Adaptive Particle Swarm Optimization Based Ensemble Classifier for Network Intrusion Detection

Tinofirei Museba (tmuseba@uj.ac.za)

University of Johannesburg CBE: University of Johannesburg College of Business and Economics

https://orcid.org/0000-0003-0356-5245

Research Article

Keywords: Intrusion detection, machine learning, ensemble method, support vector machines

Posted Date: March 25th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1460599/v1

Adaptive Particle Swarm Optimization Based Ensemble Classifier for Network Intrusion Detection

Tinofirei Museba, University of Johannesburg, College of Business and Economics, Department of Applied Information Systems, Auckland Park, Johannesburg

tmuseba@uj.ac.za

Abstract

Real time networks generate very large amounts of traffic and are always susceptible to attacks and intrusions making the security and privacy of a network system a major concern. Recently, attacks on computer networks has increased significantly. To mitigate the effects of network attacks, Intrusion Detection Systems (IDS) are used to enforce computer security by identifying and repealing malicious activities in real time computer networks. Network Intrusion approaches are getting more sophisticated and there is now a dire need for developing intrusion detection systems with optimal efficacy. To detect network attacks and intrusions, a number of contemporary intrusion detection systems have been proposed, among them, machine learning approach based ensemble learning techniques have been applied. The upsurge in the sophistication of attacks has created an intense need for the deployment of more dependable and robust network paradigms capable of detecting networking attacks more effectively and efficiently. Despite an avalanche of machine learning based ensembles, it remains a difficult task to formulate an optimal ensemble configuration capable of explicitly detecting network attacks. Existing machine learning approaches cannot achieve consistent generalization across multiple classes from large-scale imbalanced multiclass datasets that are associated with concept drift. To enhance the efficiency of network intrusion detection, this paper proposes a Particle Swarm optimized ensemble called Dynamic Heterogeneous Ensemble Particle Swarm Optimization (DHEPSO) algorithm where particles exhibit different search characteristics. Particles are at liberty to follow different search behaviors selected from a behavior pool, thereby efficiently addressing the exploration and exploitation tradeoff problem. Empirical studies conducted on the new version of NSLKDD99 that reflects modern day traffic trends showed that our proposed approach achieved comparative performance when compared with existing network intrusion detection techniques.

Keywords*: Intrusion detection, machine learning, ensemble method, support vector machines*

1. Introduction

Network intrusions pose a great risk to network security and network intrusion detection systems (IDS) have become common tools used to monitor malicious activities and to trigger alerts in order to detect suspicious attacks. Network intrusions make the system unpredictable for network traffic because of its non-linear behavior [36]. Without an intrusion detection system, security principles such as confidentiality, integrity and availability are often compromised as a result of intrusions and attacks such as spoofing, traffic analysis, cyber-attacks and other harmful vulnerabilities [3]. In practical complex scenarios, network intrusion detection is inherently challenging owing to background perturbance occasioned by concept drift. With intrusion detection, the objective is to analyze and monitor the events that take place in a computer system in order to detect if there are any signs of security problems. To formulate an effective and efficient intrusion detection system, two commonly used approaches are machine learning and deep learning. The two approaches have demonstrated the capacity to safeguard the computer system [37] from different types of attacks with intelligence. Learning from traffic patterns that exhibit non-linear behavior [2] and associated with concept drift is often a challenging task. As a result existing machine learning approaches can only detect some but not all traffic types making them unable to achieve consistent good performance across multiple classes. Network based computer systems have recently become the targets of attacks from intruders.

To properly monitor, detect and repel suspicious network activities, network intrusion systems are generally deployed [1] to provide a safe and secure network.

Real time intrusion detection systems are generally divided into two categories: signature based and anomaly based intrusion detection systems [4] The signature based approach, often referred to as a misuse-based detection performs a comparison of the signature of the recognized malicious activities and generates alerts whenever there is a match. This makes signature-based systems capable of diagnosing perceived attacks with a relatively low false alarm rate. Anomaly-based approaches observe the pattern of the systems to see if there is any deviation from the norm to trigger an alert, making them competent to encounter zero-day attacks. This approach undergoes a high false-positive rate [5].

Existing network intrusion detection systems work on the assumption of the anomaly based intrusion detection which provides a description of a class of techniques that attempt to distinguish network traffic as either normal or anomalous. In real-time network traffic, a large amount of incoming packet data need to be identified in order to resolve security issues, that is, identifying whether the traffic is normal or attack [6]. The categorization process is based on binary classification of traffic. Despite the existence of many methods designed to provide security to networks such as firewalls, cryptography and restriction of access to network such as authentication, attacks and intrusion to networks still persist. The major drawback of the existing methods is that all the proposed methods are not able to detect intrusions and attacks inside the network. To overcome the weaknesses of existing methods, there is need for intrusion detection systems capable of detecting internal and external attacks and intrusions. Such intrusion detection systems must be able to monitor inbound network traffic to detect any

abnormal behavior or abuse by users. To distinguish between a normal user and an attacker, a machine learning based intrusion detection system performs classification and anomaly detection. It categorizes all users into two groups of normal users and attackers. To expedite the detection of intrusion detection, a set of features is selected to identify patterns of different users. In most cases, the dataset of users' behaviors and interaction with other networks includes different features and using all features in time-consuming and computationally expensive.

Properly configured Intrusion Detection Systems (IDSs) have become the mainstay of identifying and repealing malevolent activities in real time networks. The success of a properly configured Intrusion Detection Systems is hugely attributed to their ability to maximize their detection accuracy and at the same time reducing false alarms. Recently, anomaly-based intrusion detection systems, classification problems and pattern recognition problems have benefitted from machine learning approaches of combining classifier outputs. Ensemble classifiers have demonstrated the ability to improve the accuracy of several regression, classification and pattern recognition tasks. Due to their modularity, ensemble classifiers have demonstrated empirically and theoretically to possess better accuracy than any single component classifier. The successful adaptation of an ensemble classifier is attributed to the diversity of the prediction outputs of the component classifiers as well as the combination method used to combine prediction outputs and optimize classifier accuracy. To address these challenges that are poorly misunderstood, several heuristics capable of selecting a suitable optimization method and improve individual classifier accuracy have been proposed. Particle Swarm Optimization (PSO), which is a stochastic optimization technique that models the analogy of a swarm of birds in flight [7] is a one of the several heuristics proposed to solve the combination problem in ensemble methods. To ameliorate the accuracy of intrusion detection, we propose the Dynamic Heterogeneous Ensemble Particle Swarm Optimization (DHEPSO) based on an ensemble of support vector machines that is optimized by Particle Swarm Optimizers to achieve the much need versatile performance by constantly enhancing diversity to improve exploration and exploitation by arresting stagnation and premature convergence.

The remainder of this paper is organized as follows. Section 2 provides a review of some previous work performed on the concept of network intrusion and feature selection. Section 3 provides a description of the algorithms used in our proposed solution such as Support Vector Machines and Particle Swarm Optimizers. Section 3.4 provides a description of our proposed methodology. Section 4 provides an explanation of our experimental setup and the classification results are presented. Section 5 provides a conclusion of the work carried out.

