International Journal of Innovative Computing, Information and Control
Volume **8**, Number **12**, December **2012**
ICIC International ©2012 ISSN 1349-4198
pp. 8231–8248

A NOVEL ANOMALY-NETWORK INTRUSION DETECTION SYSTEM USING ABC ALGORITHMS

Changseok Bae[1], Wei-Chang Yeh[2,3], Mohd Afizi Mohd Shukran[4]
Yuk Ying Chung[5] and Tsung-Jung Hsieh[2]

[1]Personal Computing Research Team
Electronics and Telecommunications Research Institute
218 Gajeongro, Yuseonggu, Daejeon 305-700, Korea
csbae@etri.re.kr

[2]Department of Industrial Engineering and Engineering Management
National Tsing-Hua University
No. 101, Sec. 2, Kuang-Fu Road, Hsinchu 30013, Taiwan

[3]Advanced Analytics Institute
Faculty of Engineering and Information Technology
University of Technology Sydney
PO Box 123, Broadway, NSW 2007, Australia

[4]Department of Computer Science
National Defence University of Malaysia
Sungai Besi Camp, 57000 Kuala Lumpur, Malaysia

[5]School of Information Technologies
University of Sydney
Sydney, NSW 2006, Australia

Received March 2011; revised January 2012

Abstract. *Network Intrusion Detection Systems (NIDSs) are increasingly in demand today as the widespread of networked machines and Internet technologies emerge rapidly. As a result, many unauthorized activities by external and internal attackers within organizations need to be detected in recent years. Thus, it is crucial that organizations should have the capability to detect these unlawful activities so that the integrity of organizational information can be protected. In previous research, NIDSs have been approached by various machine learning techniques. From our knowledge, it is first time applying the Artificial Bee Colony (ABC) to solve the intrusion detection problems. In this paper, a new network intrusion system based on ABC searching algorithm has been proposed and implemented. The performance of the proposed Anomaly-based NIDS (A-NIDS) using ABC algorithm (called A-NIDS-ABC for short) has been tested using KDD-99 datasets developed by MIT Lincoln Labs. We have also compared the proposed A-NIDS-ABC with other five traditional classification algorithms. The experimental results showed that the proposed method can outperform other five popular benchmark classifiers and is suitable for the network intrusion detection.*

Keywords: Network intrusion detection systems, Artificial bee colony algorithm, Classification

1. Introduction.

With the wide spread of computer networks, security problems in computer systems have become more and more important since various intrusions or viruses may cause significant losses to information assets [1,2]. This is due to the extensive growth of Internet connectivity and accessibility to information systems worldwide.

Intrusion detection can be considered as a pattern recognition problem to define the normal and abnormal pattern in TCP/IP connections. Therefore, machine learning algorithm can play an important role in it. In order to evaluate the performance of applying different machine learning based methods to intrusion detection, the 1998 DARPA Intrusion Detection Evaluation project was conducted by MIT Lincoln Labs, which set up a LAN network and logged normal and attacked network traffic. These records were reduced and processed by domain experts to yield KDD Cup 99 dataset for competition [3]. This dataset had been used in our experiments later to find the accuracy of the proposed Anomaly-based NIDS using ABC algorithm (A-NIDS-ABC).

Moreover, this paper aims at creating a novel algorithm for detecting the intrusion data from network information using pattern recognition approach with anomaly detection technique. For evaluation purpose, the A-NIDS-ABC is compared with other five most widely used classifier such as Naives Bayes [4], Support Vector Machine (SVM) [5], Classification Tree [6], k-Nearest Neighbour (k-NN) [6], C4.5 [7], Weka [8] and Orange software tools [9].

The rest of this paper is organized as follows. Section 2 presents a brief overview of previous work in NIDS and describes the related work on machine learning based A-NIDS. Then, Section 3 will describe the principles of ABC. Section 4 will describe the proposed A-NIDS-ABC. Section 5 is dedicated to experimental setup and results. Conclusions are given in Section 6.

2. Background and Previous Works.

Intrusion detection has been an active field of research for more than two decades. In 1987, Dorothy Denning published a seminar paper An Intrusion Detection Model [10], where he discussed various security concerns, presented a definition of Intrusion Detection and discussed different types of Intrusion Detection. Most of the contemporary computer security research work is based on the milestone established by Denning.

Intrusion detection is a component of detection processes. It tries to identify if a network is under attack or not. NIDS are classified firstly as host Intrusion Detection System (IDS) and NIDS based on location from which it collects data; secondly as signature based IDS and anomaly based NIDS (A-NIDS).

The most popular approach for today's NIDS is still signature based [11]. It performs intrusion detection by searching for predefined content or a fixed sequence of bytes in a single packet. This approach works well if the pattern of attacks could be found in advance. This approach is reliable and has low false negatives rate for detecting known attacks. However, it cannot detect new attacks or mutation of known attacks because their fingerprints have not been discovered. Anomaly-based detection (A-NIDS) builds models of normal behaviour in a system, and attempt to identify attacks on deviations from the profiles of normal network activities. Anomaly-based detectors can detect new and completely unknown attacks but may have high false positive rates [11].

Numerous machine learning methods have been applied to NIDS. For example, artificial immune theory [12], Bayesian parameter estimation [13], clustering [14], data fusion [15] and neural networks [16,17] have been adopted to build good detection model. Support Vector Machines (SVM) [18] is a newly proposed machine learning approach. Owing to its remarkable characteristics such as good generalization performance, and the absence of local minimal and the sparse representation of solution, SVM has become a popular research method in intrusion detection. Literature in [16] has reported the good results of applying SVM to NIDS. Moreover, Bao et al. developed network intrusion detection system by using SVM as a solution in [19] to solve the poor generalizing ability problems of other classifiers.

Besides that, the prior network intrusion detection system based on fuzzy classifier [20] has been developed. Another example had been using iterative rule learning using fuzzy rule based genetic classifier in intelligent intrusion detection system developed by Őzyer et al. [21]. Yao et al. [22] developed an A-NIDS using a dynamic decision boundary by extracting the fuzzy rule sets from support vectors which are the training results of SVM. Yao et al. applied these fuzzy functions from the rule set to develop a fuzzy classifier or a fuzzy decision boundary to NIDS in [21].

The other prior research work had applied optimization methodologies to NIDS. They had used, Particle Swarm Optimization (PSO) [24] and Ant Colony Optimization (ACO) [25] and Genetic Algorithm (GA) [23,26,27] to develop the detection model generation and intrusion feature selection. The main reasons why most researchers used GA in intrusion detection systems are due to the facts that GA tends to be robustness to noise, to self learning capabilities and the initial rules can be built randomly so there is no need of knowing the exact way of attack machinery at the beginning. In addition, ACO algorithm has also been used in some intrusion detection systems [28]. Another commonly used optimization algorithm is PSO. This is due to its advantage in solving non-linear programming problems [29]. From our knowledge, no research has applied the ABC algorithm to NIDS. This paper is the first time applying ABC to NIDS by taking the advantage of the rule mining associations.

