLP) which is a type of a feed-forward neural network. An MLP consists of three layers of neurons with an architecture containing one input layer, one hidden layer, and one output layer. The ANN module receives the outcoming data from the data input module which are considered as training pattern data (training dataset) for training the ANN. The training process in this module is implemented via sending the weights and biases to the HAM module.

The fourth stage in our proposal framework is using the HAM algorithm as a standalone system (black box) for generating new solutions which in turn are based on updated synaptic weights and biases (after each iteration). In each iteration of the training process the HAM module sends its individual solutions as a set of weights and biases into an ANN module; here the ANN module plays an important role by valuing these individual solutions based on a training data set and then returns their fitness values. The fitness function selected in this work to compute the fitness is the mean squared error (MSE), as it is a known fitness function and therefore good for use on the proposed HAM training algorithm.

The weights and biases are obtained by minimizing the error rate value of the MSE. The training process stops when the iterations reach the maximum number of iterations. Afterward the knowledge base (weights and biases) is updated. In the final step and after finishing training with the training dataset, we get the optimal solution from the HAM module. We use the optimal solution with the testing inputs which are fed from the testing dataset into the trained ANN module to predict the output. The testing process of the ANN can be seen as testing the predicted output with the closest match to any of the target classes.

Integrating various parts of the adaptation process that were presented in the previous section would lead to the HAM-based MLP training algorithm provided as a flow step in Fig. 4. The region of the flowchart enclosed by the dashed rectangle is the food source memory FSM initialization phase that essentially involves generating random weight and bias vectors from the allowable range of [*Lower, Upper*] and computing the relative MSE values for each by carrying out the forward pass computations.

The forward pass ingredients represented by the shaded side-framed rectangle is the pre-defined process provided previously in Fig. 4. This is also needed during the HAM training process to measure the quality (*fitness function*) of the newly improvised weight and bias vector.

3 Experimental Framework

The implementation and evaluation of the proposed framework was conducted on a Laptop with Core i5 2.4 GHz CPU and 8 GB RAM, and using a MATLAB R2014a running on a Windows 7. To evaluate the performance of HAMMLP-IDS framework we implemented three experiments, each using a different dataset for the offline evaluation of the IDSs, namely KDD Cup 99, ISCX 2012, and UNSW-NB15, against nine of the metaheuristic algorithms which have been adapted with ANNs; they are similar to our proposed framework.

3.1 Datasets Used for Experiments

There are many sets of data employed in assessing intrusion detection systems. One of the most noticeably antiquated is KDD Cup 99; although it is still used in much research hitherto. There are also a lot of datasets that have emerged in recent years in order to

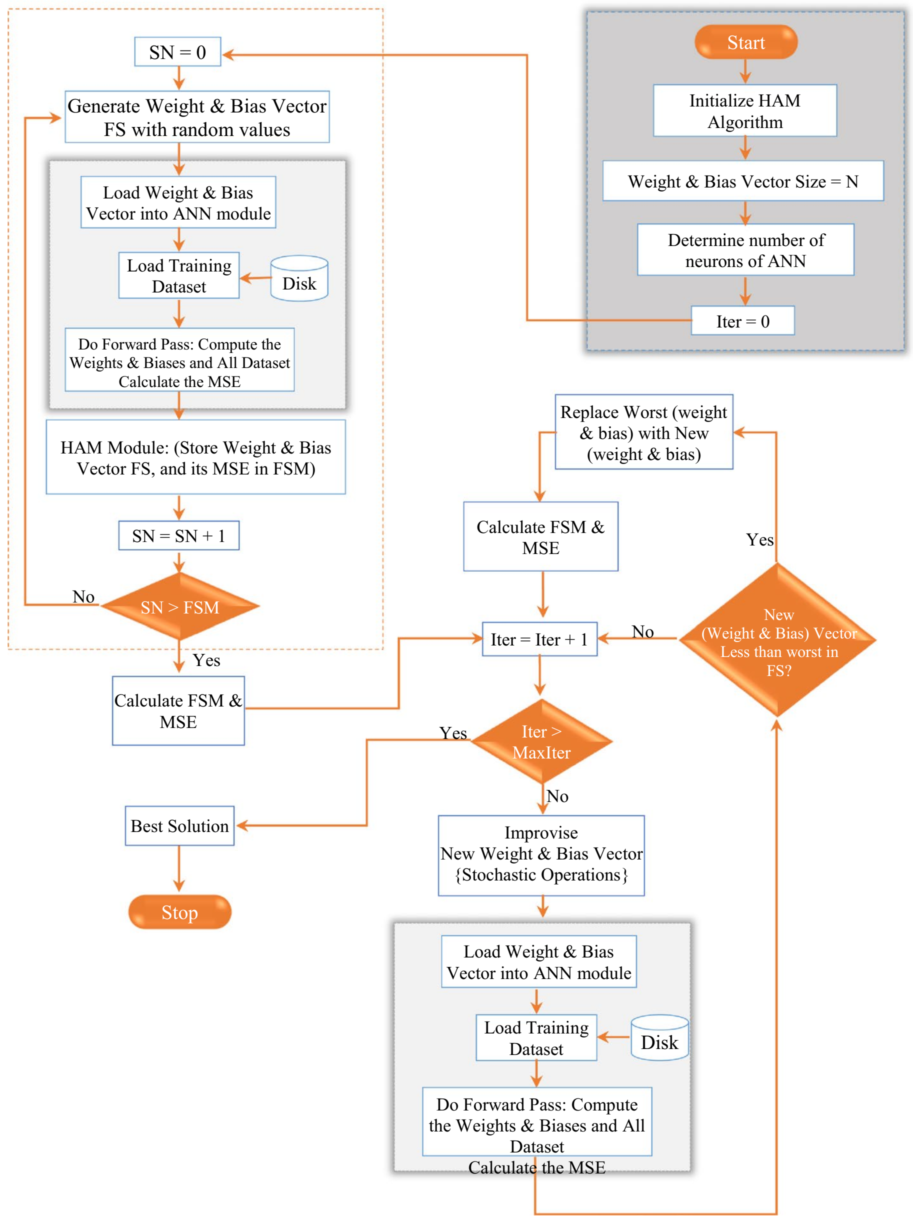

Start
Initialize HAM Algorithm
Weight & Bias Vector Size = N
Determine number of neurons of ANN
Iter = 0
SN = 0
Generate Weight & Bias Vector FS with random values
Load Weight & Bias Vector into ANN module
Load Training Dataset
Disk
Do Forward Pass: Compute the Weights & Biases and All Dataset Calculate the MSE
HAM Module: (Store Weight & Bias Vector FS, and its MSE in FSM)
SN = SN + 1
SN > FSM
No
Yes
Calculate FSM & MSE
Replace Worst (weight & bias) with New (weight & bias)
Calculate FSM & MSE
Yes
New (Weight & Bias) Vector Less than worst in FS?
No
Iter = Iter + 1
Iter > MaxIter
Yes
No
Best Solution
Stop
Improvise New Weight & Bias Vector {Stochastic Operations}
Load Weight & Bias Vector into ANN module
Load Training Dataset
Disk
Do Forward Pass: Compute the Weights & Biases and All Dataset Calculate the MSE

Fig. 4 The HAMMLP training algorithm flowchart

evaluate IDSs, such as the ISCX 2012 which was developed in 2012, and UNSW-NB15 which was developed in 2015.

3.1.1 KDD Cup 99 Dataset

The most popular and widely used dataset regarding the detection of intruders and anomalies is the aforementioned “KDD Cup 1999 Dataset” [46, 47], which was created and developed in 1999 by Lee and Stolfo [48]. It was built on the information obtained from MIT Lincoln Laboratory, under Defence Advanced Research Projects Agency (DARPA ITO) and Air Force Research Laboratory (AFRL/SNHS) sponsorship. It is made out of a set of records that can be approximated at 5 million. It represents TCP/IP packet connection, each packet connection contains 41 attributes (features) out of which 38 are numeric and 3 are symbolic. The NSL-KDD dataset is divided into training and testing sets, and it has four attack classes: DoS, U2R, R2L, and probe [49].

In this work we have used four sets of KDD Cup 99 datasets that were selected randomly by Zainal in 2007 and which have been used by many researchers [50–55]. Every single data set houses approximately 4000 records, nearly half of the data (50–55%) belonging to the normal category; the leftovers mere attacks. For training purposes dataset 1 is used, and for the purpose of testing datasets 2, 3, and 4 are utilized. The classes of all the datasets, number of records and the percentage of occurrence of the feature classes are tabulated in Table 1.

3.1.2 ISCX 2012 Dataset

In order to overcome the limitations of the KDD Cup 99 dataset, the ISCX 2012 intrusion evaluation dataset of an IDS at Information Security Center of Excellence (ISCX) is used in developing, testing and evaluating the performance of the proposed approach for intrusion and anomaly detection. The entire ISCX labeled dataset comprises nearly 1512000 packets with 20 features and covered seven days of network activity (i.e. normative and intrusive). This has been due to the fact approaches based on anomaly in particular are not reliable and suffer from an inaccurate assessment, comparison, and dissemination that arises from sufficient data scarcity. Many of these datasets are internal and cannot be shared because of privacy issues; others are completely anonymous and do not reflect current trends, or they lack some statistical characteristics.

The ISCX 2012 dataset is available in the packet capture form. Features are extracted from the packet format by using *tcptrace* utility (downloaded from http://www.tcptrace.org) and applying the following command: *tcptrace csv-l filename1.7z>filename1.csv*. Since the ready-made training and testing dataset is not available, and it is difficult to perform experiments on huge sets of data, we decided to select incoming packets for a particular host and particular days to validate the proposed approach as presented in Table 2. The

Table 1 Distribution statistics of the KDD Cup 99 training and testing datasets

Type	Dataset 1		Dataset 2		Dataset 3		Dataset 4	
	Actual	%	Actual	%	Actual	%	Actual	%
Dos	1000	25	1203	30	1050	26	903	23
Probe	563	14	400	10	491	12	475	12
R2L	122	3	55	1	30	1	62	2
U2R	15	0	45	1	30	1	10	0
Normal	2300	58	2300	57	2400	60	2550	64
Total	4000	100	4003	100	4001	100	4000	100

Table 2 Distribution statistics of the ISCX 2012 training and testing datasets

Date	Train ISCX 2012		Test ISCX 2012	
	Normal	Attack	Normal	Attack
11th	0	0	0	0
12th	2775	1388	1388	690
13th	27,144	13,572	3393	6786
14th	5028	2514	2514	1257
15th	12,459	6229	6229	3115
16th	0	0	0	0
17th	6938	3468	3468	1735
Total	54,344	27,171	16,992	13,583
	81,515		30,575	

training data contains 54344 normal traces and 27,171 attack traces while the testing data contains 16,992 normal traces and 13,583 additional attack traces. Further description on this dataset can be found in [56].