2. Related Work

A number of contemporary approaches based on machine learning have been proposed to handle network intrusion as scenarios associated with network intrusion have become prevalent. Although the last decade has witnessed a surge in the application of machine learning to detect network intrusion, network intrusion problems still exist as performances across multiple domains still cannot be achieved.

A plethora of approaches to enhance network intrusion detection such as

combining multiple classifiers to form an ensemble have been proposed. Prominent amongst these is the work of Adbulla Amin Aburomman and Mamun Bin Ibne Reaz [8]. The approach generates weights using Particle Swarm Optimizers and also to perform combination of k-nearest neighbor and classifiers from support vector machines to generate an ensemble.

Ying Zhou [9] presented an ensemble of classifiers using the modified adaptive boosting that uses the area under the curve algorithm in an effort to handle network intrusions. Generated classifiers were combined into an ensemble using particle swarm optimization. The performance of the algorithms was measured using two metrics namely model complexity and computational cost. Hariharan Rajadurai [10] proposed an ensemble classifier based on the concept of stacking to detect intrusions in wireless networks. The approach detects attacks by combining different machine learning algorithms. Accuracy is achieved at the cost of resource inefficiency. Adeel Abbas [11] proposed an ensemble based intrusion detection model that combines different learning algorithms. Although the algorithm performed well in terms of accuracy but the model had a higher complexity. To detect network intrusions efficiently, Mehedi Hassan [12] proposed a hybrid deep learning model based on the concept of convolutional neural networks. The algorithm demonstrated good performance on the smaller data set used to perform the experiment but reflected new attacks slowly.

Ishfaq Manzoor [13] presented an intrusion derection system that works on reduced features. The algorithm performs features ranking correlation to discard redundant features. The algorithm has no guarantees of retaining relevant features. Artificial Neural Networks is used as the classifier making the algorithm susceptible to convergence to local optima. To select the best subset of features and detect network intrusion efficiently, Chaouki Khammassi [14] proposed a wrapper method that uses genetic algorithm to perform the search process and logistic regression performs the learning process to detect intrusions. The application of a wrapper is time consuming and informative features are likely to be filtered. Huiwen Wang [15] suggested a network intrusion framework that uses a feature augmentation based support vector machine as the learning algorithm. The approach does not consider the prevalence of different types of attacks.

Yuyang Zhou [16] proposed a network intrusion framework that performs feature selection and performs intrusion detection using an ensemble of classifiers. The approach uses a heuristic algorithm to perform dimensionality reduction. Optimal set selection is performed via dimensionality reduction depending on the correlation that exists amongst the different features. Probability distributions are combined using majority voting. Feature selection based on correlation can filter informative features.

Hadeel Alazzam [17] presented a network intrusion detection algorithm based on the wrapper concept. To perform features selection, the algorithm utilizes the pigeon inspired optimizer. The algorithm is time consuming and filters informative features.

Wenhong Wei [18] proposed a network intrusion detection approach that performs feature selection using a multi-objective immune algorithm. To further enhance the efficiency of the algorithm, an elite selection strategy is used. The algorithm optimizes convergence and improves classification accuracy at the expense of computational complexity.

In an effort to detect abnormal behavior, networks often generate a large number of false alarms. The effectiveness of detecting network intrusions is often hampered by the presence of noise. Domain Name System (DNS) data is often corrupted by agents such as software bugs and local packets not detected as attacks can significantly increase the false alarm rate. The feature selection algorithms suggested need to take into consideration the change in the underlying distribution of the traffic data since spammers and hackers always work on tricking intrusion detection systems.

To address the current challenges faced by the intrusion detection systems, this paper proposes a Support Vector Machine (SVM) based ensemble classifier whose parameters are optimized by particle swarm optimizers to detect network intrusion. To address the problem of premature convergence and to maintain diversity throughput the optimization process, a pool of different PSO search behaviors is constructed and particles position and velocity is updated by randomly selecting from the behavior pool. To perform feature selection, the Maximal Independent Classification Information and Minimal Redundancy (MICIMR) is used.

3.1 Ensemble of Support Vector Machines

In machine learning, classifiers are often combined to constitute an ensemble classifier. In this study, Support Vector Machines is used as the base learning algorithm. Support Vector Machines (SVMs) [19] have recently emerged as an effective and efficient learning algorithm to solve pattern recognition problems that involve classification and regression problems. For linearly separable examples, Support Vector Machines maps the examples from an input space to a new high dimensional feature space. To overcome the overfitting problem and to reduce the generalization error, Support Vector Machines make use of the Structural Risk Minimization (SRM) principle. Support Vector Machines solve a quadratic optimization problem to determine the number of support vectors. The support vectors generated are then used to define the decision boundary for the classification problem. For many real life datasets, the dataset may not be linearly separable and the hyperplane to separate the two classes may not exist. For many real world applications, datasets may not be linearly separable. To make them separable, the kernel trick [20] is used to map the data into a different dimensional space that is much higher. The classifiers generated by the learning algorithm are combined into an ensemble. The success of ensemble classifiers is attributed to their modularity which allows them to achieve high prediction accuracy than single classifiers and facilitates their parallelization. In addition, factors such as diversity and accuracy of individual classifiers also contribute to the success of the ensemble for classification problems. Approaches such as majority voting and weighted voting are often used to combine the component classifiers.

3.2 Particle Swarm Optimizers

Particle Swarm Optimizer is a heuristic population based iterative, global and stochastic optimization algorithm inspired by the social behavior of bird flocking or fish schooling to conduct an intelligent search for the optimal solution [21].

Initially, the algorithm works with a set of defined agents called particles which are all set in random positions in the problem space. The particle's location calls for the definition of a fitness location. The objective of solving the optimization problem is to search for the best possible position. The best possible position found is the one capable of minimizing the fitness function. At each iteration, the fitness of each particle is evaluated by the algorithm and the velocity is updated and the new particle position is computed. PSO is a metaheuristic algorithm for solving unconstrained optimization problems [22].

The performance of an optimization algorithm is determined by its ability to explore or exploit. Exploration means the ability of a search algorithm to explore different areas of the search space in order to have high probability of finding good optimum. Exploitation on the other hand means the ability to concentrate the search around a promising region in order to refine a candidate solution [27]. Particle Swarm Optimizers offer more benefits when compared to other optimization algorithms. Particle Swarm Optimizers involves few or no assumptions regarding the problem being optimized. It is derivative free as it does not require the optimization problem to be differentiable, therefore not requiring gradients making it applicable to different sets of real life problems such as those with discontinuous or non-convex and multimodal problems. Particle Swarm Optimizers manipulates a set of potential solutions or particles to search for the best solution to minimize the objective function. Each particle i is associated with three vectors namely:, its position vector x_i, indicating a candidate solution, its velocity vector v_i , which shows the particle's search direction and its personal best vector p_i which shows the particle's best position so far in the search space. At each iteration, the best position of the ith particle is represented by pbest and the best position of all particles is represented by gbest. In the ith iteration, the velocity and position of a particular is updated as follows:

$$v_i^{t+1}(\mathrm{j}) = \omega * v_i^t + c_1 + r_1 * (pbest^t(\mathrm{j}) - x_{i,j}^t) + c_2 * r_2 * (gbest^t(\mathrm{j}) - x_{i,j}^t), \mathrm{j} = 1,2,\ldots \mathrm{D} \quad (1)$$

When the particle's velocity is calculated, then the position of the particle is updated using:

$$x_i^{t+1} = v_i^{t+1} + x_i^t \quad (2)$$

The social learning factor is represented by c_1 and the cognition factor is represented by c_2. The magnitude of the random force in the direction of particles with the previous best-visited position is determined by the coefficients c_1 and c_2. Coefficients r_1 and r_2 are two independent uniform random numbers between [0,1] and ω is the inertia weight used to control the balance between exploitation and exploration. The vectors r_1 and r_2 adds stochastic properties to the algorithm. A large value of ω enables the optimization algorithm to get out of the local optimal solution and to enhance the global search ability whereas a smaller value of ω facilitates algorithm convergence and improves the local search ability [23]. The inertia weight is used to adjust the entire search process of the PSO algorithm. Inertia weights cannot be used to dynamically change the capability to explore or exploit during the search process. For the PSO algorithm, diversity is often used to evaluate the algorithm's position of exploration and exploitation to help adjust its capacity to exploit and explore [27].