3. Artificial Bee Colony (ABC) Algorithm.

In recent years, several biological and natural processes had been influencing the methodologies in science and technologies. One of them is the newly developed swarm intelligence algorithm based on the behavior of the bees namely Artificial Bee Colony (ABC) [30-33]. Some recent application employing ABC have been presented in various fields, e.g., [34,35].

In the ABC algorithm, each food source represents a possible solution (i.e., it can be treated as a particle in PSO) to the considered problem and the size of a food source represents the quality of the solution. This algorithm represents a colony of artificial bees (referred to here simply as bees) comprising three typesemployed, onlookers and scouts. The first half of the bee colony comprises employed bees, while the second half comprises onlookers. The ABC algorithm assumes only one employed bee per food source – namely, the number of food sources equals the number of employed bees. When a food source is abandoned, a employed bee working on that food source becomes a scout until a new food source is found, at which point it once again becomes employed. The pseudo-code of the ABC algorithm is as follows.

ABC algorithm:

```
Input: initial solutions.
Output: optimal solution.
BEGIN
  1. Generate the initial population $X_h$ randomly, $h = 1, 2, \cdots, SN$.
  2. Evaluate the fitness ($fit_h$) by (1), $h = 1, 2, \cdots, SN$.
  3. Cycle = 1
  4. Repeat
  5. FOR (employed phase) {
     5.1 Produce a new solution $S_i$ by (2)
     5.2 Calculate $fit_i$ by (1) for $S_i$
     5.3 Apply greedy selection process }
  6. Calculate the probability $P_h$ for $X_h$ by (3)
```

```
 7. FOR (onlooker phase) {
    7.1 Apply Roulette Wheel to select a solution $X_h$ depending on $P_h$
    7.2 Produce a new solution $S_i$ by (2)
    7.3 Calculate $fit_i$ by (1) for $S_i$
    7.4 Apply greedy selection process }
 8. IF (scout phase)
    There is an abandoned solution for the scout depending on limit
    THEN
    Repeat it with a new one which will by randomly produced by (5)
 9. Memorize the best solution so far
10. Cycle = Cycle+1

Stop if Cycle = MGN
END.
```

Initially, the ABC algorithm generates a randomly distributed initial population of SN solutions (food source positions), where SN denotes the number of employed or onlooker bees. Each solution X_h ($h = 1, 2, \cdots, SN$) is a d-dimensional vector. Here, d represents the number of optimization parameters. The population of the positions (solutions) is subject to repeated cycles, $C = 1, 2, \cdots$, Maximum Generation Number (MGN), of the search processes of the employed bees, onlooker bees and scout bees. The subsequent work is to evaluate the food source generated by a grid scheme by a so-called fitness function, while the design of the fitness function is a crucial point in ABC, which determines what ABC should optimize.

To guide ABC toward more highly fit positions of the search space of food sources, one or more fitness values must be assigned to each solution. Usually, a standard ABC is designed for optimization problems with only one objective. Each fitness value can be determined by a fitness function that we want to optimize. In this paper we designed the fitness function as follows:

$$fit_i = \begin{cases} \dfrac{1}{f_i + 1}, & f_i \geq 0 \\ 1 + |f_i|, & f_i < 0 \end{cases} \tag{1}$$

where f_i represents the objective value of i-th solution.

After a solution is generated, that solution is improved via a local search process called greedy selection, carried out by onlooker and employed bees; according to this process, if the size (fitness) of the candidate source is better than the present source, the bee abandons the present source in favor of the candidate one. This is achieved by adding to the current value of the selected parameter the product of a uniform variable in $[-1, 1]$ and the difference in values of this parameter for this food source and some other randomly selected food source.

Specifically speaking, suppose each solution comprises d parameters, and let $X_h = (Xh_1, Xh_2, \cdots, Xh_d)$ denote a solution with parameter values $Xh_1, Xh_2, \cdots, Xh_d$. To determine a solution S_h near X_h, a solution parameter j and another solution $X_k = (Xk_1, Xk_2, \cdots, Xk_d)$ are selected randomly. Except for the value of the selected parameter j, all other parameter values of S_h are the same as X_h, namely, $S_h = (Xh_1, Xh_2, \cdots, Xh_{(j-1)}, Sh_j, Xh_{(j+1)}, \cdots, Xh_d)$. The value Sh_j of the selected parameter j in S_h is determined using the following formula:

$$S_{hj} = X_{hj} + u\,(X_{hj} - X_{kj}) \tag{2}$$

where u denotes an uniform variable in $[-1, 1]$. If the resulting value falls outside the acceptable range for parameter j, it is set to the corresponding extreme value in that range.

The ABC algorithm is an iterative algorithm, and first associates all employed bees with randomly generated food sources (solutions). Next, during iteration, every employed bee determines a food source near its currently associated food source and evaluates its nectar amount (fitness). If the food source is larger than the food source it is currently working with, the bee abandons the current food source in favor of the new one; otherwise, it remains at the original food source. Once all employed bees have completed this process, they share the information on the size of the food sources with the onlookers, each of whom then selects a food source, with the probability of their selecting any individual source being proportional to the amount of nectar contained in that food source; that is, the probabilities established by Equation (3) and the probability P_h of selecting a food source (solution) X_h is determined.

$$P_k = \frac{fit_h}{\sum_{h=1}^{SN} fit_h} \tag{3}$$

where fit_h denotes the fitness of the solution represented by food source h, and SN represents the total number of food sources. Clearly, this scheme leads to good food sources attracting more onlookers than bad ones. After all onlookers have selected their food sources, each onlooker determines a food source near his chosen food source and calculates its fitness. The best among the neighboring food sources as determined by the onlookers associated with a particular food source, h, is selected as the next location for food source h.

If a solution represented by a particular food source does not improve for a predetermined number of iterations, that food source is abandoned by its associated employed bee and the employed bee becomes a scout; namely, the bee randomly searches for a new food source. This arrangement is tantamount to assigning a randomly generated food source (solution) to the scout and again changing its status from scout to employed. After determining the new location of each food source, another iteration of the ABC algorithm begins.

The process is repeated until the termination condition is satisfied. In the ABC algorithm, if a position cannot be further improved through a predetermined number of cycles, the food source is assumed to be abandoned. The value of the predetermined number of cycles is an important control parameter of the ABC algorithm, and is called the "*limit*" for abandonment, and the relation defined as Equation (4). Assuming the abandoned source is X_h, and $j \in 1, 2, \cdots, d$, then the scout discovers a new food source to be replaced with X_h, and the operation is defined as Equation (5).

$$limit = SN \times d \tag{4}$$

$$X_h^j = X_{\min}^j + \text{rand}[0, 1] \left(X_{\max}^j - X_{\min}^j \right) \tag{5}$$

In the later section, it is imperative that the control parameters such as "*limit*" value and "*MGN*" should be tested so that we can know how these control parameters would affect the A-NIDS-ABC.