3.1.3 UNSW-NB15 Dataset

UNSW-NB15 dataset is hybrid of the contemporary synthesized attack activities of the network traffic and the real modern normal. This data set is available in the following link: http://www.cybersecurity.unsw.adfa.edu.au/ADFA%20NB15%20Datasets/. In 2015 the researchers Nour and Jill created the UNSW-NB15 hybrid of the abnormal network traffic and modern normal by using IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security. In both of these datasets (i.e. KDD'99, NSL-KDD and UNSW-NB15 datasets) has more than forty of the features. It is important to mention two datasets KDD'99 and NSL-KDD share only slight common features with UNSW-NB15 datasets, and the rest of the features are different, making it harder to compare them [57, 58].

The UNSW-NB15 dataset included nine different moderns attack types (compared to 14 attack types in KDD'99 and NSL-KDD datasets) and wide varieties of real and normal activities as well as 44 features inclusive of the class label consisting total of 2, 540, 044 records. The features UNSW-NB15 are classified into six groups, and they are as follows: Basic Features (BF), Content Features (CF), Flow Features (FF), Time Features (TF), Additional Generated Features (AGF) and class features. The Additional Generated Features are further classified into two sub-groups namely Connection Features and General Purpose Features.

The UNSW-NB15 dataset has been divided into two subsets, the first containing 175,341 records (56,000 Attacks and 119,341 Normal) represents the training datasets and the second contains 82,332 records (45,332 Attacks and 37,000 Normal) which represent the testing dataset including all attack types and normal traffic records. Both the training and testing datasets have 45 features, the distribution statistics of the UNSW-NB15 is shown in Table 3. It is important to note that the first feature (i.e. id) was not mentioned in the full UNSW-NB15 dataset features, and also the features scrip, sport, dstip, stime and ltime are missing in the Training and Testing dataset [59].

Table 3 Distribution statistics of the UNSW-NB15 training and testing datasets

No.	Type of attacks	Train UNSW-NB15	Test UNSW-NB15
		No. of record	No. of record
1	Analysis	2000	677
2	Backdoor	1746	583
3	Dos	12,264	4089
4	Exploits	33,393	11,132
5	Fuzzers	18,184	6062
6	Generic	40,000	18,871
7	Reconnaissance	10,491	3496
8	Shellcode	1133	378
9	Worms	130	44
10	Normal	56,000	37,000
Total		175,341	82,332

3.1.4 Preprocessing Dataset

Dataset processing is the most important step; it is implemented on all the datasets used in this work. The dataset processing is divided into two phases:

In the first phase we determine the data set that we will use to evaluate algorithms. Because the size of this dataset is too huge to load it into memory, we selected random sample records from the datasets. This random sample is divided into two a samples, the first sample is called training dataset, and the second sample is called testing dataset.

The second phase is to convert a symbolic value to a numeric value and then to normalize the numeric value. Because all the datasets include symbolic features, for example [protocol feature (e.g. *tcp*, *udp*, etc.), service feature (e.g. *telnet*, *ftp*, etc.), and flag feature]. These are incompatible with the classification method so we need to ensure that all the symbols are a set of numeric values, and to ensure that all class is represented as a numeric value.

(Normalization means to make the numeric values in the same range. The attributes of the numeric valuesin the dataset have to be normalized before using them in the training algorithm in order to provide regular semantics to the attribute values [60]. To normalize numeric values to a regular semantics between x_{min} and x_{max} (the min and max values for attribute x) first one would convert x_{min} and x_{max} to the new range, as per Eq. (17).

$$x_{new} = \frac{x_{current} - x_{min}}{x_{max} - x_{min}} \tag{17}$$

Identification of class where all the four data sets that are used in this work includes a class for each set of features, where it is either a normal connection or an attack. This means that each record in the dataset belongs to one of the major classes: Normal or Attack. The values for each class are mapped to a numeric value. More specifically the Normal class was mapped to the number 0 and the attack class to 1. All the data sets used to evaluate our proposal contain two features: either normal representing zero or abnormal representing one.

3.2 Algorithms and Parameters

In this work, study was carried out involving several algorithms for the fair analysis of the proposal, these are: Artificial Bee Colony algorithm (ABC) [61], Ant colony optimization (ACO) [62], Ant Lion Optimizer (ALO) [63], Elephant Herding Optimization (EHO) [64], Evolution Strategy (ES) [65], Harmony Search (HS) [66], monarch butterfly optimization (MBO) [67], A Sine Cosine Algorithm (SCA) [68], and Whale Optimization Algorithm (WOA) [69].

We set the common control parameters for all algorithms to the same values, including the population size *SN*, and the dimensionality of the search space *D*, which represents the number of the features in the dataset. The parameters of all algorithms used in this work are presented below:

- ABC parameter settings: The number of colony size is 50 employed bees and 50 onlooker bees as the colony size is 100. *limit* is set to 100.
- MBO parameter settings: There are many parameters for the MBO method. In this work we follow the setup in the original work of MBO, and set the butterfly adjusting rate BAR = 0.4167, max step $S_{max} = 1.0$, migration period $Peri = 1.2$, the migration ratio P = 0.4167 and the population size NP is the same as the colony size, being 50.
- ALO parameter settings: The population size in this method is also set to 50, and the vector a: linear decrease: which were set in this work is equal to 2.
- ACO parameter settings: The ACO method involves many parameters, which were set as follows: pheromone update constant $Q = 20$, local pheromone decay rate $\rho l = 0.5$, global pheromone decay rate $\rho g = 0.9$, exploration constant $q0 = 1$, pheromone sensitivity $s = 1$, visibility sensitivity $\beta = 5$ and initial pheromone value $\tau 0 = 1E - 6$.
- EHO parameter settings: This algorithm is based on three parameters: the scale factor $\alpha = 0.5$, $\beta = 0.1$, and the number of clan nClan = 5.
- HS parameter settings: The HS algorithm involves many parameters, which were set as follows: harmony memory consideration rate = 0.95, pitch adjustment rate = 0.1, fret width damp ratio = 0.1, Harmony Memory Size = 50, number of new harmonies = 50.
- ES parameter settings: This method is based on two parameters: the number of offspring $\lambda = 10$ produced in each generation and the standard deviation $\sigma = 1$ for changing solutions.
- SCA parameter settings: In this method the population size in this method is also set to 50, and the vector a: linear decrease: is set to 2.
- WOA parameter setting: There is one parameter for WOA algorithm, is Vector a: linear decrease: which were set in this work from 2 to 0.

3.3 Performance Measures for IDS

We compared and evaluated the proposed model based on three performance measures: The detection rate (*DR*), False Positive Rate (*FPR*), and Accuracy Rate (*Acc*). These main factors are calculated based on the main performance measurement for IDS, namely true positive (*TP*), false positive (*FP*), true negative (*TN*), and false negative (*FN*). These four main criteria have been collected from the confusion matrix. The confusion matrix has a dimension of the neural network, and it shows the classification results. Table 4 shows the confusion matrix for a 2-class classification.

Table 4 A confusion matrix for binary classification

	Actual		Total
	Normal	Attack	
Predicted			
Normal	TN	FN	TN + FN
Attack	FP	TP	FP + TP
Total	TN + FP	FN + TP	

The abbreviations of the confusion matrix a 2-class classification are as follows:

True positive (TP) Indicates the amount of attack data detected is actually attack data.
True negative (TN) Indicates the amount of putative normal data detected is indeed normal data.
False positive (FP) Represents the normal data that is detected as attack data.
False negative (FN) Represents the attack data that is detected as normal data.
ACC returns the global percentage of correctly classified instances to the total number of instances. It can be derived as follows:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{18}$$

FAR a shortened name for the false alarm rate; it is known also as the false positive rate. This factor computes by dividing the number of normal instances which are misclassified as anomalies by the overall number of normal instances. It can be computed as follows:

$$FAR = \frac{FP}{FP + TN} \tag{19}$$

DR a shortened name for the detection rate; it is also known as a true positive rate (*TPR*) or recall. It is the rate of the number of successfully classified anomalies to the overall number of instances. It refers to the reliability of the new approach in detecting abnormality from all of the abnormal instances. It can be computed as follows:

$$DR = \frac{TP}{TP + FN} \tag{20}$$

Specificity is also known as a true negative rate (*TNR*). It is the rate of the number of successfully classified non-anomalies to the total number of instances. It refers to the precision of the new approach in detecting non-anomalies from all of the non-anomalous instances. It can be computed as follows:

$$Specificity = \frac{TN}{TN + FP} \tag{21}$$

Precision is the ratio of negatives that are correctly identified as instances among the results of the new approach, which is another information retrieval term, and often is paired with "*Recall*".

$$Precision = \frac{TP}{TP + FP} \tag{22}$$

F-Measure is also known as F-score; it is a standard f-measure and computes as a tradeoff between precision and recall of both classes. It is computed in binary form, and is a measure of a test's accuracy. It is defined as the weighted harmonic mean of the precision and recall when testing a new approach.

$$F - measure = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{23}$$

4 Results and Discussion

The design elements explained in the previous section are used to implement multilayer-perceptron training algorithms: HAMMLP, named after the metaheuristics developed in our previous study [34, 35]. The MLP prefix highlights the fact the algorithm is being used to train a MLP (for the purpose of intrusion detection). This section is devoted to the results of evaluating these algorithms against a number of standard IDS datasets. Before listing the results, it is in order to explain the common setup in which all experimental evaluations were conducted.

First, each evaluation experiment compares ten algorithms, including the proposed algorithms, against three different IDS benchmark datasets: KDD Cup 99, ISCX 2012, and UNSW-NB15. These datasets range from the old to the very recent and were introduced in Sect. 3.1. For the KDD Cup 99 we used a subset of it; this subset is divided into 4 sets of data, one for training (Dataset 1) and another three (Dataset 2, Dataset 3, Dataset 4) for testing. Each set of data has around 4000 randomly chosen records of the original dataset. Also for the ISCX 2012 dataset there is a set of five experiments, as the dataset comprises subsets of traffic on five different days. The compared algorithms include the following list: ABC, ACO, ALO, EHO, ES, HS, MBO, SCA, and WOA. Each of these algorithms is applied for training a MLP and trained as well as tested using the above datasets (our experiments used training dataset and a testing dataset).

Second, the results of each experiment are presented in three forms: a table that lists the numerical values of the performance indicators for each algorithm; a plot that visually represents the performance of each algorithm; a set of confusion matrices for all algorithms with the proposed HAMMLP. Each algorithm was run for a maximum of 50 iterations, and the results are calculated based on 10 runs. The aforementioned performance indicators include the accuracy ACC, detection rate DR, false alarm rate FAR, precision, specificity, and F-Measure. The *FAR*, *DR*, and *ACC* are calculated based on the certain types of instances: true positives TP, false positives FP, true negatives TN, and false negatives FN. The definitions of these types are given in Sect. 3.3, while the definitions of all performance indicators are given in Eqs. (18–23).