In PSO, particles share its swarm's best experience (the global best) that can lead the particles to cluster around the global best. If the global best is located near a local

minimum, escaping from local optimum becomes difficult and PSO suffers diversity loss near the local minimum [28]. This makes PSO to suffer from the lack of balance between the exploration and exploitation. The internal characteristics of a search process can be revealed by diversity. Numerous variants of the standard PSO and its modifications have been developed to handle dynamic changes and all follow the same behaviors. The particles in the swarm implement the same velocity and position update rule thereby exhibiting the same search characteristics. Intuitively speaking, the success of an optimization algorithm is attributed to its ability to balance exploration and exploitation. To achieve the balance, the use of heterogeneous swarms where particles are allowed to assume different velocity and position update is proposed. To allow particles to use different update rules, a swarm may consist of explorative particles as well as exploitative particles, enabling the optimization algorithm to explore and exploit throughout the search process thereby avoiding premature convergence. PSO algorithms endowed with diversity converge quickly to optimal solutions without premature convergence [29]. In this paper, a heterogeneous ensemble PSO (HEPSO) is proposed. HEPSO allows particles in a swarm to assume different search behaviors by randomly selecting velocity and position update rules from a behavior pool. The dynamic HEPSO implies that particle behaviors can randomly change during the search process. If a particle fails to improve its personal best position over a window of recent iterations, that particle randomly selects new behaviors from the behavior pool. If there is no indication of change of the personal best of the particle, it may be an indication of early stagnation, which can be changed by assigning a new search behavior to the particle.

3.2 Feature Selection for Network Intrusion Detection

For intrusion detection tasks, traffic data is associated with the curse of dimensionality. As the volume of features increases, the complexity of any intrusion detection system to perform classification increases.

Intrusion detection is made complex as the dataset is associated with various irrelevant, redundant and extravagant features which increase the computation complexity and consumption of memory resources for the analysis of intrusions [24]. To reduce computational complexity, we select an optimal subset of features. Two popular approaches used in the literature to perform feature selection are the filter model and the wrapper model. The wrapper model exploits the prediction accuracy of an expert as a way of evaluating the "goodness" attribute relating to a set of features and filter employs a number of measurement metrics. For large datasets and streaming data, such approaches are inappropriate. Firstly, when you supply arbitrary features to a classifier, it can lead to biased results, making it difficult to depend on the prediction accuracy of the classifier as a metric to perform feature selection. In addition, trying different combinations to supply features to a classifier with a set of N features which scales to $2N$ combinations is not a feasible approach.

To perform feature selection, this study employs a feature selection method called Maximal Independent Classification Information and Minimal Redundancy (MICIMR) [25] based on Symmetrical Uncertainty. The algorithm is used to perform feature selection due to its easy interpretability, low-over fitting and

exemplary predictive performance. The Symmetrical Uncertainty feature selection technique employed by the Maximal Independent Classification Information and Minimal Redundancy (MICIMR) measures the uncertainty of a random variable x to another variable y as illustrated by the following equations:

$$H(x) = \sum P_{(x_i)} \, log_2(P(x_i)) \quad [1]$$

$$H(x|y) = \sum P_{(y_i)} \sum_j P_{(x_i)}|y_i) \, log_2(P(x_i)|y_i) \quad [2]$$

$P(x_i)$ represents the prior probability for all values of x and $P(x_i|y_i)$ is the posterior probability of x given y. Symmetric Uncertainty serves as a correlation metric for the features and the class and is computed as:

$$SU = \frac{H(x)+H(y)-H(x|y)}{H(x)+H(y)} \quad [3]$$

$H(x)$ and $H(y)$ represent the entropies related to the probability computed from each feature and the associated class values and $H(x,y)$ represents the joint probability of combinations of values x and y [26]. Symmetrical Uncertainty is not biased by the number of values of an attribute.

The MICIMR algorithm based on Symmetrical Uncertainty has the ability to screen out highly relevant features. Based on the symmetric uncertainty, MICIMR computes the relevance and redundancy terms of class independent characteristics respectively.

The independent classification criterion is used to compute the relevance and redundancy terms. The MICIMR algorithm then combines the two characteristics for the solution criteria. The MICIMR performs feature selection in a two stage format. The first phase performs initialization of the set S and specify the number of features selected δ. Mutual information is calculated for each feature and selects the largest feature. The second phase calculates the values of the variables created. The pseudocode of the MICIMR algorithm is as follows:

Algorithm 1: MICIMR algorithm Pseudocode

Input: Dataset D with F= $\{f_1, f_2, .., f_n\}$ and the class C, Set the threshold δ.

Output: the set S with the selected optimal features

1. Initialize parameters: S= θ, k=0
2. Specify choice of first feature
3. Define relevance, irrelevance and redundancy from set features
4. Find $f_i \in$ F that arg max RC (f_i;C)
5. $J_{MICIMR}(f_i)$ = arg max RC(f_I;C)
6. Set F ← F\{f_i}
7. Set S← {f_I}
8. While k $\leq \delta$ (next feature selection)
9. Find the optimal feature f_i
10. Calculate the value of I(f_i; C\f_{select})
11. Calculate the value of I(f_{select};C\f_i)
12. Calculate the value of RR(f_i; f_{select})
13. Update the value of J_{MICIMR} using $G_{mean-maco}$ formula
14. Find the candidate feature f_i with the largest J_{MICIMR}
15. Set F ← F\{f_i}
16. Set S ← {f_i}
17. K= k + 1
18. End While

3.3 Heterogeneous Ensemble Particle Swarm Optimizer

To ascertain the behavior of a particle swarm, three factors are taken into consideration and these are: population topology, the model of influence and the update rule. The PSO algorithm has demonstrated the ability to work well for most static optimization problems. However, in real life, many applications are dynamic in nature as optimal solutions obtained can become suboptimal as the environment changes.

To remain relevant, the PSO inspired algorithm then has to adapt the evolving optimal solution.