4. The Proposed Anomaly-based NIDS (A-NIDS) Using ABC.

Many NIDS employ techniques for both signature and anomaly based IDS. Signature based NIDS can only derive from well known attacks, whereas anomaly IDS can derive from both unknown and known attacks. Based on this reason, we have chosen the anomaly technique in the proposed system.

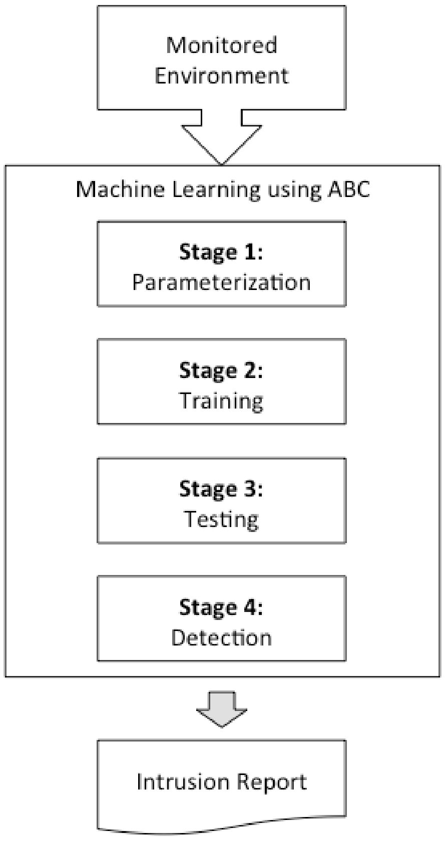

Figure 1. Functional architecture of the proposed A-NIDS-ABC

Basically, the anomaly intrusion detection technique can be categorised into three categories: 1) statistically based, 2) knowledge based, and 3) machine learning based. The traditional method is called "statistical" or "behavioural" anomaly detection which selects key statistics about network traffic as features for a model trained to recognize the normal activity [36]. Another anomaly technique is knowledge based anomaly detection which compromise of set of rules. These set of rules are the basis of the desired model that determine the system behavior [11]. In addition, there is another anomaly technique based on machine learning which we have adopted in the proposed A-NIDS-ABC. We used this technique because it can predict the future patterns and be more flexible in training the labelled data (i.e., KDD-99 [3] datasets).

Figure 1 shows the functional architecture of A-NIDS-ABC. We have implemented the machine learning using ABC to recognise all the patterns existed in the NIDS datasets. Precisely speaking, we can divide the A-NIDS-ABC into four steps and the detail description can be found in the following sub-sections.

In machine learning process, classification rule is crucial to identify the correct class in a population. The aim of classification rule is to find a set of rules which can identify the specific class from different groups. The process of constructing classification rule on ABC algorithm has been shown in Figure 2.

Stage 1 (Parameterization) in Figure 1 is the initialization stage. Firstly, we need to set the lower bound and upper bound for every attribute. The procedures are defined in Equations (6) and (7) as below:

$$Lowerbound = f - k_1 * (F_{\max} - F_{\min}) \tag{6}$$

$$Upperbound = f - k_2 * (F_{\max} - F_{\min}) \tag{7}$$

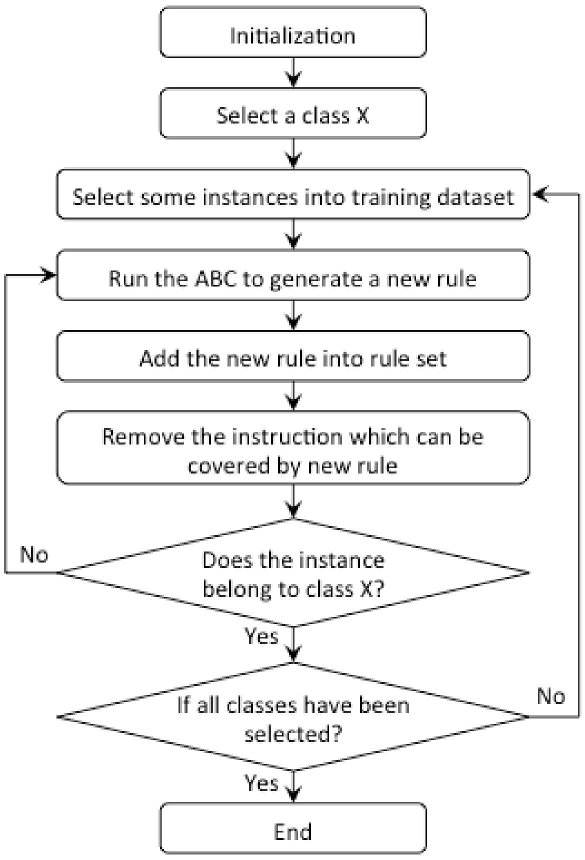
Initialization
Select a class X
Select some instances into training dataset
Run the ABC to generate a new rule
Add the new rule into rule set
Remove the instruction which can be covered by new rule
Does the instance belong to class X?
No
Yes
If all classes have been selected?
No
Yes
End

Figure 2. Flowchart to generate rules for ABC data mining

Feature 1		**Feature 2**		......	**Feature *N***	
Lower Bound	Upper Bound	Lower Bound	Upper Bound		Lower Bound	Upper Bound

Figure 3. Format for the classification rule

where $F_{\max}$ and $F_{\min}$ are the maximum and the minimum value of a specific feature respectively. f is the original value of the feature. k_1 and k_2 are two random values between 0 and 1. Moreover, the classification rule mining algorithm can discover the rules for each class automatically. For the selected class, the rules will be found iteratively until the rule set can cover all instances belong to that class. Every single rule abides by the rule structure and the rule set consists of many rules.

As seen from Figure 3, attributes of a certain dataset have their own lower and upper bound values. The lower bound is the lowest value for this rule and the upper bound is the highest value for this rule. There are three other values associated with the classification rule described following: predictive class (i.e., selected class X), the fitness value (Equation (9)), and the cover percentage of the rule (Equation (11)).