Third, most of the benchmarking datasets contain data with different ranges; hence, there is a need to normalize feature values so that they can be effectively applied for training MLPs. The min–max normalization method was used, for which the formula is shown in Sect. 3.1.4 in Eq. (17) above.

Finally, one of the most important factors that influence the outcome of a neural network is the network structure in terms of the number of nodes in the hidden layer(s). For all experiments in this chapter the formula shown in Eq. (24) is used to determine the number of nodes in the hidden layer of the trained MLP. *N* is the number of attributes in the datasets (number of input nodes) and *H* is the number of hidden nodes.

$$No. of\ hidden\ nodes = 2 \times N + 1 \tag{24}$$

As mentioned earlier the results of evaluating the proposed algorithms comprise eight sets: one for the evaluation against the KDD Cup 99 dataset, one for NSL-KDD dataset, five sets for ISCX 2012 datasets (one subset per day for network traffic over five days), and a final set for the UNSW-NB15 dataset. Each set of results comprises a table, a plot, and a confusion matrix. The following subsections list the results by the datasets.

4.1 KDD Cup 99 Results

Using the classic KDD Cup 99 dataset, Tables 5, 6, 7 and 8 show the detailed performance measurements of every compared algorithm when trying to detect anomalies via a MLP trained by the algorithm itself. As mentioned previously we have four sets of data for this experiment that will be conducted for each data set four times (one for training and three for testing) to evaluate the performance of the proposed approach. The results of our proposed algorithm will be shown shaded in bold. Our experiments were only conducted on dataset 1 as training and datasets 2, 3 and 4 as testing. In order to make our experiences more effectively, we used data set 2, 3 and 4 also as training whilst other datasets are used as testing. Tables 5, 6, 7 and 8 show the results of twelve experiments which were conducted during the evaluation of our proposal by KDD Cup 99 dataset, Figs. 5 and 6, respectively.

The results in Tables 5, 6, 7 and 8 are calculated based on the definitions in Sect. 3.3 and Eqs. (18–23). The TP, TN, FN and FP measurements are averaged over 10 runs, the remaining columns are derived from these basic measurements. The most important indicators are the classification accuracy, the false alarm rate, and the detection rate. It is evident from the results that our proposed algorithms are among the top performing MLP trainers. In particular, HAMMLP is the absolute best among the 9 compared algorithms as shown in Table 9. Table 9 shows the average of the classification accuracy, the false alarm rate, and the detection rate for the four datasets. The results of our proposed algorithm will be shown shaded in bold. In the listed experiment it was ranked first with respect to accuracy at around 87.19% score, the first with respect to detection rate at 90.89% score, and the first best with respect to false alarm rate at a score of 0.1670 (this is shown in the Fig. 5).

The comparative performance of the 9 algorithms against the whole KDD Cup 99 dataset for the average of ACC, DR and FAR measurements is also depicted visually in Fig. 5. Another useful descriptor for the performance of classification models is the confusion matrix, which is a tabular layout to visualize the performance of supervised classifiers. The content of this matrix is the basic measurements of TP, TN, FN, and FP, which result from the mapping between the number of correct and wrong predictions of the classifier for both positive (attack) and negative (normal) instances of testing data. The general template of a confusion matrix is shown in Table 4.

4.2 ISCX 2012 Results

Similar to the previous set of results this section presents the numerical performance measurements and their visual representation for the compared 10 metaheuristics including our proposal when running against the ISCX 2012 intrusion detection benchmark dataset. As before, sample confusion matrices are also given for the proposed algorithm. However, the ISCX 2012 dataset is different from the other dataset as it was divided due to its large size into a number of subsets, each of which corresponds to the collected traffic in a single day.

Table 5 Evaluate the performance of 10 algorithms used to train an MLP, by training across dataset 1 and testing across data sets 2, 3, and 4

No.	Alg.	Testing dataset	TP	TN	FN	FP	ACC (%)	FAR	DR (%)	Spec.	Prec.	F-M
1	ABC	Dataset 2	2228	1224	72	479	86.24	0.2813	96.87	0.7187	0.8231	0.8900
2	ACO		2299	1117	1	586	85.34	0.3441	99.96	0.6559	0.7969	0.8868
3	ALO		2288	1378	12	325	91.58	0.1908	99.48	0.8092	0.8756	0.9314
4	EHO		2234	1431	66	272	91.56	0.1597	97.13	0.8403	0.8915	0.9297
5	ES		2255	868	45	835	78.02	0.4903	98.04	0.5097	0.7298	0.8367
6	**HAM**		**2210**	**1497**	**90**	**206**	**92.61**	**0.1210**	**96.09**	**0.8790**	**0.9147**	**0.9372**
7	HS		2290	1341	10	362	90.71	0.2126	99.57	0.7874	0.8635	0.9249
8	MBO		1642	1354	658	349	74.84	0.2049	71.39	0.7951	0.8247	0.7653
9	SCA		2267	1127	33	576	84.79	0.3382	98.57	0.6618	0.7974	0.8816
10	WOA		2241	1114	59	589	83.81	0.3459	97.43	0.6541	0.7919	0.8737
1	ABC	Dataset 3	2361	362	40	1238	68.06	0.7738	98.33	0.2263	0.6560	0.7870
2	ACO		2307	1141	94	459	86.18	0.2869	96.08	0.7131	0.8341	0.8930
3	ALO		2265	1165	136	435	85.73	0.2719	94.34	0.7281	0.8389	0.8881
4	EHO		1315	834	1086	766	53.71	0.4788	54.77	0.5213	0.6319	0.5868
5	ES		2400	791	1	809	79.76	0.5056	99.96	0.4944	0.7479	0.8556
6	**HAM**		**2368**	**1140**	**33**	**460**	**87.68**	**0.2875**	**98.63**	**0.7125**	**0.8373**	**0.9057**
7	HS		2038	1457	363	143	87.35	0.0894	84.88	0.9106	0.9344	0.8896
8	MBO		2037	1350	364	250	84.65	0.1563	84.84	0.8438	0.8907	0.8690
9	SCA		2401	549	0	1051	73.73	0.6569	100.00	0.3431	0.6955	0.8204
10	WOA		2219	739	182	861	73.93	0.5381	92.42	0.4619	0.7205	0.8097

Table 5 (continued)

No.	Alg.	Testing dataset	TP	TN	FN	FP	ACC (%)	FAR	DR (%)	Spec.	Prec.	F-M
1	ABC	Dataset 4	1604	1742	96	558	83.65	0.2426	94.35	0.7574	0.7419	0.8307
2	ACO		1690	1443	10	857	78.33	0.3726	99.41	0.6274	0.6635	0.7959
3	ALO		1610	1375	90	925	74.63	0.4022	94.71	0.5978	0.6351	0.7603
4	EHO		987	1250	713	1050	55.93	0.4565	58.06	0.5435	0.4845	0.5282
5	ES		1690	1102	10	1198	69.80	0.5209	99.41	0.4791	0.5852	0.7367
6	**HAM**		**1657**	**1646**	**43**	**654**	**82.58**	**0.2843**	**97.47**	**0.7157**	**0.7170**	**0.8262**
7	HS		1561	1929	139	371	87.25	0.1613	91.82	0.8387	0.8080	0.8596
8	MBO		1683	888	17	1412	64.28	0.6139	99.00	0.3861	0.5438	0.7020
9	SCA		1700	1076	0	1224	69.40	0.5322	100.00	0.4678	0.5814	0.7353
10	WOA		1609	686	91	1614	57.38	0.7017	94.65	0.2983	0.4992	0.6537

Table 6 Evaluate the performance of 10 algorithms used to train an MLP, by training across dataset 2 and testing across data sets 1, 3, and 4

No.	Alg.	Testing dataset	TP	TN	FN	FP	ACC (%)	FAR	DR (%)	Spec.	Prec.	F-M.
1	ABC	Dataset 1	1754	920	796	530	66.85	0.3655	68.78	0.6345	0.7680	0.7257
2	ACO		1499	1168	1051	282	66.68	0.1945	58.78	0.8055	0.8417	0.6922
3	ALO		1836	1185	714	265	75.53	0.1828	72.00	0.8172	0.8739	0.7895
4	EHO		2076	964	474	486	76.00	0.3352	81.41	0.6648	0.8103	0.8122
5	ES		1937	1225	613	225	79.05	0.1552	75.96	0.8448	0.8959	0.8222
6	**HAM**		**2249**	**1176**	**301**	**274**	**85.63**	**0.1890**	**88.20**	**0.8110**	**0.8914**	**0.8867**
7	HS		1790	1205	760	245	74.88	0.1690	70.20	0.8310	0.8796	0.7808
8	MBO		1951	876	599	574	70.68	0.3959	76.51	0.6041	0.7727	0.7689
9	SCA		2079	931	471	519	75.25	0.3579	81.53	0.6421	0.8002	0.8077
10	WOA		1897	810	653	640	67.68	0.4414	74.39	0.5586	0.7477	0.7458
1	ABC	Dataset 3	1961	1428	440	172	84.70	0.1075	81.67	0.8925	0.9194	0.8650
2	ACO		1873	1473	528	127	83.63	0.0794	78.01	0.9206	0.9365	0.8512
3	ALO		1966	1556	435	44	88.03	0.0275	81.88	0.9725	0.9781	0.8914
4	EHO		1905	1546	496	54	86.25	0.0338	79.34	0.9663	0.9724	0.8739
5	ES		2261	1116	140	484	84.40	0.3025	94.17	0.6975	0.8237	0.8787
6	**HAM**		**1949**	**1597**	**452**	**3**	**88.63**	**0.0019**	**81.17**	**0.9981**	**0.9985**	**0.8955**
7	HS		1949	1575	452	25	88.08	0.0156	81.17	0.9844	0.9873	0.8910
8	MBO		2348	1157	53	443	87.60	0.2769	97.79	0.7231	0.8413	0.9045
9	SCA		1919	1595	482	5	87.83	0.0031	79.93	0.9969	0.9974	0.8874
10	WOA		2206	1344	195	256	88.73	0.1600	91.88	0.8400	0.8960	0.9073

Table 6 (continued)

No.	Alg.	Testing dataset	TP	TN	FN	FP	ACC (%)	FAR	DR (%)	Spec.	Prec.	F-M.
1	ABC	Dataset 4	1447	1978	253	322	85.63	0.1400	85.12	0.8600	0.8180	0.8342
2	ACO		1398	2084	302	216	87.05	0.0939	82.24	0.9061	0.8662	0.8437
3	ALO		1517	2005	183	295	88.05	0.1283	89.24	0.8717	0.8372	0.8639
4	EHO		1387	2129	313	171	87.90	0.0743	81.59	0.9257	0.8902	0.8514
5	ES		1569	1461	131	839	75.75	0.3648	92.29	0.6352	0.6516	0.7639
6	**HAM**		**1626**	**1913**	**74**	**387**	**88.48**	**0.1683**	**95.65**	**0.8317**	**0.8077**	**0.8758**
7	HS		1439	2021	261	279	86.50	0.1213	84.65	0.8787	0.8376	0.8420
8	MBO		1566	2039	134	261	90.13	0.1135	92.12	0.8865	0.8571	0.8880
9	SCA		1536	1662	164	638	79.95	0.2774	90.35	0.7226	0.7065	0.7930
10	WOA		1456	2131	244	169	89.68	0.0735	85.65	0.9265	0.8960	0.8758