Recent studies have discovered that the loss of diversity in every learning setup can be detrimental to the generalization performance of a PSO inspired algorithm in time-varying environments [30]. Real world applications such as network intrusion are dynamic in nature, that is, they are associated with concept drift [31], where the underlying data distribution changes or evolves with time. Due to conceptual simplicity, models composed of populations of homogeneous individuals are attractive, however heterogeneity is ubiquitous in nature [32]. The heterogeneous particle swarm is potentially well suited for time varying and evolving environments that exhibit concept drift as particles have the liberty to assume new behavior in response to stagnation and premature convergence. If the environment evolves, during the process of trying to adapt to a changing optimal solution, the memory may get outdated. In addition, the PSO algorithm may suffer from diversity loss as a result of swarm convergence. With a heterogeneous dynamic PSO algorithm, particles follow different behaviors and assume different pairs of velocity and position update conventions to avoid diversity loss in an effort to optimize the exploitation and exploration process.

The Dynamic Heterogeneous Ensemble Particle Optimization (DHEPSO) allows particles to assume different behaviors to evade stagnation, diversity loss and premature convergence. The DHEPSO algorithm is proposed for this work as it is potentially well suited for detecting network intrusions in time varying and evolving environments.

The mechanism enables the swarm to manage its diversity dynamically in response to changes in the environment.

For most optimization tasks, balancing exploration and exploitation is considered to be an important aspect for many optimization algorithms. When the search process starts, the algorithm focuses on exploration while preferring exploitation as the search process converges on an optimum [30]. Determining the point of change of switch of the algorithm from an explorative behavior to exploration behavior is not an easy task.

To solve the non-linearity behavior and concept drift, heterogeneous swarms that allow particles to assume different velocity and position update rules are implemented. When particles are allowed to implement different velocity and position update rules, the composition of the swarm will be made up of explorative particles as well as exploitative particles. At each iteration, behaviors change by randomly selecting new behaviors from the behavior pool. This equips the optimization algorithm with the ability to explore and exploit throughout the search process. This paper proposes the Dynamic Heterogeneous Ensemble Particle Optimization algorithm (DHEPSO) where particles in the swarm assume different search behaviors by randomly selecting velocity and position update rules from a behavior pool to detect network intrusion.

3.4 Dynamic Heterogeneous Ensemble Particle Optimization (DHEPSO)

To address universality and avoid premature convergence to a local optimum and generate robust performance of PSO, this paper introduces the Dynamic Heterogeneous Ensemble Particle Swarm Optimizer (DHEPSO). DHEPSO constructs a pool of different PSO search behaviors and the particles are allocated their positions and velocity by randomly selecting the update rules from the behavior pool in order to improve the quality of the solution and scalability. Recent studies have demonstrated that the diversity of the population plays a pivotal role in avoiding the premature convergence. The success of ensemble classifiers for classification and regression tasks is hugely attributed to the performance and diversity of individual models. In the proposed DHEPSO, the population is split into two clusters or subpopulations to enhance the diversity throughout the optimization process. Clustering is performed for the adaptive separation of the population into two clusters. To perform the clustering process, the fast search clustering method [35] is adopted for the adaptive clustering of the population into two clusters.

The Dynamic Heterogeneous Ensemble Particle Optimization algorithm allows particles to assume different behaviors in the process of searching for the optimal solution. Instantiation of update rules and their parameters can be performed at individual level as they are considered to be part of the swarm thereby introducing heterogeneity. For most real life applications, particles change their configurations overtime resulting in dynamic heterogeneity. Adaptive particle swarms at the individual level build on dynamic heterogeneity by triggering configuration changes as a response to some event caused by the behavior of the swarm, thus “guiding” the otherwise random dynamism [33]. The DHEPSO proposed in this paper selects random behaviors for particles from a pool of behaviors. Each behavior is composed of a pair that consists of a position update and a velocity update. For the Dynamic Heterogeneous Ensemble Particle Swarm Optimization (DHEPSO), the behavior pool is populated with five behavior models

that include the standard PSO behavior, social PSO behavior, Cognitive PSO behavior, Bare bones PSO behavior and the Modified Bare bones PSO behavior. With the standard PSO behavior, the particles use equation (1) and (2) to update its velocity and position. Throughout the search process, the cognitive acceleration constant c_1 is assigned a value of 2.5 and linearly decreasing to 0.5. The social acceleration constant c_2 initial value is set to 0.5 and is linearly decreased to 2.5 throughout the search process. This behavior setup facilitates exploration near the beginning of the search process and promotes exploitation near the end of the search process. The social PSO behavior discards the cognitive component of equation (1), allowing the particle to be attracted to the global best position attained so far. A value of 2.5 is assigned to the social acceleration constant c_2. The setup of this behavior promotes exploitation as all particles that align to this behavior at time step t effectively becomes a single stochastic hill-climber. Cognitive PSO behavior discards the social component of equation (1), making the particle to be only attracted to its own personal best position. Position updates are performed by equation two. A constant value of 2.5 is assigned to the cognitive acceleration constant c_1. The setup of this behavior facilitates exploration as all particles that assume this behavior become independent hill-climbers. The Bare bones PSO behavior replaces the velocity update equation with a Gaussian distribution. The velocity becomes the particle's new position. The behavior set up leads to the exploitation of the point in the middle of a particle's personal best position and the global best position. When the personal best positions converge towards the global best, the focus changes towards exploitation. The Modified Bare bones PSO behavior exhibits better initial exploration and later exploitation than the first version Bare bones PSO as particles focus on their personal best positions most of the time. The Dynamic Heterogeneous Ensemble Particle Optimizer allows particle behaviors to randomly change during the search process. A particle randomly selects new behaviors from the behavior pool when the particle fails to improve its personal best position over a window of recent iterations. If there are no changes to the personal best position, it may be an indication of early stagnation or premature convergence and has to be addressed by assigning a new search behavior to the particle. When DHEPSO is combined with re-initialization and the re-evaluation processes, it facilitates effective re-diversification and re-convergence of the swarm if changes in the concept is detected during the search process.

3.5 System Methodology

The objective of the paper is to develop a particle swarm optimized ensemble classifier based on Support Vector Machines (SVM) to improve the accuracy and detection rates of network intrusion detection. To find high quality parameters, a Dynamic Heterogeneous Ensemble Particle Swarm Optimizer (DHEPSO) is implemented. In DHEPSO, a pool of different PSO search behavior is constructed and the particles are assigned position and update values by randomly selecting from the behavior pool to avoid premature convergence to a local optimum. For most applications, the faster convergence of PSO algorithm can easily cause parameter search to fall into local optimality resulting in premature onvergence. To further avoid premature convergence the diversity of particles is further improved by performing clustering on the particles to adaptively split the

particle swarm into two subpopulations and use the behavior pool to update the position and velocity of particles. Update rule heterogeneity makes it possible to have groups of specialized particles that perform different but complementary tasks. As one group can explore the search space while another performs the local search. The strategy further enhances the diversity of particles and helps the particles to jump out of local optimum.