Fitness value is an important component in classification rule mining. Witten and Frank [37] suggested that fitness function is required to determine the quality of a certain classifier. Therefore, fitness function can be considered as the evidence to distinguish between the good and bad individuals. According to Correa et al. [38], the measurement of how well that model classifies test examples can be based on Equation (8). However,

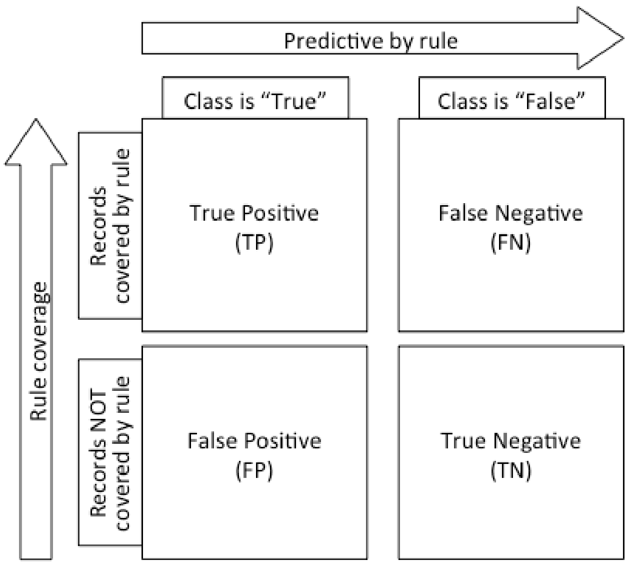

Figure 4. Definition for the parameters TP, TN, FP and FN

the proposed machine learning process uses Equation (8) instead of Equation (9) as the fitness function. This is because Equation (9) is more suitable for datasets that has a highly unbalanced class distribution such as KDD-99 datasets which has been proven by [38].

$$\text{Standard accuracy rate} = \frac{TP + TN}{TP + FP + FN + TN}. \tag{8}$$

$$\text{Fitness value} = \frac{TP}{TP + FN} \times \frac{TN}{TN + FP}. \tag{9}$$

In the proposed A-NIDS-ABC, two assumptions have been used in the proposed fitness function. Firstly, when the proposed method exams the type of the record, it will measure every features in the record. If the value of a feature is between the lower bound and upper bound for this feature, it is called the feature can be covered by the rule. If all features for a record can be covered by the rule, it means that the record can be covered by the rule. Secondly, if the class of the evaluated record equals to the predictive class by the rule, the record has the class predicted by the rule.

The values of TP, FN, FP and TN in Figure 4 are the number of different record types and representing of True Positives, False Negatives, False Positives and True Negatives associated with the rule respectively. For instance, parameter TP represents the number of records covered by the rule that have the class predicted by the rule, whereas parameter TN is the number of records which has not been covered by the rule and that does not have the class predicted by the rule.

In data mining, the local search strategy is a fundamental components to accurately classify the datasets such as KDD-99 for NIDS [37]. The proposed A-NIDS-ABC uses the original ABC local search strategy. This is crucial for gaining high quality accuracy for classifying the KDD-99 dataset. In ABC algorithm, the artificial bees need to do the local search for finding the new possible solution. Equation (2) shows the local search strategy of the A-NIDS-ABC which similar to local search strategy for ABC optimization algorithm.

Stage 2 (Training) in Figure 1 is the implementation of rule pruning, which is important for the A-NIDS-ABC. Rule pruning is an essential data mining technique and has been used in the vast majority of rule induction algorithms [37]. Pruning can improve the

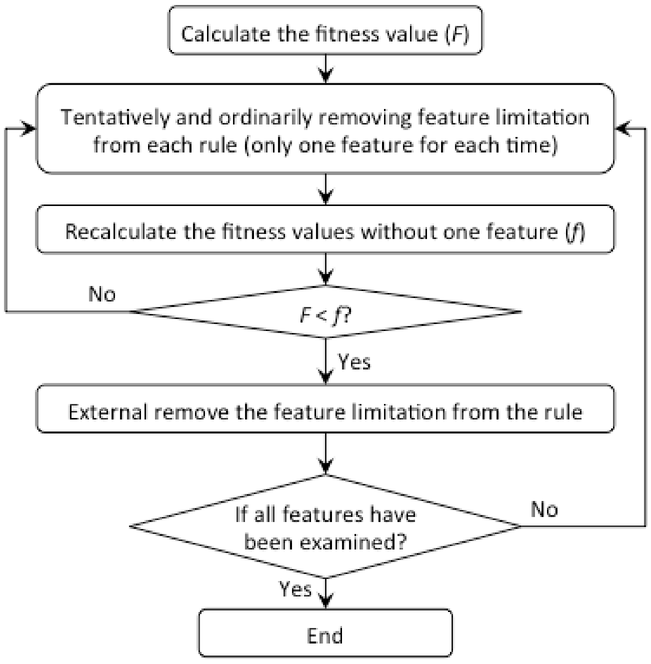
Calculate the fitness value (F)
Tentatively and ordinarily removing feature limitation from each rule (only one feature for each time)
Recalculate the fitness values without one feature (f)
No
F < f?
Yes
External remove the feature limitation from the rule
If all features have been examined?
No
Yes
End

FIGURE 5. Flowchart of rule pruning procedure

quality of a rule by removing the irrelevant terms from the rule antecedent, and then improve both the predictive accuracy and the comprehensibility of the rule. In this study, the main reason why we adopt rule pruning is to remove the redundant feature limitation that might have included in the rule sets. This is because some irrelative attributes will negatively influence the classification result, thus affect the classification accuracy of A-NIDS-ABC. The details of the rule pruning process have been shown in Figure 5. This process will be repeated until all rules in the rule set are evaluated.

After removing redundant feature limitation that might have been unnecessary included in rule sets, the pruned rule set will be used to predict the new data which their classes are unknown. On the other hand, one testing data record may be covered by more than one rule for different classes in some cases. When this happen, it would involve **Stage 3** (Testing) in Figure 1 where the prediction strategy will determine which class should be predicted. To resolve the above problem, the three main steps for prediction approach are drawn as follows:

(1) Calculate the prediction value for all rules which can cover the test data record;
(2) Accumulate these prediction values according to different possible classes;
(3) Select the class which has the highest prediction value as the final class.

After the procedure of prediction strategy, the core is the prediction function which is used to compute the prediction value for each rule. It is defined in Equation (10).

$$\text{Prediction value} = (\alpha \times \text{rule fitness value}) + (\beta \times \text{rule cover percentage}), \qquad (10)$$

where $\alpha \in [0, 1]$ and $\beta = (1 - \alpha)$ are two weighted parameters associated with rule fitness value and rule cover percentage respectively. As mentioned above, Equation (9) can calculate the fitness value for each rule; the rule cover percentage defines that the proportion of the records which was covered by the rule that have the class predicted by the rule (TP). It is calculated by the expression shows as follows:

$$\text{Cover percentage} = \frac{TP}{N}, \qquad (11)$$

where N is the total number of records which belongs to the predicted class by the rules. The prediction strategy balanced the effect of fitness value and cover percentage for the final predicted class. The value of α and β need to be chosen carefully since they will affect the classification accuracy.