Table 7 Evaluate the performance of 10 algorithms used to train an MLP, by training across dataset 3 and testing across data sets 1, 2, and 4

No.	Alg.	Testing dataset	TP	TN	FN	FP	ACC (%)	FAR	DR (%)	Spec.	Prec.	F-M.
1	ABC	Dataset 1	1733	1197	817	253	73.25	0.1745	67.96	0.8255	0.8726	0.7641
2	ACO		1596	1059	954	391	66.38	0.2697	62.59	0.7303	0.8032	0.7035
3	ALO		2027	982	523	468	75.23	0.3228	79.49	0.6772	0.8124	0.8036
4	EHO		1394	1193	1156	257	64.68	0.1772	54.67	0.8228	0.8443	0.6637
5	ES		1300	977	1250	473	56.93	0.3262	50.98	0.6738	0.7332	0.6014
6	**HAM**		**2477**	**1022**	**73**	**428**	**87.48**	**0.2952**	**97.14**	**0.7048**	**0.8527**	**0.9082**
7	HS		2147	481	403	969	65.70	0.6683	84.20	0.3317	0.6890	0.7579
8	MBO		1988	961	562	489	73.73	0.3372	77.96	0.6628	0.8026	0.7909
9	SCA		2010	565	540	885	64.38	0.6103	78.82	0.3897	0.6943	0.7383
10	WOA		1563	1072	987	378	65.88	0.2607	61.29	0.7393	0.8053	0.6961
1	ABC	Dataset 2	2112	1592	188	111	92.53	0.0652	91.83	0.9348	0.9501	0.9339
2	ACO		2204	1556	96	147	93.93	0.0863	95.83	0.9137	0.9375	0.9478
3	ALO		2218	805	82	898	75.52	0.5273	96.43	0.4727	0.7118	0.8191
4	EHO		2224	1637	76	66	96.45	0.0388	96.70	0.9612	0.9712	0.9691
5	ES		1420	1232	880	471	66.25	0.2766	61.74	0.7234	0.7509	0.6776
6	**HAM**		**2197**	**1643**	**103**	**60**	**95.93**	**0.0352**	**95.52**	**0.9648**	**0.9734**	**0.9642**
7	HS		2142	1653	158	50	94.80	0.0294	93.13	0.9706	0.9772	0.9537
8	MBO		2128	1634	172	69	93.98	0.0405	92.52	0.9595	0.9686	0.9464
9	SCA		2137	1441	163	262	89.38	0.1538	92.91	0.8462	0.8908	0.9096
10	WOA		2121	1028	179	675	78.67	0.3964	92.22	0.6036	0.7586	0.8324

Table 7 (continued)

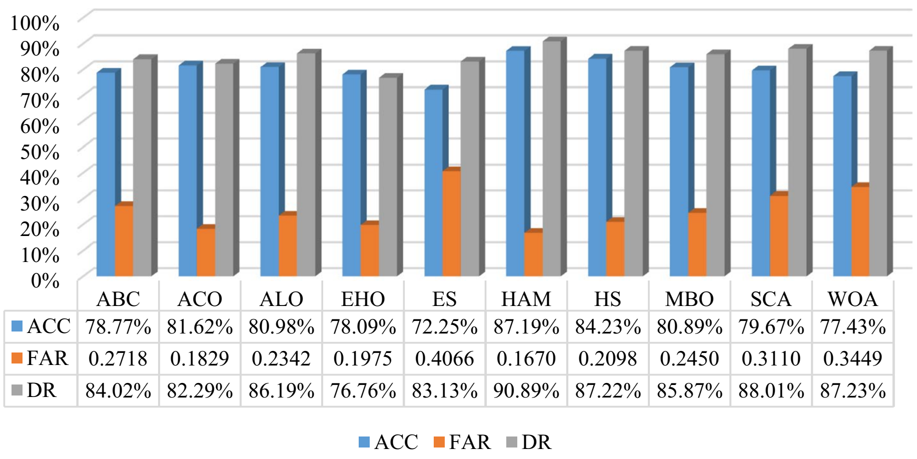
No.	Alg.	Testing dataset	TP	TN	FN	FP	ACC (%)	FAR	DR (%)	Spec.	Prec.	F-M.
1	ABC	Dataset 4	1447	1976	253	324	85.58	0.1409	85.12	0.8591	0.8171	0.8338
2	ACO		1408	2111	292	189	87.98	0.0822	82.82	0.9178	0.8817	0.8541
3	ALO		1616	1242	84	1058	71.45	0.4600	95.06	0.5400	0.6043	0.7389
4	EHO		1534	2038	166	262	89.30	0.1139	90.24	0.8861	0.8541	0.8776
5	ES		1156	1417	544	883	64.33	0.3839	68.00	0.6161	0.5669	0.6183
6	**HAM**		**1588**	**1547**	**112**	**753**	**78.38**	**0.3274**	**93.41**	**0.6726**	**0.6783**	**0.7859**
7	HS		1630	1906	70	394	88.40	0.1713	95.88	0.8287	0.8053	0.8754
8	MBO		1625	1632	75	668	81.43	0.2904	95.59	0.7096	0.7087	0.8139
9	SCA		1480	1949	220	351	85.73	0.1526	87.06	0.8474	0.8083	0.8383
10	WOA		1585	1488	115	812	76.83	0.3530	93.24	0.6470	0.6612	0.7737

Table 8 Evaluate the performance of 10 algorithms used to train an MLP, by training across dataset 4 and testing across data sets 1, 2, and 3

No.	Alg.	Testing dataset	TP	TN	FN	FP	ACC (%)	FAR	DR (%)	Spec.	Prec.	F-M.
1	ABC	Dataset 1	1496	655	1054	795	53.78	0.4300	58.67	0.4517	0.6530	0.6181
2	ACO		1558	1259	992	191	70.43	0.1615	61.10	0.8683	0.8908	0.7248
3	ALO		1315	1238	1235	212	63.83	0.1465	51.57	0.8538	0.8612	0.6451
4	EHO		1534	915	1016	535	61.23	0.3449	60.16	0.6310	0.7414	0.6642
5	ES		2017	538	533	912	63.88	0.6311	79.10	0.3710	0.6886	0.7363
6	**HAM**		**1791**	**1249**	**759**	**201**	**76.00**	**0.2094**	**70.24**	**0.8614**	**0.8991**	**0.7886**
7	HS		2018	901	532	549	72.98	0.5079	79.14	0.6214	0.7861	0.7887
8	MBO		1884	1120	666	330	75.10	0.3313	73.88	0.7724	0.8509	0.7909
9	SCA		1856	1070	694	380	73.15	0.3538	72.78	0.7379	0.8301	0.7756
10	WOA		1942	833	608	617	69.38	0.5037	76.16	0.5745	0.7589	0.7602
1	ABC	Dataset 2	2250	965	50	738	80.31	0.4334	97.83	0.5666	0.7530	0.8510
2	ACO		2094	1586	206	117	91.93	0.0687	91.04	0.9313	0.9471	0.9284
3	ALO		1971	1626	329	77	89.86	0.0452	85.70	0.9548	0.9624	0.9066
4	EHO		2073	1595	227	108	91.63	0.0634	90.13	0.9366	0.9505	0.9252
5	ES		2237	980	63	723	80.36	0.4245	97.26	0.5755	0.7557	0.8506
6	**HAM**		**2194**	**1617**	**106**	**86**	**95.20**	**0.0505**	**95.39**	**0.9495**	**0.9623**	**0.9581**
7	HS		2234	1451	66	252	92.06	0.1480	97.13	0.8520	0.8986	0.9336
8	MBO		2074	1609	226	94	92.01	0.0552	90.17	0.9448	0.9566	0.9284
9	SCA		2137	1441	163	262	89.38	0.1538	92.91	0.8462	0.8908	0.9096
10	WOA		2201	1195	99	508	84.84	0.2983	95.70	0.7017	0.8125	0.8788

Table 8 (continued)

No.	Alg.	Testing dataset	TP	TN	FN	FP	ACC (%)	FAR	DR (%)	Spec.	Prec.	F-M.
1	ABC	Dataset 3	1961	1428	440	172	84.70	0.1075	81.67	0.8925	0.9194	0.8650
2	ACO		1912	1352	489	248	81.58	0.1550	79.63	0.8450	0.8852	0.8384
3	ALO		2265	1431	136	169	92.38	0.1056	94.34	0.8944	0.9306	0.9369
4	EHO		1848	1451	553	149	82.45	0.0931	76.97	0.9069	0.9254	0.8404
5	ES		1935	804	466	796	68.46	0.4975	80.59	0.5025	0.7085	0.7541
6	**HAM**		**1963**	**1545**	**438**	**55**	**87.68**	**0.0344**	**81.76**	**0.9656**	**0.9727**	**0.8884**
7	HS		2039	1242	362	358	82.00	0.2238	84.92	0.7763	0.8506	0.8499
8	MBO		1888	1402	513	198	82.23	0.1238	78.63	0.8763	0.9051	0.8415
9	SCA		1952	1374	449	226	83.13	0.1413	81.30	0.8588	0.8962	0.8526
10	WOA		2204	1494	197	106	92.43	0.0663	91.80	0.9338	0.9541	0.9357



	ABC	ACO	ALO	EHO	ES	HAM	HS	MBO	SCA	WOA
🟦 ACC	78.77%	81.62%	80.98%	78.09%	72.25%	87.19%	84.23%	80.89%	79.67%	77.43%
🟧 FAR	0.2718	0.1829	0.2342	0.1975	0.4066	0.1670	0.2098	0.2450	0.3110	0.3449
⬜ DR	84.02%	82.29%	86.19%	76.76%	83.13%	90.89%	87.22%	85.87%	88.01%	87.23%

Fig. 5 The average of the evaluation variables (ACC, FAR, and DR) for the four datasets

Five days were used in the experimental evaluation of the tested metaheuristics: 12, 13, 14, 15 and 17. The respective subsets are named ISCX 2012-12, ISCX 2012-13, ISCX 2012-14, ISCX 2012-15, and ISCX 2012-17. Consequently, the results of this section include five sets for each of ISCX 2012 subsets.