Algorithm 2: SVM-PSO Optimization

Require: NSL-KDD dataset $\{(x_i, y_i)\}$

1. Perform cross-validation to set initial values of C and δ
2. $C_{G_k} \leftarrow$ output vector with cost C values generated by grid search with cross validation
   $\delta_{G_k} \leftarrow$ output vector with kernel parameters generated by grid search with cross validation
   $acc_{Gbest}, C_{Gbest}, \delta_{Gbest} \leftarrow$ Train SVM(Train_data, C_{G_k}, δ_{G_k})
3. Input variables
   $C_{min} \leftarrow C_{Gbest}$; $\delta_{min} \leftarrow \delta_{Gbest}$ Neighborhood of C_{G_k} and δ_{Gbest} where PSO search is performed
   $C_{max} \leftarrow C_{Gbest}$, $\delta_{max} \leftarrow \delta_{Gbest}$ plus radius of neighborhood C_{G_k} and δ_{G_k}
4. Initialize particles with population size with the number of particles in the swarm
   $C \sim \cup(C_{min}, C_{max})$; $C \sim \cup(\delta_{min}, \delta_{max})$
   $v \leftarrow 0$
   For k = 1 to number of particles in the swarm do
   acc[k] ← evaluate SVM (C[k], training data)
   endfor
   $acc_{best} \leftarrow acc$
   $C_{best} \leftarrow C$; $\delta_{best} \leftarrow \delta$
   $acc_{Gbest} \leftarrow \max(acc_{Gbest}, \max(acc_{best}))$
   $C_{Gbest} \leftarrow \max(C_{Gbest}, \max(C_{best}))$
5. Execute PSO algorithm

For i = 1 to number of iterations do

$r_1 \sim (0,1)$; $r_2 \sim v(0,1)$

$v \leftarrow w \times v + c_1 \times r_1 \times (C_{best} - C) + c_2 \times r_2 \times (C_{Gbest} - C)$

$C \leftarrow C + v$; $\delta \leftarrow \delta + v$

for k = 1 to number of particles in the swarm do

If [k] > 0 then

acc[k] ← Evaluate SVM (C[k], training_data)

acc_{best} {k} ← max (acc_{best} [k], acc[k]

C_{best}[k] ←max(C_{best}[k], C[k])

$endif$

Endfor

$acc_{Gbest} \leftarrow \max(acc_{Gbest}, \max(acc_{best}))$

$C_{Gbest} \leftarrow \max((C_{Gbest}, \max(C_{best}))$

$Endor$

Output G_{best}

For the classification accuracy of the SVM model and detection rates, the kernel parameter δ and error penalty c play a significant role. To improve the classification accuracy and detection rates, the parameters are determined by the PSO algorithm. With the PSO algorithm, the optimization process can be regarded as searching for optimal parameters in a feature space through iterative calculations. A fitness function determines the fitness values for each particle and a velocity vector determines its flying direction and distance. At each iteration, the particles update themselves through two best positions: one is found by the single particle called Pbest and the other is found in the whole particle group named gbest.

The objective is to perform optimization using PSO algorithm to select a regularization penalty cost parameter of the error term C and the kernel parameter δ for SVM training to increase the accuracy of the DHEPSO approach. The starting values are generated using cross-validated grid search over a predefined grid of possible values. The description of the algorithm is as follows:

Algorithm 3: DHEPSO algorithm

Input: NSL-KDD Dataset, classifier predictions, swarm sizes, number of iterations t, inertia weight, acceleration coefficients, behavior pool p, error tolerance, stagnation threshold

For particle m = I to M do

Initialize position u_m^0 and velocity v_m^0

For iteration t = 1 to T do

For particle m = 1 to M do

Perform feature selection using MICIMR

Populate behavior pool p(x,y)

Update velocity v_m^t and position u_m^t

Combine SVM classifiers using majority voting

Calculate fitness value f

If(u_m^t) > f(p_m^t) then update particle's position

If f(p_m^t) > f(g^t) then update group's best position using p_m^t to g^{t+1}

If error tolerance is not met and threshold higher then Update velocity with pool behavior p

Update position with pool behavior p

Endif

Endfor

Output the best position and final prediction of DHEPSO ensemble classifier

Perform classification with DHEPSO with varying values of inertia and acceleration coefficients

Evaluate performance using performance metric

End

3.6 Dataset Used

To conduct empirical experiments, we use the NSL-KDD variant of the Knowledge Discovery and Data Mining 1999 (KDD99) dataset is used [34]. The dataset consists of 41 attributes and the dataset is partitioned into 5 classes composed of normal and 42nd attribute which represents the class attribute and contains information about the attack groups and the attribute has positive or negative instances. The dataset consists of 23 different types of simulated attacks and the 23 types of attacks are further reduced into four categories. A brief description of the four categories of simulated attacks is provided.

- **Denial of Service Attack (DoS),** in this simulated attack, some computing or other storage facilities are made to be too full to respond to legitimate requests or in most cases denies legitimate users access to a machine.
- **User to Root Attack (U2R)** – the attacker starts with a normal user and then attempts to exploit vulnerabilities to access the root of the system.
- **Remote to Local Attack (R2L)**, In this category of attack, an attacker attempts to get access to a machine where he does not have an existing account by sending packets over a network and exploits the machine's vulnerability to illegally gain access to a local user.
- **Probing Attack,** to find known vulnerabilities in this class of attack, the attacker scans the network. This information can be used by the attacker with known services and machines on a network.

The following table shows the attacks assigned to different categories

Table 1: Categories and different types of attacks

Category	Attack Type
Normal	Normal records
Denial of Service	Back, land, Neptune, pod, smurf, teardrop, SYN flooding
PROBE Attack	Ping sweep, nmap, portsweep, satan,
User to Root Attack	Buffer_overflow, loadmodule, perl, rootkit, SQL attacks, spy
Remote to Local Attack	ftp_write, guest_passwd, imap, multihop, phf, spy, warezclient, warezmaster

The attributes of the NSL-KDD dataset are further split into four labels [38].

- Basic: The associated attributes of this label are composed of separate transmission control protocol (TCP) connections. The label consists og 9 attributes.
- Domain Knowledge: These attributes are found wrapped within ithe connections. The label consists of 13 attributes.
- Traffic: The associated attributes are numbered with the aid of windows 2-s of the time taken. The label consists of 9 attributes.
- Host: The associated attributes are grouped together for the evaluation of the types of attacks that will

survive for greater than 2-s. The label consists of 10 attributes.

The NSL-KDD dataset is split into four types of attacks for the purposes of training and testing. The dataset for training consists of 22 attack types and the testing set is composed of 15 attack varieties.

Table 2: Basic label attribute description of the NSL-KDD dataset

Name	Description	Type
Association Duration	Number of the association in seconds	Continuous
The Protocol Type in use	What is the type of protocol in use	Discrete
Network Service	Network Services occupied by the destinations	Discrete
Display Flag	Name the connection's status error or normal	Discrete
Source byte	Output bytes from origin to destination	Continuous
Transfer Destination bytes	Account for bytes from destination to origin	Continuous
Port/host Land	Same port/host to/from assign 1 and assign 0 otherwise.	Discrete
Snippets fragment status	Number of associations for wrong snippets	Continuous
Status: Urgent/Not Urgent	Number of urgent packets	Discrete

4.1 Experimental Results and Analysis

We implemented DHEPSO using Massive Online Analysis (MOA) [39] since network traffic data is streaming and associated with concept drift. The DHEPSO algorithm and the comparative experiments conducted are implemented in Java programming language that is used to extend the MOA software. The empirical experiments conducted are evaluated using metrics such as prediction accuracy, Kappa, Model cost, time and memory consumption. To ascertain the significance of the results obtained, we evaluate the statistical differences and non-parametric tests using the Demsar methodology [40] of all the algorithms compared to see how significant the differences are.