In sum, **Stage 1** (Parameterization) initializes all control parameters such as the number of data size, the number of generation and the bound value limitation. Here, K-fold cross-validation is used for assessing the accuracy of our testing results, where $K = 10$, the 10-fold cross-validation is the most commonly used algorithm to test the accuracy of classification [39]. In **Stage 2** (Training) and **Stage 3** (Testing), for each time, a single subset is used as testing data, and the remaining $K - 1$ subsets are retained as training data. In addition, each of the K subsets is employed only once as the validation data. After two data sets have separated, the training set in **Stage 3** (Training) will be used to find the classification rule set. In **Stage 3**, we calculate the accuracy for each validation data set, and then let the average of K validations be the final accuracy. Finally, in **Stage 4** (Detection), the accuracy will be analysed so that decision on suspicious unauthorized activities can be concluded and thus detection result can be produced accurately.

5. Experiments.

In this section, we discuss how we applied ABC algorithm to detect the intrusion data from the network information. The test dataset we selected is KDD Cup 1999 Dataset [3] that used for the Third International Knowledge Discovery and Data Mining Tools Competition, which was held in conjunction with KDD-99 the Fifth International Conference on Knowledge Discovery and Data Mining. These data were from MIT Lincoln Labs and its objective was to survey and evaluate research in intrusion detection. According to the description from the homepage of KDD-99, network attacks can be included into the following four main categories:

- DOS: denial-of-service, e.g,. syn flood;
- R2L: unauthorized access from a remote machine, e.g., guessing password;
- U2R: unauthorized access to local superuser (root) privileges, e.g., various “buffer overflow” attacks;
- Probing: surveillance and other probing, e.g., port scanning.

The main reason or the main motivation of using KDD Cup 1999 Dataset [3] as a medium to achieve theritic results is because to show that the proposed method has the advantage of becoming an efficient classification algorithm based on the Artificial Bee Colony (ABC) that performs better than state-ofthe-art classification algorithms such as Support Vector Machine (SVM), Nearest Neighbour (K-NN), Decision Trees, and Bayesian Method when applied to the Network Intrusion Detection System (NIDS). This comparison will be explained in detail later in this paper.

5.1. Conditions for experiments.

The computer configurations of running this experiment, the values assigned to the important parameters of the proposed method is shown below:

Item	Configuration
CPU	Intel®Core$^{\text{TM}}$2 P7350 @ 2.00 GHz
Memory	4.00GB DDR2 SDRAM
Hard Disk	320GB (5400RPM)
Operating System	Windows Vista$^{\text{TM}}$ Home Premium

Computation issues have implicated the time to process the proposed algorithm. This would be a disadvantage and thus can be overcome by eliminating redundant parameter within the proposed method.

5.2. **Experiments design and data pre-process.** The testing dataset in the experiments contains a total of 24 training attack types in the above four categories, which means there are 24 different classes in the dataset. In addition, each network connection record includes the information that has been separated into 41 features plus 1 class label in the KDD-99 dataset. Details about these 41 features are listed in Table 1.

We randomly selected 4,000 records from KDD-99 dataset for testing in our experiments, and mapped these string features into numeral type. The mapping formats are defined in Tables 2-4 as follows.

After mapping the testing data, we discovered that many records in the same column have the same value, such as all 0 or 1 in the original file. Thus, it is helpful to remove those redundancy features by using feature selection technologies. The feature selection

TABLE 1. The features for network intrusion data

No	Feature Name	No	Feature Name	No	Feature Name	No	Feature Name
1	duration	12	logged_in	22	is_guest_login	32	dst_host_count
2	protocol_type	13	num_compromised	23	count	33	dst_host_srv_count
3	service	14	root_shell	24	srv_count	34	dst_host_same_srv_rate
4	src_bytes	15	su_attempted	25	serror_rate	35	dst_host_diff_srv_rate
5	dst_bytes	16	num_root	26	srv_serror_rate	36	dst_host_same_src_port_rate
6	flag	17	num_file_creations	27	rerror_rate	37	dst_host_srv_diff_host_rate
7	land	18	num_shells	28	srv_rerror_rate	38	dst_host_serror_rate
8	wrong_fragment	19	num_sccess_files	29	same_srv_rate	39	dst_host_srv_serror_rate
9	urgent	20	num_outbound_cmds	30	diff_srv_rate	40	dst_host_rerror_rate
10	hot	21	is_hot_login	31	srv_diff_host_rate	41	dst_host_srv_rerror_rate
11	num_failed_logins						

TABLE 2. Feature 3's mapping format

No	String	No	String	No	String	No	String	No	String	No	String
1	http	12	3_data	22	link	32	netbios_ssn	42	k36	52	supdup
2	smtp	13	private	23	host16s	33	netbios_dgm	43	Z39_50	53	echo
3	ftp	14	time	24	iso_tsap	34	exec	44	imap4	54	discard
4	domain	15	remote_job	25	csnet_ns	35	login	45	sql_net	55	systat
5	finger	16	name	26	pop_3	36	shell	46	ldap	56	day14
6	4_u	17	whois	27	pop_2	37	printer	47	vmnet	57	netstat
7	lauth	18	mtp	28	sunrpc	38	efs	48	bgp	58	urp_i
8	telnet	19	gopher	29	uucp_path	39	courier	49	nnsp	59	IRC
9	eco_i	20	rje	30	nntp	40	uucp	50	l_443	60	other
10	ecr_i	21	ctf	31	netbios_ns	41	k35	51	ssh	61	X11
11	ntp_u										

TABLE 3. Feature 2's mapping format

String	Number
tcp	1
icmp	2
udp	3

Table 4. Feature 4's mapping format

String	Number
SF	1
S2	2
REJ	3
S0	4
RSTR	5
RSTO	6
S1	7

Table 5. The detail about the selected 6 features from KDD-99 dataset [40]

Feature Name	No. in Table 1	Description
protocol_type	2	type of the protocol, e.g., tcp and udp
flag	6	normal or error status of the connection
srv_count	24	number of connections to the same service as the current connection in the past two seconds
rerror_rate	27	% of connections that have "REJ" errors
dst_host_same_srv_count	34	% of connections from the same host with same service to the destination host during a specified time window
dst_host_diff_srv_rate	35	% of connections from the same host with different service to the destination host during a specified time window

can improve the performance of intrusion detection and decrease the computation time. According to [40], 6 features have been selected by using Rosetta rough set software. For the experiments in the later Section 5.2, we had used the same 6 selected features from [40] which have listed in Table 5 for the 4,000 randomly selected records. However, there are some deficiencies using these selected records. This is because the selected records are selected manually. The experimental results would be more reliable and more accurate if the selected records are done automated randomly selected record so that the results of the classification accuracy are unbiased.