Tables 10, 11, 12, 13 and 14 list the detailed performance measurements of the evaluated 10 algorithms, in the form of one table per day. The scores of our proposed algorithms are shaded in bold, and the last three columns show the rank of each algorithm with respect to the three main performance indicators: ACC, DR and FAR. The results of this dataset are quite unique when compared to the results of all other datasets (KDD Cup 99 and UNSW-NB15). On the one hand, several algorithms perform outstandingly on most of the days. For example, accuracies of 100% and false alarm rates of zero can be found on several rows of the 12th, 14th, 15th and 17th day. On the other hand, unlike the other datasets, our proposal outperformed by the other algorithms that are used to train the MLP on all days. The superior performance of our algorithm is, it is fair to say, remarkable (on this dataset particularly).

For the 12th day (Table 10), the results show that our algorithms have surprisingly achieved the perfect score of 100% accuracy and detection rate, as well as zero false alarms. It also outperforms the rest of the other algorithms in terms of specificity, precision, and F-measure. Besides our model (HAMMLP), there are two models which were able to record the perfect score of 100% detection rate: ABCMLP and HSMLP are all ranked the same. HSMLP records the second rank in the accuracy rate and false alarms followed directly by SCAMLP. Conversely, the EHOMLP and ESMLP algorithms recorded zero false alarms but were not able to achieve perfect results in terms of accuracy and detection rate. This result is unusual and seems particular for this set of data.

The results for the second set of data on the 13th day are less impressive (Table 11). In terms of accuracy, HAMMLP did the best at a score of 94.69%, followed by ESMLP at an accuracy of 90% and then ABCMLP at a score of 88.53%. EHeOMLP is ranked the tenth at an accuracy of 30.37%. In terms of detection rate, HAMMLP is ranked the first (92.07% DR), followed by WOAMLP (91.60% DR). EHO has ranked again the tenth at a detection rate of 0.66%. In terms of false alarm rate, the HAMMLP is ranked the first and followed

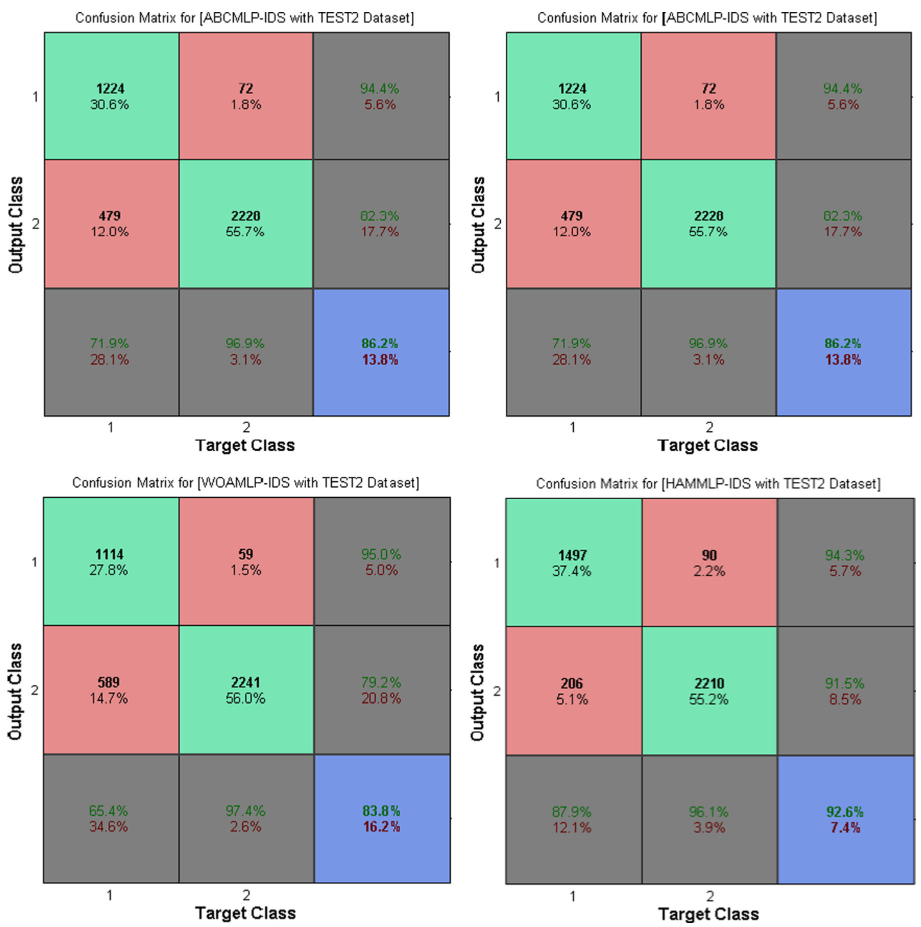


Fig. 6 The confusion matrices for ABCMLP, MBOMLP, WOAMLP, and HAMMLP against the KDD Cup 99 dataset by training across dataset 1 and testing across dataset 2

by HSMLP and MBOMLP algorithm. Overall, combining the three performance indicators (operating under the assumption they have equal importance), HAMMLP performed the most superior, followed by ESMLP then ABCMLP with respect to the 13th day ISCX 2012 dataset.

On the dataset of the 14th day (Table 12), our proposed model is ranked at the top with superior performance across the three main performance indicators: ACC, DR and FAR. HAMMLP being the best performing algorithms here with the maximum score of 100% accuracy, 100% detection rate, and zero false alarm rate. The WOAMLP and SCAMLP follow our proposal only across the two main performance indicators: ACC and DR, where the score of the accuracy was 98.91%, 98.14% respectively, while the detection rate was 99.76%, 99.60% respectively. On the contrary, the false alarm rate on this set scored the second and third ranks in favor of other algorithms which are ESMLP and ALOMLP; the score of the FAR was 0.0012, 0.0016 respectively.

The results of the following day (Table 13) showed our algorithm still stable in its ability to obtain the best results compared with the rest of the algorithms. In addition, the other

Table 9 Performance measurements of 10 algorithms used to train an MLP to detect anomalies in the KDD Cup 99 dataset

Alg.	Dataset 1			Dataset 2		
	ACC (%)	FAR	DR (%)	ACC (%)	FAR	DR (%)
ABC	79.31	0.4325	96.52	79.06	0.2043	78.53
ACO	83.28	0.3345	98.48	79.12	0.1226	73.01
ALO	83.98	0.2883	96.17	83.87	0.1128	81.04
EHO	67.06	0.3650	69.99	83.38	0.1478	80.78
ES	75.86	0.5056	99.14	79.73	0.2742	87.47
HAM	**87.62**	**0.2309**	**97.39**	**87.58**	**0.1197**	**88.34**
HS	88.44	0.1544	92.09	83.15	0.1020	78.67
MBO	74.59	0.3250	85.08	82.80	0.2621	88.81
SCA	75.97	0.5091	99.52	81.01	0.2128	83.94
WOA	71.71	0.5286	94.83	82.03	0.2250	83.97
Alg.	Dataset 3			Dataset 4		
	DR (%)	ACC	FAR (%)	DR (%)	ACC	FAR (%)
ABC	83.79	0.1268	81.63	72.93	0.3236	79.39
ACO	82.76	0.1460	80.41	81.31	0.1284	77.26
ALO	74.06	0.4367	90.33	82.02	0.0991	77.20
EHO	83.48	0.1100	80.53	78.44	0.1672	75.75
ES	62.50	0.3289	60.24	70.90	0.5177	85.65
HAM	**87.26**	**0.2193**	**95.36**	**86.29**	**0.0981**	**82.46**
HS	82.97	0.2896	91.07	82.35	0.2932	87.06
MBO	83.04	0.2227	88.69	83.11	0.1701	80.90
SCA	79.83	0.3056	86.27	81.89	0.2163	82.33
WOA	73.79	0.3367	82.25	82.21	0.2894	87.88

Table 10 Performance measurements of 10 algorithms used to train an MLP to detect anomalies in the ISCX 2012-12 dataset

No.	Alg.	ACC (%)	FAR	DR (%)	Spec.	Prec.	F-M.	ACC-rank	FAR-rank	DR-rank
1	ABC	93.61	0.0958	100	0.9042	0.8392	0.9126	4	10	1
2	ACO	65.61	0.0159	0	0.9841	0	–	9	8	9
3	ALO	65.23	0.0216	0	0.9784	0	–	10	9	9
4	EHO	71.18	0	13.54	1	1	0.2386	8	1	8
5	ES	79.54	0	38.62	1	1	0.5572	6	1	7
6	**HAM**	**100**	**0**	**100**	**1**	**1**	**1**	**1**	**1**	**1**
7	HS	99.71	0.0043	100	0.9957	0.9914	0.9957	2	5	1
8	MBO	88.42	0	65.27	1	1	0.7899	5	1	5
9	SCA	95.82	0.0144	90.35	0.9856	0.9691	0.9351	3	7	4
10	WOA	79.30	0.0072	39.34	0.9928	0.9647	0.5589	7	6	6

Table 11 Performance measurements of 10 algorithms used to train an MLP to detect anomalies in the ISCX 2012-13 dataset

No.	Alg.	ACC (%)	FAR	DR (%)	Spec.	Prec.	F-M.	ACC-rank	FAR-rank	DR-rank
1	ABC	88.53	0.1568	90.63	0.8432	0.9204	0.9133	3	8	3
2	ACO	66.64	0.0981	54.86	0.9019	0.9179	0.6868	8	4	9
3	ALO	88.51	0.1232	88.92	0.8768	0.9352	0.9116	4	7	5
4	EHO	30.37	0.1023	0.66	0.8977	0.1148	0.0125	10	6	10
5	ES	90.00	0.0984	89.92	0.9016	0.9481	0.9230	2	5	4
6	**HAM**	**94.69**	**0.0006**	**92.07**	**0.9994**	**0.9997**	**0.9586**	**1**	**1**	**1**
7	HS	75.20	0.0083	63.22	0.9917	0.9935	0.7727	7	2	7
8	MBO	82.40	0.0124	74.23	0.9876	0.9917	0.8491	6	3	6
9	SCA	57.73	0.3805	55.61	0.6195	0.7451	0.6369	9	10	8
10	WOA	88.47	0.1780	91.60	0.8220	0.9114	0.9137	5	9	2

Table 12 Performance measurements of 10 algorithms used to train an MLP to detect anomalies in the ISCX 2012-14 dataset

No.	Alg.	ACC (%)	FAR	DR (%)	Spec.	Prec.	F-M.	ACC-rank	FAR-rank	DR-rank
1	ABC	96.18	0.0072	89.98	0.9928	0.9843	0.9401	4	6	4
2	ACO	68.76	0.0374	13.76	0.9626	0.6479	0.2270	9	10	8
3	ALO	69.93	0.0016	10.10	0.9984	0.9695	0.1830	8	3	9
4	EHO	94.35	0.0052	84.09	0.9948	0.9879	0.9085	5	5	5
5	ES	68.07	0.0012	4.46	0.9988	0.9492	0.0851	10	2	10
6	**HAM**	**100**	**0**	**100**	**1**	**1**	**1**	**1**	**1**	**1**
7	HS	89.21	0.0179	71.20	0.9821	0.9521	0.8147	7	8	7
8	MBO	93.56	0.0040	81.46	0.9960	0.9903	0.8939	6	4	6
9	SCA	98.14	0.0259	99.60	0.9741	0.9506	0.9728	3	9	3
10	WOA	98.91	0.0151	99.76	0.9849	0.9706	0.9839	2	7	2