To determine the statistical significant differences between algorithms, non-parametric tests were conducted using Demsar's methodology [40]. To perform statistical test, the Friedman test is applied with $\alpha = 0.05$ with no statistical difference between the algorithms being the null hypothesis. If the null hypothesis is rejected, then the Nemenyi post hoc test is then used to identify which pairs of algorithms differ from each other.

4.2 Performance Evaluation

The generalization performance of a classifier is measured using statistical measures such as Accuracy, precision, recall and F_1 score. The performance measurement is reached by calculating the False Positive Rate (FPR) and True Positive Rate (TPR) and also referred to as False Alarm Rate (FAR) or failure rate and traffic is classified as malicious. The formulation can be expressed as the number of incorrect detection of normal records as intrusions over the total number of normal records.

The Detection Rate (DR) can be expressed as the number of correctly identified positive instances divided by the total number of instances identified as positive. The Detection Rate (DR) is also known as True Positive Rate (TPR) or recall or sensitivity. Commonly used performance measures are as follows:

True Positive Rate of intrusions (TPR) = $\frac{True\ Positive}{TruePositive+FalseNegative}$ x 100

True Negative Rate of intrusions (TNR) = $\frac{TrueNegative}{TrueNegative+FalsePositive}$ x 100

False Negative Rate of intrusions (FNR)=100– TruePositiveFalsex100

False Positive Rate (FPR) = 100-TrueNegativeRatex100

Overall Precision = $\frac{TruePositive}{TruePositive+FalsePositive}$ x 100

Overall Prediction accuracy = $\frac{TruePositive+TrueNegative}{TP+TN+FP+FN}$ x 100

F_1Score = $\frac{2TruePositive}{2TruePositive+FalsePositive+FalseNegative}\ x\ 100$

The Accuracy metric is expressed as the proportion of the correctly classified instances to the total number of instances and is calculated as:

Prediction Accuracy = $\frac{TruePositive+TrueNegative}{TruePositive+TrueNegative+FalsePositive+FalseNegative}$

Accuracy alone as a performance metric does not reflect which class has been classified better as most datasets are imbalanced and exhibit skewed distributions. The Kappa evaluation measure is often used in dynamic environments associated with concept drift.

The Kappa metric has the capacity to deal with multiclass and datasets with skewed distributions. A larger value of the Kappa is an indication of how generalized the classifier is whereas a Kappa value in the negative is a reflection of poor prediction accuracy. The Area Under the Receiver Operating Characteristics (AUC) is another metric implemented and is used to measure the quality of the model predictions.

4.3 Analysis and Results

The empirical experiments described in section 4.1 are performed and the results on the testing dataset of the NSL-KDD are compared with other state of the art algorithms designed to detect network intrusion on the same dataset. Table 3 provides average results for the DHEPSO algorithm in comparison with M-AdaBoost-A-PSO, M-AdaBoost-A-SMV [9] and WITMPR-SEMI-P [41]. The metric measurements based on multiple runs of M-AdaBoost –A-PSO, M-AdaBoost-A-SMV, WITMPR-SEMI-P and DHEPSO are summarized in Table 3 and Table 4.

Table 3: Average overall performance on NSL-KDD dataset

	Accuracy	**Precision**	**Recall**	$F_1 Score$	**Kappa**	**AUC**	**TPR**	**TNR**	**FPR**
DHEPSO	91.43	93.46	94.26	96.74	83.45	86.47	94.18	96.53	0.65
Ada-PSO	86.37	91.28	92.47	93.49	81.32	85.63	91.34	93.42	2.47
Ada-SMV	82.38	87.19	89.17	91.32	79.42	83.48	87.36	89.49	3.34
WITMPR	83.27	84.39	91.53	92.68	82.13	81.67	83.48	91.35	3.21

Figure 1 shows a comparative study of the accuracy of DHEPSO and other state of the art algorithms designed to detect network intrusion on selected features.

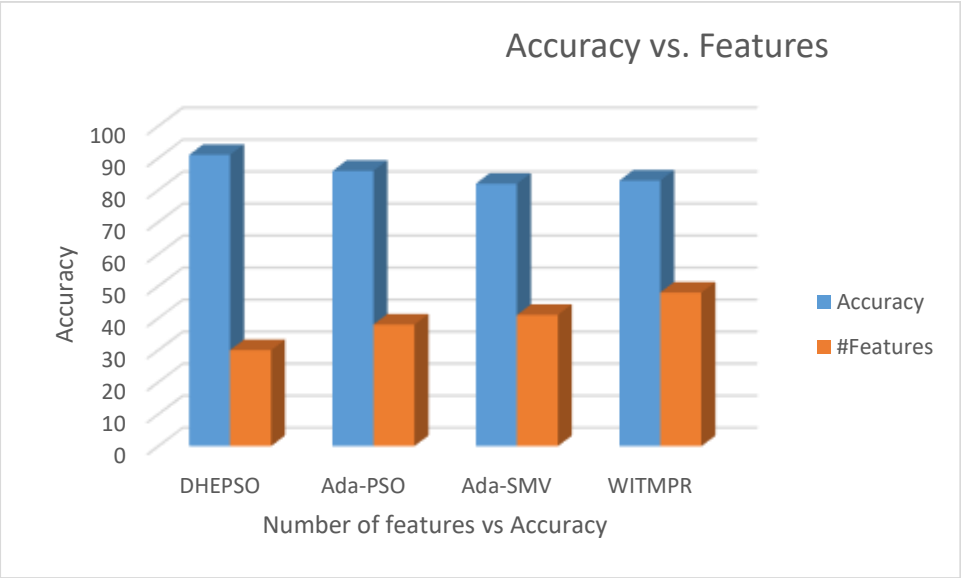

Figure 1: Number of features vs. Accuracy

Among all the algorithms compared, DHEPSO achieved a comparable accuracy and high TPRs

Table 4 shows the per class performance metrics for each class. The performance is based on the True Positive Rate (TPR) and False Positive Rate (FPR) per each class. Precision, Recall and AUC have the lowest FPRs for classes DoS, Normal and Probe. For the NSL-KDD dataset, our proposed approach, DHEPSO performed the best in recognizing minority classes, maximizing TPRs and minimizing FPRs across multiple classes.

To justify the significance of the results obtained from the experiments conducted on the NSL-KDD dataset, our proposed approach is further compared with the state of the art approaches designed to detect network intrusion reported in the literature. Table 3 captures study results generated by various researchers who have used the NSL-KDD dataset for detecting network intrusion and are ranked based on their accuracy and effectiveness measured based on the precision, Recall, Kappa and AUC. With a significantly higher precision, Recall and AUC, our proposed approach, DHEPSO, performed comparatively well to other state of the art approaches in previous studies. For minority classes with both high TPRs and low FPRs, DHEPSO performed remarkably well.