5.3. **Experiment results.**

5.3.1. *Experiment 1: Comparison with existing classifiers.* To further proving the robustness of the proposed A-NIDS-ABC, we have also tested the same testing data by using other five popular data mining algorithms from Weka [8] and Orange [9] software tools. The Naives Bayes [4], Support Vector Machine (SVM) [5,42], Classification Tree [6], k-Nearest Neighbour (k-NN) [6] and C4.5 classifier [7] were selected to run the KDD-99 dataset. For each classification algorithm, we used their default control parameter. The comparison results of these five methodologies and ABC algorithm are shown in Table 6.

From Table 6, we found that the Classification Tree algorithm is not suitable for network intrusion detection, since the classification accuracy of this classifier is dramatically lower than other methods. Although Naives Bayes algorithm can reach 95.47%, the A-NIDS-ABC can outperform the Naives Bayes algorithm by at least 3%. As for SVM, k-NN and C4.5, the A-NIDS-ABC can outperform by at least 5%. Therefore, according to the experimental results in Table 6, we can conclude that the proposed A-NIDS-ABC can identify the network attack efficiently when compared to other methods. However, the results would be more accurate if more methodologies in classification are tested and

TABLE 6. Comparison of the A-NIDS using different classifiers for KDD-99 data set

Algorithm	Average Accuracy	Best Accuracy	Standard Deviation
ABC	98.5%	99.5%	0.004
Naives Bayes	95.47%	98.02%	0.029
SVM	93.4%	95.01%	0.0002
Classification Tree	78.5%	79.9%	0.0022
k-NN	93.75%	94.6%	0.0357
C4.5	93.94%	94.5%	0.0155

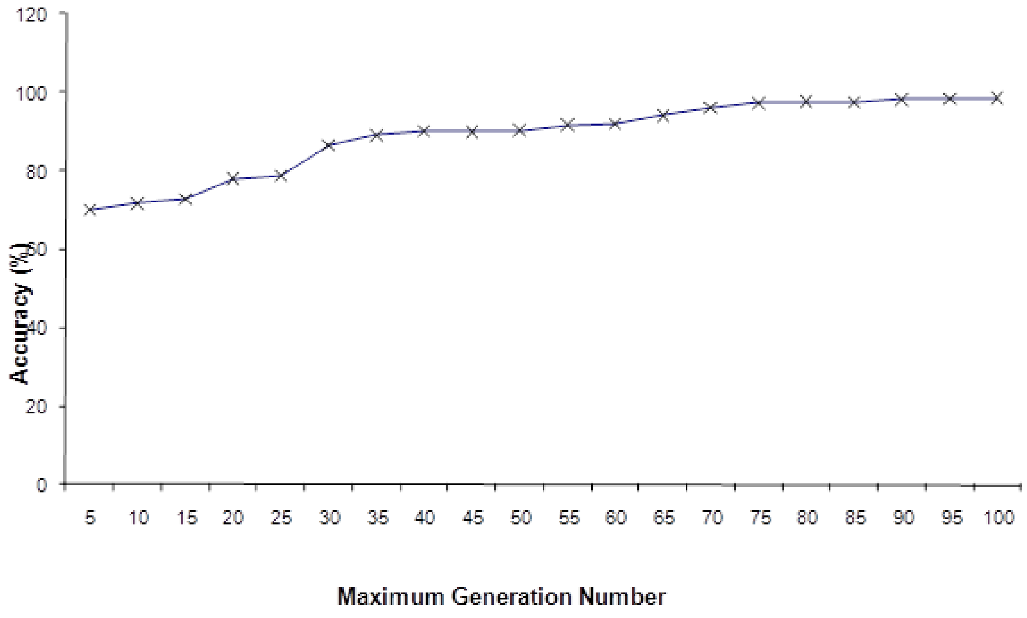

FIGURE 6. The effect of maximum generation number (MGN) on the performance of A-NIDS-ABC

compared to the proposed method. Thus, would increase the integrity of the classification accuracy.

5.3.2. *Experiment 2: Testing for the control parameters.* Another experiment is conducted in determining the effect of Generation Number to the accuracy of the A-NIDS-ABC. Similarly, this experiment has been performed in literature by Akay and Karaboga [41]. In this experiment, we have tested the classification accuracy using 10 types of Generation Numbers ranging from 5 to 100. Figure 6 shows that low values of the Generation Number worsen the performance. But when Generation Number gradually increases, the accuracy of the A-NIDS-ABC will be steady and become constant. This shows that the Generation Number can only affect the A-NIDS-ABC in low generation part. This can also indicate that the proposed A-NIDS-ABC is flexible and can find the optimal solution quickly when the MGN equals to 30.

However, there is also another issue of solution that cannot be improved after predetermined number of generations. This is referred as "*Limit*" and its value can be calculated using Equation (4) which has mentioned earlier in this paper. Moreover, "*Limit*" is another control parameter in the ABC. "*Limit*" determines whether the food source should

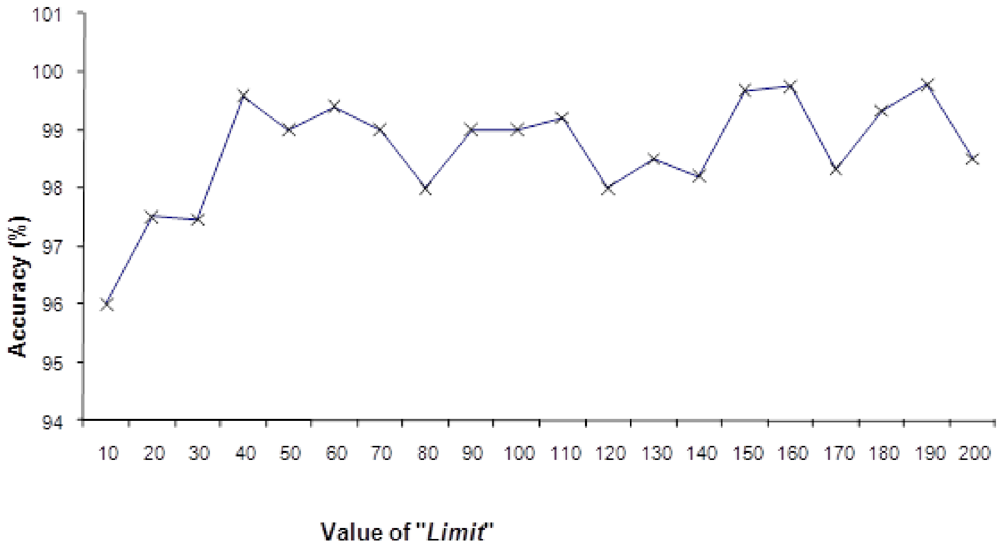

Figure 7. The effect of “Limit” for A-NIDS-ABC

be abandoned [33]. For analysing the effect of this control parameter, we have conducted an empirical experiment where the “*Limit*” values have been set to the value from 10 to 200. According to literature by Akay and Karaboga [41] “*Limit*” values plays important in determining the proposed method is robust or not. Furthermore, the results achieved in the proposed method are superior to in literature [41].