Table 13 Performance measurements of 10 algorithms used to train an MLP to detect anomalies in the ISCX 2012-15 dataset

No.	Alg.	ACC (%)	FAR	DR (%)	Spec.	Prec.	F-M.	ACC-rank	FAR-rank	DR-rank
1	ABC	76.15	0.1088	50.21	0.8912	0.6976	0.5839	10	8	10
2	ACO	90.15	0.1227	94.99	0.8773	0.7948	0.8655	9	10	8
3	ALO	97.77	0.0331	99.94	0.9669	0.9379	0.9677	4	5	3
4	EHO	96.78	0	90.34	1	1	0.9492	6	1	9
5	ES	96.07	0.0588	99.97	0.9412	0.8948	0.9444	7	7	2
6	**HAM**	**100**	**0**	**100**	**1**	**1**	**1**	**1**	**1**	**1**
7	HS	99.06	0.0090	98.97	0.9910	0.9822	0.9859	3	3	6
8	MBO	99.14	0.0119	99.81	0.9881	0.9768	0.9873	2	4	4
9	SCA	91.28	0.1140	96.63	0.8860	0.8091	0.8808	8	9	7
10	WOA	96.83	0.0466	99.81	0.9534	0.9147	0.9546	5	6	4

Table 14 Performance measurements of 10 algorithms used to train an MLP to detect anomalies in the ISCX 2012-17 dataset

No.	Alg.	ACC (%)	FAR	DR (%)	Spec.	Prec.	F-M.	ACC-rank	FAR-rank	DR-rank
1	ABC	62.23	0.0663	0	0.9337	0	–	10	9	10
2	ACO	63.81	0.0467	0.81	0.9533	0.0795	0.0147	9	8	8
3	ALO	98.71	0.0193	100	0.9807	0.9628	0.9811	1	7	1
4	EHO	67.06	0	1.21	1	1	0.0239	6	1	6
5	ES	66.73	0	0.23	1	1	0.0046	8	1	9
6	**HAM**	**98.58**	**0.0173**	**99.19**	**0.9827**	**0.9663**	**0.9790**	**2**	**6**	**2**
7	HS	67.02	0.0006	1.21	0.9994	0.9130	0.0239	7	4	6
8	MBO	84.78	0.1101	76.37	0.8899	0.7762	0.7699	5	10	5
9	SCA	97.12	0.0124	93.83	0.9876	0.9743	0.9560	4	5	4
10	WOA	98.17	0	94.52	1	1	0.9719	3	1	3

algorithms were able to record the best results in this set by comparing it with the previous sets; excepting the ABCMLP algorithm which records the worst result across the two main performance indicators: ACC and DR.

Lastly, the results of the 17th day (Table 14) are quite different from the previous datasets, deviating from their normal pattern. In this set, ALOMLP got the first rank in performance with an accuracy of 98.71%, a detection rate of 100% and got on the seventh rank in the performer with false alarms of 0.0193. Our proposal HAMMLP follows closely it records the second rank in the performer with an accuracy of 98.58%, a detection rate of 99.19% and it records the sixth rank for false alarms by 0.0173. HAMMLP show relatively inferior performance compared to their previous scores and to other algorithms with respect to this final subset of data. While the ACC and DR for WOAMLP recorded the third rank with 98.17%, 94.52% respectively and the first rank with zero. These last results suggest an important point: the HAMMLP shows more impressive results than the other nine algorithms with respect to the ISCX 2012 dataset. It is generally more consistent, across the various data subsets, and sometimes even the absolute best. This conclusion is also consistent with the results from the other benchmarking datasets, as will be shown in subsequent sections. Figure 7 show the confusion matrices from MATLAB program for the best three algorithms against the five days. It is a tabular layout to visualize the performance of supervised classifiers. The content of this matrix is the basic measurements of TP, TN, FN and FP, which result from the mapping between the number of correct and wrong predictions of the classifier for both positive (attack) and negative (normal) instances of testing data. The general template of a confusion matrix is shown in Table 4.

4.3 UNSW-NB15 Results

Finally, this section presents the numerical performance measurements and their visual representation for the most recent UNSW-NB15 intrusion detection benchmark dataset. A sample of four confusion matrices is also given for our proposed algorithms in addition to other best performing algorithms. These results are shown in Table 15, Figs. 8 and 9, respectively. The results of our proposed algorithm will be shown shaded in bold.

Confirming all previous results, HAMMLP is the top performing algorithm in this dataset, too. Looking at the three last columns of Table 15 showing the ranks per ACC, DR,

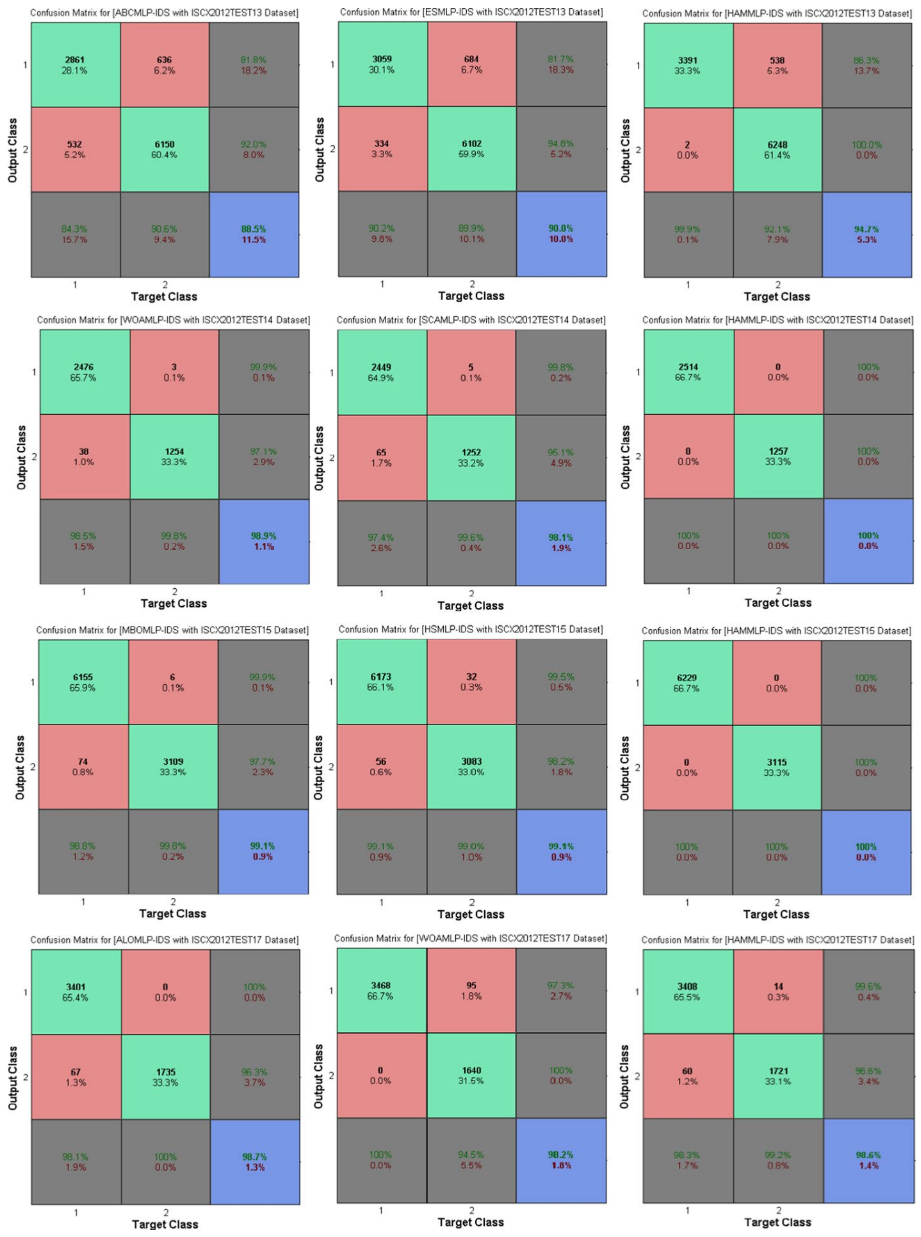


Fig. 7 The confusion matrices for ABCMLP, ALOMLP, ESMLP, MBOMLP, SCAMLP, WOAMLP, and HAMMLP against the ISCX 2012 dataset

and FAR, HAMMLP is ranked first in ACC and DR, and FAR at scores of 95.72%, 96.41% and 0.0507, respectively. HAMMLP is followed by WOAMLP at an accuracy of 92.52%, detection rate of 92.01% and false alarm rate of 0.0701. This algorithm is ranked 2nd with respect to ACC, 3rd with respect to DR, and 3rd with respect to FAR. SCAMLP closely follows with ACC, DR and FAR of 90.75%, 92.50% and 0.1140, respectively. It is ranked

Table 15 Performance measurements of 10 algorithms used to train an MLP to detect anomalies in the UNSW-NB15 dataset

No.	Alg.	ACC (%)	FAR	DR (%)	Spec.	Prec.	F-M.	ACC-rank	FAR-rank	DR-rank
1	ABC	85.40	0.1208	83.35	0.8792	0.8942	0.8628	7	6	8
2	ACO	83.80	0.0822	77.29	0.9178	0.9201	0.8401	9	4	10
3	ALO	89.75	0.0594	85.86	0.9406	0.9414	0.8981	4	2	7
4	EHO	84.53	0.2346	91.05	0.7654	0.8262	0.8663	8	10	5
5	ES	80.34	0.1793	78.23	0.8207	0.7827	0.7825	10	9	9
6	**HAM**	**95.72**	**0.0507**	**96.41**	**0.9493**	**0.9558**	**0.9599**	**1**	**1**	**1**
7	HS	87.93	0.1316	88.81	0.8684	0.8921	0.8901	5	7	6
8	MBO	87.54	0.1713	91.35	0.8287	0.8672	0.8898	6	8	4
9	SCA	90.75	0.1140	92.50	0.8860	0.9086	0.9167	3	5	2
10	WOA	92.52	0.0701	92.01	0.9299	0.9231	0.9216	2	3	3

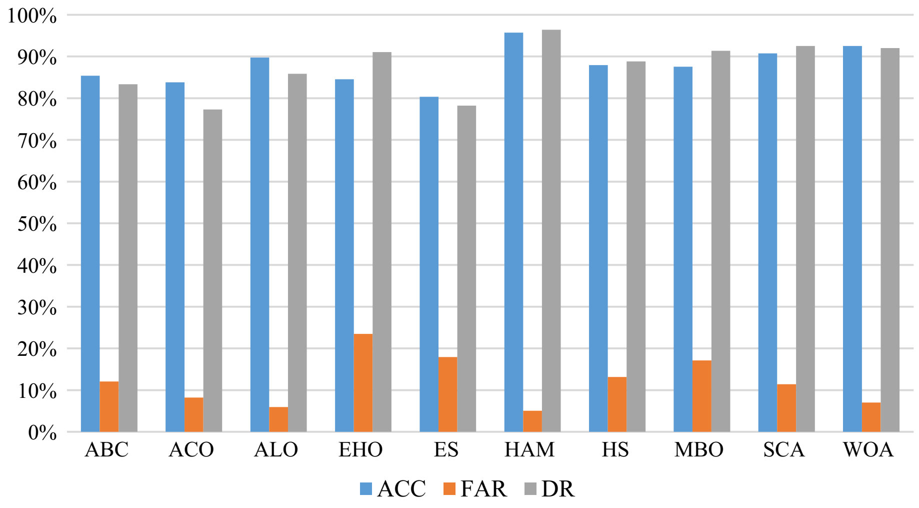


Fig. 8 The performance of 10 MLP trainer algorithms for the UNSW-NB15 dataset

3rd in ACC, 5th in DR and 2nd with respect to FAR. ESMLP; on the other hand, it performs relatively poorly with the UNSW-NB15 dataset compared to the remaining algorithms with an accuracy of 80.34%, a detection rate of 78.23% and a false alarm rate of 0.1793, respectively.