Table 4: Average per class performance on NSL-KDD dataset

		DoS	**Normal**	**Probe**	**R2L**	**U2R**
DHEPSO	TPR	0.9897863	0.9968437	0.9896843	0.9996792	0.8764982
	FPR	0.0001987	0.0002164	0.0002892	0.0002193	0.0003184
Ada-PSO	TPR	0.9997683	0.9948362	0.9974893	0.9879437	0.7968236
	FPR	0.0001354	0.0005262	0.0003172	0.0002146	0.0003648
Ada-SMV	TPR	0.9976418	0.9798374	0.9893784	0.9867847	0.7863648

	FPR	0.0002437	0.0002763	0.0001482	0.0003145	0.0003126
WITMPR	TPR	0.9965143	0.994832	0.9976843	0.9786783	0.8694748
	FPR	0.0001489	0.000134	0.0002165	0.0002148	0.0007984

The empirical experiments conducted on the NSL-KDD dataset also measured training times, model complexity and computational cost. Table 5 lists the average training time, Kappa values and memory consumption over 10 data splits performed on a computer with a core i7 processor and 32 GB of RAM.

Table 5: Metrics used to measure effectiveness

	DHEPSO	Ada-PSO	Ada-SMV	WITMPR
Training time	93.08	85.13	79.23	73.43
Kappa Statistic	83.45	69.26	69.48	65.43
Model Cost	0.46	1.26	1.29	1.34
Time	18.46	32.43	38.49	36.67
Memory	0.08	0.32	0.49	0.53

Table 6 provides a comparative analysis of the detection rates of the algorithms on selected number of features.

Table 6: Number of features vs. detection rates

Algorithm	Accuracy	Detection Rate	#Features
DHEPSO	94	98	11
Ada-Boost	85	94	14
Ada-SVM	80	86	18
WITMPR	76	83	24

A comparison of the detection rates of the algorithms is performed on specified percentages of the algorithm. The graphical depiction is provided in Figure 2. The DHEPSO algorithm is superior to the compared algorithms as the use of the behavior pool allows the algorithm to quickly detect changes in the traffic data and produces results with minimal computational cost. The use of different parameters to update velocity and position vectors results in better detection rates performance and better generalization performance as it enhances diversity of the two sets of particle swarms and allows the algorithm to converge rapidly without the need to overlap and mutate resulting in a higher probability of finding the global optima. Figure 2 shows a comparison of detection rates with other algorithms designed to detect network intrusion.

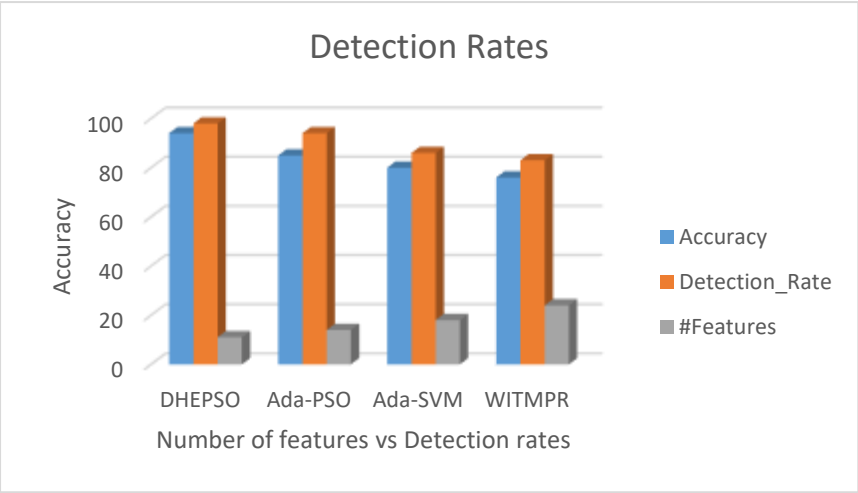

Figure 2: Comparison of detection rates with other algorithms

The accuracies achieved by the different algorithms over the NSL-KDD are compared to each other. Firstly, we applied the Friedman test to determine if there are significant statistical differences between the rankings of the compared algorithms. Secondly, the Nemenyi post hoc test using the average rank diagram is performed. To perform the Nemenyi post hoc test, the KEEL 3.0 software is used [42]. As shown in Figure 3, the algorithms whose differences are insignificant are connected with a line. The Critical Difference (CD) is shown above the average rank diagram.

The observation from the CD plot is that, DHEPSO is ranked first, followed by Ada-PSO. However, the performances are not statistically distinguishable from the performances of the Ada-SVM and WITMPR.

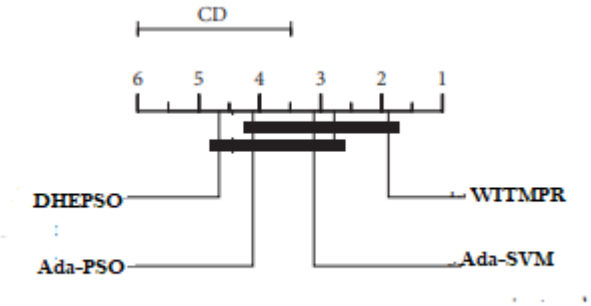

Figure 3: Average Rank Diagram

5. Conclusion

The study introduced a novel machine learning approach inspired by particle swarm optimizers to detect network intrusion efficiently and effectively using a Dynamic Heterogeneous Ensemble Particle Swarm Optimizer. This is the first study that enhances more diversity to optimize parameters to the learning algorithm in order to avoid premature convergence and increase detection rates. As a network intrusion detection approach, DHEPSO provides specific details related to the algorithm performance per class. Our proposed approach was comprehensively evaluated using several metrics to avoid bias and overfitting. Compared with

existing network intrusion detection algorithms, DHEPSO achieved excellent generalization performance that is consistent across multiple classes. DHEPSO efficiently recognized minority classes, maximized True Positive Rates and minimized False Positive Rates across multiple classes. DHEPSO is a robust framework for learning in dynamic environments associated with changing concepts like network intrusion detection. For future studies, we will focus on the use of different datasets generated by different networks that exhibit different attacks. Various strategies will be employed including combining experts' algorithmically via optimization and the performance of numerous variants of the PSO algorithm will be compared.

Compliance with Ethical Standards:

(
Funding: No funding provided for this study.

No research grants received

I declare that I have no conflict of interest.

No animals used in the research

No human beings used in the research (And/or in case humans were involved)
Ethical approval: All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards.

(If articles do not contain studies with human participants or animals by any of the authors, please select one of the following statements)
Ethical approval: This article does not contain any studies with human participants performed by any of the authors.

References

1. C.F Tsai, Y.F Hsu, C.Y Lin and W.Y Lin, “Intrusion detection by machine learning: A review”, Expert Systems with Applications, 2009.
2. M.H Aghdam, P.Kabiri, “Feature Selection for intrusion detection system using ant colony optimization”, International Journal of Network Security, Volume 18, Number 3, pages 420-432, May 2016
3. J.M Myerson “Identifying enterprise network vulnerabilities”, International Journal of Network Management 12(3), pages 135-144, 2002.
4. H.J Liao, C.H.R Lin, Y.C Lin and K.Y Tung, “Intrusion Detection System: A comprehensive review “, Journal of Network and Computer Applications, Volume 36, Issue 1, pages 16-24, January 2013.
5. P.G Teodoro, J.D Verdejo, G.M Fernandez, E.Vazquez, “Anomaly-based network intrusion detection: Techniques, Sysems and Challenges”, Computer and Security, Volume 28, Issues 1-2, pages 18-28, March 2009.
6. D.Stiawan, A.Heryanto, “Ensemble Intrusion Detection Systems”, IEEE Access, Volume 9, pages 6930-6947, 2020.
7. J.Kennedy, R.C Eberhart, “Swarm Intelligence”, Morgan Kaufmann Publishers, Inc. San Francisco, CA, USA, 2001
8. Abdulla Amin Aburomman, Mamun Bin Ibne Reaz, “A novel SVM-KNN-PSO ensemble method for Intrusion Detection Systems”, Applied Soft Computing 38, pages 360-372, 2016.
9. Ying Zhou, Thomas A. Mazzuchi, Shahram Sarkani, “M-AdaBoost-A based ensemble system for network intrusion detection”, Expert Systems with Applications 162, December 2020
10. Hariharan Rajadurai, Usha Devi Gandhi, “A stacked ensemble learning model for intrusion detection in wireless networks”, Neural Computing and Applications, May 2020.