From Figure 7, we found that when the “*Limit*” values were gradually increased, the accuracy of A-NIDS using ABC did not change significantly. This can indicate that the diversity of the individuals in the population is sufficient and the solutions proceed with evolution since the number of solution is large enough. Therefore, ABC based A-NIDS did not need scouts to explore new regions.

5.3.3. *Experiment 3: Testing for the data size.* In order to analyse how testing data size will affect the performance of the A-NIDS-ABC or we can find out which testing data size can be large enough for a fixed dimension. We have kept the testing data sizes constant and changed the testing data sizes as 500, 1,000, 1,500, 2,000, 2,500, 3,000, 3,500 and 4,000. Results of this experiment are reported in Figure 8. These data sizes are quite similar to the data sizes tested in literature [41]. To summarize the results in Figure 8, the performance of the A-NIDS-ABC has increased as the dimensionality of the problem was increased from 500 to 3,000. However, the performance of the ABC based A-NIDS maintained at 98.5% for 3,000 to 4,000. Therefore, the superior performance of the proposed A-NIDS-ABC can prove for its robustness in different data sizes.

Furthermore, an empirical experiment had been done to analyse the effect of testing data Dimension (d) in classification accuracy of the A-NIDS-ABC using different classifiers. In fact, we want to find the optimum number of d. In this experiment, we kept the data size constant by using 1,000 randomly selected data and changed the dimension of the problem as 1, 2, 3, 4, 5 and 6. Results of this experiment have been reported in Table 7. From the result in Table 7, for a fixed data size of 1,000, the performance of the Classification Tree algorithm severely deteriorated as the dimension of the problem was increased. The proposed A-NIDS-ABC had superior performance over other five classifiers with 97.5% accuracy when $d = 6$. Therefore, it shows that when the dimension was increasing, for a fixed number of populations, the proposed ABC based A-NIDS can

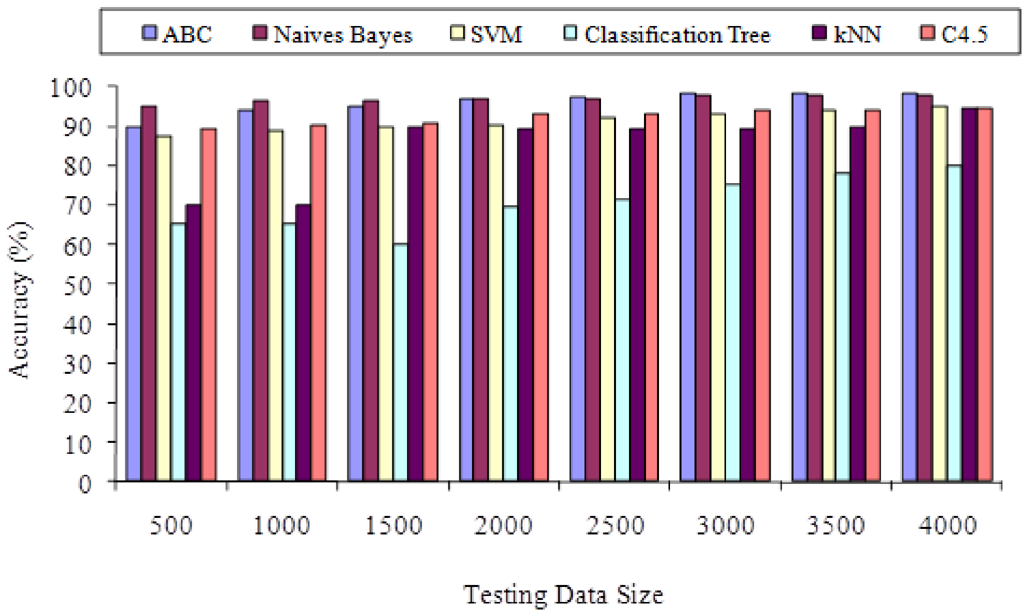

FIGURE 8. Summary of overall testing results for different data sizes using six different classifiers in A-NIDS-ABC

TABLE 7. Classification results of six different classifiers using six different dimensions

Methods	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$	$d = 6$
	(2)	(2,6)	(2, 6, 24)	(2, 6, 24, 27)	(2, 6, 24, 27, 34)	(2, 6, 24, 27, 34, 35)
ABC	100%	99.87%	98.7%	97.9%	97.6%	97.5%
Naives Bayes	99%	98%	97.5%	97.37%	97.11%	97.02%
SVM	100%	98%	97.5%	94.44%	94.34%	94.01%
Classification Tree	95%	95%	93.76%	80%	79%	79.9%
k-NN	100%	96.43%	3.46%	93.40%	93.56%	93.6%
C4.5	100%	97%	97.56%	95.53%	94.31%	93.34%

still produce reasonable good result even for 6 dimensions ($d = 6$). Similarly, [41] also shows the same result as the proposed method. However, [41] can be considered inferior compared to the proposed method because the proposed method has been tested using more ranges in data sizes and thus the results become more reliable.

6. Conclusions and Discussion.

The Network Intrusion Detection System (NIDS) has become a critical component of an organizations security strategy. However, deployment of network-based intrusion detection brings with a number of potential pitfalls, which can compromise security. An ideal network based intrusion detection deployment must provide 100% network intrusion coverage and ensure network availability. This includes recognising potential threat or unauthorised activity. One way of doing this is to use anomaly network intrusion detection technique (A-NIDS). A-NIDS is required as an additional wall for protecting systems and is useful not only in detecting successful intrusions, but also provides important information for timely countermeasures. Therefore, we have proposed a new A-NIDS using artificial bee colony (ABC) machine learning (A-NIDS-ABC). This paper is first time applying ABC machine learning in A-NIDS. We have tested the proposed A-NIDS-ABC with KDD Cup 1999 dataset. The experiment results reveal that the proposed A-NIDS is very promising and has outperformed other five traditional classifiers such as Naives Bayes, Support Vector Machine (SVM), Classification Tree, k-Nearest Neighbour (k-NN) and C4.5.

In addition, several experiments have been conducted to test the important control parameters such as MGN, "Limit" values, and varying testing data sizes. More important among them, varying testing data sizes results showed that the A-NIDS-ABC can maintain high classification accuracy even though the testing data size increases. This is the most main point because the A-NIDS-ABC is able to preserve its robustness for different population sizes.

In sum, the proposed A-NIDS-ABC is flexible and adaptable in classifying intrusion data sets. Therefore, it can be considered as an effective A-NIDS tool to recognise known and unknown attacks within organizational network.

Acknowledgment. This work was supported by the Global Frontier R&D Program on <Human-centered Interaction for Coexistence> funded by the National Research Foundation of Korea grant funded by the Korean Government (MEST) (NRF-M1AXA003-2011-0028371).