Consistently, across all result sets in these experiments for the different benchmarking datasets, HAMMLP outperformed all other algorithms in the set of 10 compared metaheuristics. This result is of profound importance and relevant to this article; for as the evaluation of the algorithm as an MLP trainer was conducted against actual and various IDS benchmarking datasets, which serve the very purpose of the article, it ought to lead inexorably to the selection of HAMMLP for the last step towards a complete IDS solution: feature selection.

The algorithms for training ANNs do not need only strong exploration ability, but also precise exploitation ability. The results of the classification accuracy, DR and FAR

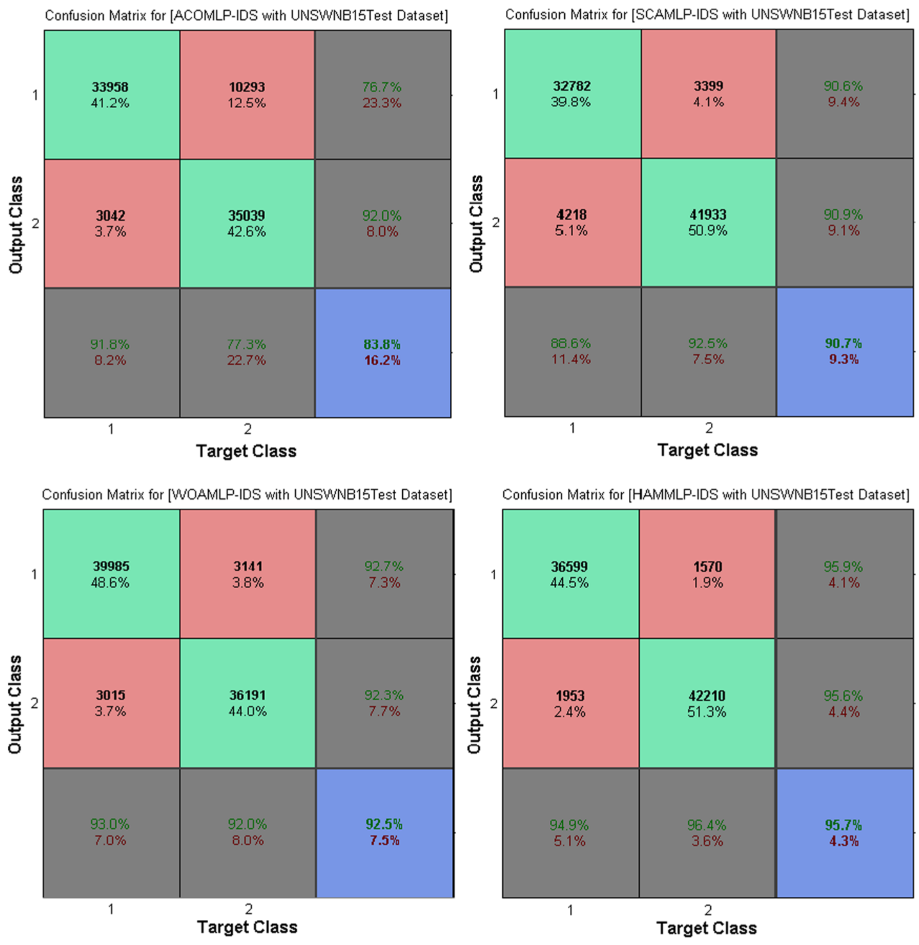


Fig. 9 The confusion matrices for ACOMLP, SCAMLP, WOAMLP, and HAMMLP against the UNSW-NB15 dataset

obtained by ABCMLP, ACOMLP, ALOMLP, EHOMLP, ESMLP, HAMMLP, HSMLP, MBOMLP, SCAMLP, and WOAMLP, show that HAMMLP performs far better than the others due to more precise exploitation ability afforded by the HAM algorithm, while others still suffer from the problem of becoming trapped in local minima. This weakness means that the approach of others leads to unstable performance. The results obtained by HAMMLP prove that it has both strong exploitation and good exploration abilities. In other words, the strength of some the other algorithms have been successfully utilized, giving outstanding performance in the MLP training. This result means that HAMMLP is capable of solving the hitherto problem of becoming trapped in local minima, giving a fast convergence speed.

The comparison of the performance results of the proposed approach and other approaches for the KDD Cup 99, ISCX 2012 dataset and UNSW-NB15 datasets are shown in Table 16, respectively. The proposed model clearly performs the best in terms of ACC, DR and FAR. The data correctly classified by the proposed approach are more than those

Table 16 Performance comparison of HAMMLP approach with other intrusion approaches

Dataset	Method	Reference	ACC (%)	DR (%)	FAR
KDD Cup 99	SVDF	[50–55]	–	64.3	0.39
	MARS		–	79.27	11.88
	Rough set		–	64.71	0.26
	Rough-DPSO		–	79.2	10.1
	BA		–	90.22	4.56
	BA-MC		–	89.11	0.004
	Rough set		–	64.71	0.26
	Markov model		–	68.3	0.48
	Rough-DPSO		–	79.2	10.1
	GDA-SVM		–	90.84	0.17
	HAMMLP-IDS		**87.19**	**90.89**	**0.167**
UNSW-NB15	ANN (MLP)	[58]	81.34	–	21.13
	LR (logistic regression)		83.15	–	18.84
	DT (decision tree)		85.56	–	15.78
	NB (Naive Bayes)		82.07	–	18.56
	EM (expectation max)		78.47	–	23.79
	Ramp-KSVCR	[70]	93.52	98.68	02.46
	GA-LR	[71]	81.42	–	6.39
	Dendron	[72]	84.33	63.76	2.61
	HAMMLP-IDS		**96.86**	**97.54**	**0.04**
ISCX 2012	MLP	[73]	–	90.6	0.660
	Bagged MLP		–	90.6	0.660
	Boosted MLP		–	94.7	0.083
	AMGA2-MLP		–	97	0.024
	AMGA2-NB		–	94.5	0.070
	RFA-bigram	[74]	92.9	89.6	2.6
	NBC	[75]	88.2	85.0	33.7
	KMC + NBC	[75]	99	98.8	2.2
	HAMMLP-IDS		**98.65**	**98.25**	**0.003**

correctly classified by the static approaches. Furthermore, HAMMLP exhibits a significantly lower FAR. Therefore, the proposed method noticeably offers the greatest, exploration and precise exploitation capabilities. The proposed training algorithm HAM is effective and feasible for application to IDS research.

5 Conclusion and Future Work

In this article a new approach for an intrusion detection system, namely, a HAM trained MLP is proposed and presented. The study has mainly focused on the applicability of the new hybrid algorithm HAM to train MLP. The confusion matrix is the basic measurements of TP, TN, FN, and FP of the proposed model; they have been obtained using the KDD Cup 99, ISCX 2012, and UNSW-NB15 datasets. The performance of the model was compared with those of popular intrusion detection techniques. Also it was compared with nine optimization algorithms that are used to train MLP, these being ABC, ACO, ALO, EHO, ES, HS, MBO, SCA, and WOA. The HAMMLP showed it was a good candidate that trained with the KDD Cup 99, ISCX 2012, and UNSW-NB15 datasets, attaining a detection rate of 90.89%, 98.25%, and 96.41%, respectively. These values are higher than those currently obtained by other methods tested using the KDD Cup 99, ISCX 2012, and UNSW-NB15 datasets. The results evidence the potential applicability of the proposed model for developing practical IDSs. However, this study has mainly evaluated the models according to the feature intrusion detection datasets; an adequate feature selection technique has not yet been selected. Therefore future work will focus on minimizing the number of selected features and applying the proposed model to develop an effective IDS.

Acknowledgements This research has been funded by Universiti Sains Malaysia under USM Fellowship [APEX (308/AIPS/415401) (1002/CIPS/ATSG4001)]. And by the RUI Grant, Account No. [1001/PKOMP/8014017] also under the Universiti Sains Malaysia.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

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

References

1. Anderson JP (1980) Computer security threat monitoring and surveillance. Technical report, James P. Anderson Company
2. Denning DE (1987) An intrusion-detection model. IEEE Trans Software Eng 2:222–232
3. Ghanem WAH, Belaton B (2013) Improving accuracy of applications fingerprinting on local networks using NMAP-AMAP-ETTERCAP as a hybrid framework. In: 2013 IEEE international conference on control system, computing and engineering. IEEE, pp 403–407
4. Inayat Z, Gani A, Anuar NB, Anwar S, Khan MK (2017) Cloud-based intrusion detection and response system: open research issues, and solutions. Arab J Sci Eng 42(2):399–423
5. Narayana GS, Vasumathi D (2018) An attributes similarity-based K-medoids clustering technique in data mining. Arab J Sci Eng 43(8):3979–3992
6. Fisch D, Hofmann A, Sick B (2010) On the versatility of radial basis function neural networks: a case study in the field of intrusion detection. Inf Sci 180(12):2421–2439
7. Ding S, Ma G, Shi Z (2014) A rough RBF neural network based on weighted regularized extreme learning machine. Neural Process Lett 40(3):245–260
8. Hajimirzaei B, Navimipour NJ (2019) Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm. ICT Exp 5(1):56–59
9. Li H (2016) Research on prediction of traffic flow based on dynamic fuzzy neural networks. Neural Comput Appl 27(7):1969–1980
10. Alauthaman M, Aslam N, Zhang L, Alasem R, Hossain MA (2018) A P2P Botnet detection scheme based on decision tree and adaptive multilayer neural networks. Neural Comput Appl 29(11):991–1004
11. Pillutla H, Arjunan A (2019) Fuzzy self organizing maps-based DDoS mitigation mechanism for software defined networking in cloud computing. J Ambient Intell Humaniz Comput 10(4):1547–1559
12. Aguayo L, Barreto GA (2018) Novelty detection in time series using self-organizing neural networks: a comprehensive evaluation. Neural Process Lett 47(2):717–744
13. Pozi MSM, Sulaiman MN, Mustapha N, Perumal T (2016) Improving anomalous rare attack detection rate for intrusion detection system using support vector machine and genetic programming. Neural Process Lett 44(2):279–290
14. Thaseen IS, Kumar CA, Ahmad A (2019) Integrated intrusion detection model using chi square feature selection and ensemble of classifiers. Arab J Sci Eng 44(4):3357–3368
15. Catania CA, Bromberg F, Garino CG (2012) An autonomous labeling approach to support vector machines algorithms for network traffic anomaly detection. Expert Syst Appl 39(2):1822–1829