11. Adeel Abbas , Muazzam A. Khan, Shahid Latif, Maria Ajaz, Awais Aziz Shah, Jawad Ahmad, “A New Ensemble-Based Intrusion System for Internet of Things”, Arabian Journal for Science and Engineering, 2021.
12. Mohammad Mehedi Hassan, Abdu Gumaei, Ahmed Alsanad, Majed Alrubaian, Giancarlo Fortino, “A hybrid deep learning model for efficient intrusion detection in a big data environment”, Information Sciences, Volume 513, pages 386-396, 2020.
13. Akashdeep, Ishfaq Manzoor, Neeraj Kumar, “A feature reduced intrusion detection system using ANN Classifier”, Expert Systems with Applications, Volume 88, pages 249-257, 2017.
14. Chaouki Khammassi, Saoussen Krichen, “A GA-LR wrapper approach for feature selection in network intrusion detection”, Computers and Security, Volume 70, pages 255-277, 2017.
15. Huiwen Wang, Jie Gu, Shanshan Wang, “An effective intrusion detection framework based on SVM with feature augmentation”, Knowledge Based Systems, Volume 136, pages 130-139, November 2017.
16. Yuyang Zhou, Guang Cheng, Shanqing Jiang, Mian Dai, “Building an efficient intrusion detection system based on feature selection and ensemble classifier”, Computer Networks, Volume 174, June 2020.
17. Hadeel Alazzam, Ahmad Shariah, Khair Eddin Sabri, “A feature selection algorithm for intrusion detection system based on Pigeon Inspired optimizer”. Expert Systems with Applications, Volume 148, June 2020
18. Wenhong Wei, Shuo Chen, Qiuzhen Lin, Junkai Ji, Jianyong Chen, “A multi-objective immune algorithm for intrusion feature selection”, Applied Soft Computing, Volume 95, October 2020.
19. V.N Vapnik, “The nature of statistical theory, First Edition, New York, NY, Springer, 1995.
20. M.N Murty and R.Raghawa, “Kernel-based SVM”, New York. NY, Springer, pages 57-67, 2016.
21. Andries P. Engelbrecht, “Computational Intelligence, An Introduction”, John Wiley and Sons, Chichester, December, 2002
22. R.C Eberhart, J.Kennedy, “A new optimizer using particle swarm theory”, in: Proceedings of the sixth International Symposium on Micro Machine and Human Science, Volume 1, New York, NY, pages 39-43, 1995.
23. Gui-Rong You, Yeou-Ren Shiue, Wei-Chang Yeh, Xi-Li Chen, Chih-Ming Chen, “A Weighted Ensemble Learning Algorithm Based on Diversity Using a Novel Particle Swarm Optimization Approach”, MDPI, Algorithms, October 2020.
24. B.Xue, M.Zhang and W.N Browne, “Particle Swarm Optimization for feature selection in classification, novel initialization and updating mechanisms”, Applied Soft Computing 18, pages 261-276, 2014.
25. Li Zhang, Xiaobo Chen “Feature Selection Methods Based on Symmetric Uncertainty Coefficients and Independent Classification Information”, IEEE

Access, Volume 9, pages 13845-13856, 8 January 2021.
26. Z.Karimi, M.Mansour Riahi Kashani, A.Harounabadi, “Feature ranking in intrusion detetion dataset using combination of filtering methods”, International Journal of Computer Applications; Volume 78, Number 4, pages 21-27, 2013.
27. Shi Cheng and Yuhui Shi, “Diversity Control in Particle Swarm Optimization”, IEEE Symposium on Swarm Intelligence, pages 1-9, 2011.
28. S. Yuhwi and R.Eberhart, “A modified particle swarm optimizer”, in Evolutionary Computation Proceedings, 1998, IEEE World Congress on Computational Intelligence. The 1998 IEEE International Conference 1998, pages 69-73, 1998.
29. Y.Shi and R.C Eberhart, “Empirical study of particle swarm optimization”, In Evolutionary Computation 1999 CEC 1999. Proceedings of the 1999 Congress 1999.
30. T.Blackwell and P.Bentley, “Dynamic search with charged swarms”, In Proceedings of the genetic and evolutionary computation conference, pages 19-26, CiteSeer, 2002.
31. Alexey Tsymbal, “The problem of Concept Drift: Definitions and Related Work”, Technical Report, Trinity College Dublin, Ireland, 2004.
32. M.Hansell,”Built by Animals”, New York, NY, Oxford University Press, 2007.
33. Marco A. Montes de Oca, Jorge Pena, Thomas Stutzte, “ Heterogeneous Particle Optimizers”, Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2009, 18-21 May 2009, Norway.
34. Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, Ali A. Ghorbani, “A detailed analysis of the KDD Cup 99 dataset”, IEEE Symposium on Computational Intelligence for Security and Defense Applications”, Ottawa, Canada, 8-10 July, 2009.
35. A.Rodriguez and A. Laio, “Clustering by fast search and find of density peaks”, Science, Volume 344, Number 6191, pages 1492-1496, 2014.
36. M.H Aghdam and P.Kabiri, “Feature selection for intrusion detection using ant colony optimization”, International Journal of Network Security, Volume 18, Number 3, pages 420-432, May 2016.
37. Nevrus Kaja, Adnan Shaout and Di Ma; “An intelligent intrusion detection system”, Applied Intelligence, Volume 49, pages 3235-3247, 25 March 2019.
38. S.Maza and M Touahria, “Feature selection for intrusion detection using new multi-objective estimation of distribution algorithms”, Applied Intelligence, 49(1), pages 1-21, 2019.
39. A.Bifet, G.Holmes, R.Kirkby, B.Pfahringer, “MOA: Massive Online Analysis”, Journal of Machine Learning, Res 11, pages 1601-1604, 2010.
40. J.Demsar, “Statistical comparisons of classifiers over multiple datasets”, Journal of Machine Learning Research, Volume 7, pages 1-30, 2006.

41. J.W Mikhail, J.M Fossacera, R.Lammartino, “A semi-boosted nested model with sensitivity-based weighted binarization for multidomain network intrusion detection”, ACM Transaction on Intelligent Systems and Technology, 10(3), 2019.
42. I.Triguero, S.Gonzalez, J.M Moyano, “KEEL 3.0: An Open Source Software for multi-stage analysis in data mining”, International Journal of Computational Intelligence Systems, Volume 10, Number 1, pages 1238-1249, 2017.