16. Vijayanand R, Devaraj D, Kannapiran B (2018) Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection. Comput Secur 77:304–314
17. Zou X, Cao J, Guo Q, Wen T (2018) A novel network security algorithm based on improved support vector machine from smart city perspective. Comput Electr Eng 65:67–78
18. Shams EA, Rizaner A (2018) A novel support vector machine based intrusion detection system for mobile ad hoc networks. Wireless Netw 24(5):1821–1829
19. Asghari S, Navimipour NJ (2019) Resource discovery in the peer to peer networks using an inverted ant colony optimization algorithm. Peer-to-Peer Netw Appl 12(1):129–142
20. Kolias C, Kambourakis G, Maragoudakis M (2011) Swarm intelligence in intrusion detection: a survey. Comput Secur 30(8):625–642
21. Ozturk C, Karaboga D (2011) Hybrid artificial bee colony algorithm for neural network training. In: 2011 IEEE congress of evolutionary computation (CEC). IEEE, pp 84–88
22. Garro BA, Sossa H, Vázquez RA (2011) Artificial neural network synthesis by means of artificial bee colony (abc) algorithm. In: 2011 IEEE congress of evolutionary computation (CEC). IEEE, pp 331–338
23. Ojha VK, Abraham A, Snášel V (2017) Metaheuristic design of feedforward neural networks: a review of two decades of research. Eng Appl Artif Intell 60:97–116
24. Razmjooy N, Sheykhahmad FR, Ghadimi N (2018) A hybrid neural network–world cup optimization algorithm for melanoma detection. Open Med 13(1):9–16
25. Hagh MT, Ebrahimian H, Ghadimi N (2015) Hybrid intelligent water drop bundled wavelet neural network to solve the islanding detection by inverter-based DG. Front Energy 9(1):75–90
26. Abedinia O, Amjady N, Ghadimi N (2018) Solar energy forecasting based on hybrid neural network and improved metaheuristic algorithm. Comput Intell 34(1):241–260
27. Abusnaina AA, Abdullah R, Kattan A (2019) Supervised training of spiking neural network by adapting the E-MWO algorithm for pattern classification. Neural Process Lett 49(2):661–682
28. Karaboga D, Akay B, Ozturk C (2007) Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In: International conference on modeling decisions for artificial intelligence. Springer, Berlin, pp 318–329
29. Dang TL, Hoshino Y (2019) Hardware/software co-design for a neural network trained by particle swarm optimization algorithm. Neural Process Lett 49(2):481–505
30. Meissner M, Schmuker M, Schneider G (2006) Optimized Particle Swarm Optimization (OPSO) and its application to artificial neural network training. BMC Bioinformatics 7(1):125
31. Li F (2010) Hybrid neural network intrusion detection system using genetic algorithm. In: 2010 International conference on multimedia technology. IEEE, pp 1–4
32. Moradi M, Zulkernine M (2004) A neural network based system for intrusion detection and classification of attacks. In: Proceedings of the IEEE international conference on advances in intelligent systems-theory and applications, pp 15–18
33. Liu C, Niu P, Li G, You X, Ma Y, Zhang W (2017) A hybrid heat rate forecasting model using optimized LSSVM based on improved GSA. Neural Process Lett 45(1):299–318
34. Ghanem WA, Jantan A (2018) Hybridizing artificial bee colony with monarch butterfly optimization for numerical optimization problems. Neural Comput Appl 30(1):163–181
35. Ghanem WAH, Jantan A (2018) A novel hybrid artificial bee colony with monarch butterfly optimization for global optimization problems. In: Vasant P, Litvinchev I, Marmolejo-Saucedo J (eds) Modeling, simulation, and optimization. Springer, Cham, pp 27–38
36. Yu J, Xi L, Wang S (2007) An improved particle swarm optimization for evolving feedforward artificial neural networks. Neural Process Lett 26(3):217–231
37. Mizuta S, Sato T, Lao D, Ikeda M, Shimizu T (2001) Structure design of neural networks using genetic algorithms. Complex Syst 13(2):161–176
38. Lam HK, Ling SH, Leung FH, Tam PKS (2001) Tuning of the structure and parameters of neural network using an improved genetic algorithm. In: IECON'01. 27th Annual conference of the IEEE industrial electronics society (Cat. No. 37243), vol 1. IEEE, pp 25–30
39. Ghanem WAH, Jantan A (2014) Using hybrid artificial bee colony algorithm and particle swarm optimization for training feed-forward neural networks. J Theor Appl Inf Technol 67(3):664–674
40. Ghanem WAH, Jantan A (2014) Swarm intelligence and neural network for data classification. In: 2014 IEEE international conference on control system, computing and engineering (ICCSCE 2014). IEEE, pp 196–201
41. Mirjalili S, Hashim SZM, Sardroudi HM (2012) Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. Appl Math Comput 218(22):11125–11137

42. Ghanem WAH, Jantan A (2018) New approach to improve anomaly detection using a neural network optimized by hybrid ABC and PSO Algorithms. Pak J Stat 34(1):1–14
43. Mirjalili S, Mirjalili SM, Lewis A (2014) Let a biogeography-based optimizer train your multi-layer perceptron. Inf Sci 269:188–209
44. Mirjalili S (2015) How effective is the Grey Wolf optimizer in training multi-layer perceptrons. Appl Intell 43(1):150–161
45. Ghanem WA, Jantan A (2018) A cognitively inspired hybridization of artificial bee colony and dragonfly algorithms for training multi-layer perceptrons. Cognit Comput 10(6):1096–1134
46. Özgür A, Erdem H (2016) A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015. PeerJ Preprints 4:e19541
47. Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE symposium on computational intelligence for security and defense applications. IEEE, pp 1–6
48. Lee W, Stolfo SJ (2000) A framework for constructing features and models for intrusion detection systems. ACM Trans Inf Syst Secur (TiSSEC) 3(4):227–261
49. Siddiqui MK, Naahid S (2013) Analysis of KDD CUP 99 dataset using clustering based data mining. Int J Database Theory Appl 6(5):23–34
50. Zainal A, Maarof MA, Shamsuddin SM (2007) Feature selection using rough-DPSO in anomaly intrusion detection. In: International conference on computational science and its applications. Springer, Berlin, pp 512–524
51. Alomari O, Othman ZA (2012) Bee's algorithm for feature selection in network anomaly detection. J Appl Sci Res 8(3):1748–1756
52. Jebur HH, Maarof MA, Zainal A (2015) Identifying generic features of KDD Cup 1999 for intrusion detection. JurnalTeknologi 74(1):1–9
53. Othman ZA, Muda Z, Theng LM, Othman MR (2014) Record to record feature selection algorithm for network intrusion detection. Int J Adv Comput Technol 6(2):163
54. Yassin W, Udzir NI, Muda Z, Sulaiman MN (2013) Anomaly-based intrusion detection through k-means clustering and Naives Bayes classification. In: Proceedings of 4th international conference on computing informatics, ICOCI, vol 49, pp 298–303
55. Rufai KI, Muniyandi RC, Othman ZA (2014) Improving bee algorithm based feature selection in intrusion detection system using membrane computing. J Netw 9(3):523
56. Shiravi A, Shiravi H, Tavallaee M, Ghorbani AA (2012) Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Comput Secur 31(3):357–374
57. Moustafa N, Slay J (2015) UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military communications and information systems conference (MilCIS). IEEE, pp 1–6
58. Moustafa N, Slay J (2016) The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. Inf Secur J Glob Perspect 25(1–3):18–31
59. Moustafa N, Slay J (2015) The significant features of the UNSW-NB15 and the KDD99 data sets for network intrusion detection systems. In: 2015 4th international workshop on building analysis datasets and gathering experience returns for security (BADGERS). IEEE, pp 25–31
60. Sindhu SSS, Geetha S, Kannan A (2012) Decision tree based light weight intrusion detection using a wrapper approach. Expert Syst Appl 39(1):129–141
61. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization, vol 200. Technical report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department
62. Rojas I, Cabestany J, Catala A (2015) Advances in artificial neural networks and computational intelligence. Neural Process Lett 42(1):1–3
63. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98
64. Wang GG, Deb S, Coelho LDS (2015) Elephant herding optimization. In: 2015 3rd International symposium on computational and business intelligence (ISCBI). IEEE, pp 1–5
65. Beyer H-G (2013) The theory of evolution strategies. Springer, New York
66. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
67. Wang GG, Deb S, Cui Z (2015) Monarch butterfly optimization. Neural Comput Appl 31:1–20
68. Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl-Based Syst 96:120–133
69. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
70. Bamakan SMH, Wang H, Shi Y (2017) Ramp loss K-support vector classification-regression; a robust and sparse multi-class approach to the intrusion detection problem. Knowl-Based Syst 126:113–126

71. Khammassi C, Krichen S (2017) A GA-LR wrapper approach for feature selection in network intrusion detection. Comput Secur 70:255–277
72. Papamartzivanos D, Mármol FG, Kambourakis G (2018) Dendron: genetic trees driven rule induction for network intrusion detection systems. Future Gener Comput Syst 79:558–574
73. Kumar G, Kumar K (2015) A multi-objective genetic algorithm based approach for effective intrusion detection using neural networks. In: Yager R, Reformat M, Alajlan N (eds) Intelligent methods for cyber warfare. Springer, Cham, pp 173–200
74. Hamed T, Dara R, Kremer SC (2018) Network intrusion detection system based on recursive feature addition and bigram technique. Comput Secur 73:137–155
75. Yassin W, Udzir NI, Muda Z, Sulaiman MN (2013) Anomaly-based intrusion detection through k-means clustering and Naives Bayes classification. In: Proceedings of 4th international conference on computing informatics, ICOCI, vol 49, pp 298–303