REFERENCES

[1] A. Yu, J. Tsai and T. Weigert, An automatically tuning intrusion detection system, *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol.37, no.2, pp.373-384, 2007.

[2] W. Hu, W. Hu and S. Maybank, AdaBoost-based algorithm for network intrusion detection, *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol.38, no.2, pp.577-583, 2008.

[3] *KDD Cup 1999 Data*, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[4] G. H. John and P. Langley, Estimating continuous distributions in Bayesian classifiers, *Proc. of the 11th Conference on Uncertainty in Artificial Intelligence*, pp.338-345, 1995.

[5] C. Cortes and V. Vapnik, Support-vector networks, *Machine Learning*, vol.20, no.3, pp.273-297, 1995.

[6] R. Caruana and A. Niculescu-Mizil, An empirical comparison of supervised learning algorithms, *Proc. of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, USA, 2006.

[7] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufman, San Mateo, CA, USA, 1992.

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. Witten, The WEKA data mining software: An update, *ACM SIGKDD Explorations Newsletter*, vol.11, no.1, pp.10-18, 2009.

[9] J. Demšar, B. Zupan, G. Leban and T. Curk, Orange: From experimental machine learning to interactive data mining, *LNAI*, vol.3202, pp.537-539, 2004.

[10] D. E. Denning, An intrusion-detection model, *IEEE Transactions on Software Engineering*, vol.13, no.2, pp.222-232, 1987.

[11] P. García-Teodoro, J. Díaz-Verdejo, G. Marciá-Fernández and E. Vázquez, Anomaly-based network intrusion detection: Techniques, systems and challenges, *Computers & Security*, vol.28, pp.18-28, 2009.

[12] X. R. Yang, J. Y. Shen and R. Wang, Artificial immune theory based network intrusion detection system and the algorithms design, *Proc. of 2002 International Conference on Machine Learning and Cybernetics*, Beijing, China, pp.73-77, 2002.

[13] S. Cho and S. Cha, SAD: Web session anomaly detection based on parameter estimation, *Computers & Security*, vol.23, no.4, pp.312-319, 2004.

[14] S. H. Oh and W. S. Lee, An anomaly intrusion detection method by clustering normal user behaviour, *Computer & Security*, vol.22, pp.596-612, 2003.

[15] Y. Wang, H. Yang, X. Wang and R. Zhang, Distributed intrusion system based on data fusion method, *Proc. of the 5th World Congress on Intelligent Control and Automation*, Hangzhou, China, pp.4331-4334, 2004.

[16] A. H. Sung and S. Mukkamala, Identify important features for intrusion detection using support vector machines and neural networks, *Proc. of the 2003 Symposium on Application and the Internet*, pp.209-217, 2003.

[17] S. Han and S. Cho, Evolutionary neural networks for anomaly detection based on the behavior of a program, *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, vol.36, no.3, pp.559-570, 2006.

[18] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, 1995.

[19] X. Bao, T. Xu and H. Hou, Network intrusion detection based on support vector machine, *Proc. of the Management and Service Science*, pp.1-4, 2009.

[20] C. Chen, S. Mabu, C. Yue, K. Shimada and K. Hirasawa, Network intrusion detection using fuzzy class association rule mining based on genetic network programming, *Proc. of the 2009 IEEE International Conference on Systems, Man, and Cybernetics*, pp.60-67, 2009.

[21] T. Őzyer, R. Alhajj and K. Barker, Intrusion detection by integrating boosting genetic fuzzy classifier and data mining criteria for rule pre-screening, *Journal of Network and Computer Applications*, vol.30, no.1, pp.99-113, 2007.

[22] J. T. Yao, S. L. Zhao and L. V. Saxton, A study on fuzzy intrusion detection, *Proc. of SPIE, Data Mining, Intrusion Detection, Information Assurance and Data Network Security*, pp.23-30, FL, USA, 2005.

[23] B. H. Sumida, A. I. Houston, J. M. McNamara and W. D. Hamilton, Genetic algorithms and evolution, *Journal of Theoretical Biology*, vol.147, pp.59-84, 1990.

[24] J. Kennedy and R. Eberhart, Particle swarm optimization, *Proc. of the IEEE International Conference in Neural Networks*, vol.4, pp.1942-1948, 1995.

[25] A. Colorni, M. Dorigo and V. Maniezzo, Distributed optimization by ant colonies, *Proc. of the European Conference on Artificial Life*, pp.134-142, 1991.

[26] Z. Banković, D. Stepanović, S. Bojanić and O. Nieto-Taladriz, Improving network security using genetic algorithm approach, *Computers & Electrical Engineering*, vol.33, no.5-6, pp.438-451, 2007.

[27] B. Balajinath and S. V. Raghavan, Intrusion detection through learning behavior model, *Computer Communications*, vol.24, no.12, pp.1202-1212, 2001.

[28] H.-H. Gao, H.-H. Yang and X.-Y. Wang, Ant colony optimization based network intrusion feature selection and detection, *Proc. of International Conference in Machine Learning and Cybernetics*, vol.6, pp.3871-3875, 2005.

[29] Z. Chen and P. Qian, Application of PSO-RBF neural network in network intrusion detection, *Proc. of the 3rd International Symposium in Intelligent Information Technology Application*, pp.362-364, 2009.

[30] D. Karaboga, An idea based on honey bee swarm for numerical optimization, *Technical Report-TR06*, Erciyes University, 2005.

[31] B. Basturk and D. Karaboga, An artificial bee colony (ABC) algorithm for numeric function optimization, *Proc. of the IEEE Swarm Intelligence Symposium 2006*, Indianapolis, IN, USA, 2006.

[32] D. Karaboga and B. Basturk, A powerful and efficient algorithm for numerical function optimization: Articial bee colony (ABC) algorithm, *Journal of Global Optimization*, vol.39, pp.459-471, 2007.

[33] D. Karaboga and B. Basturk, On the performance of artificial bee colony (ABC) algorithm, *Applied Soft Computing*, pp.687-697, 2008.

[34] A. Singh, An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem, *Applied Soft Computing*, vol.9, pp.625-631, 2009.

[35] W. C. Yeh and T. J. Hsieh, Artificial bee colony algorithm-neural networks for S-system models of biochemical networks approximation, *Neural Computing & Applications*, vol.21, no.2, pp.365-375, 2010.

[36] A. Lazarevic, V. Kumar and J. Srivastava, Intrusion detection: A survey, *Managing Cyber Threats, Massive Computing 2005*, vol.5, pp.19-78, 2005.

[37] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, Elsvier, 2005.

[38] E. S. Correa, A. Freitas and C. Johnson, A new discrete particle swarm algorithm applied to attribute selection in a bioinformatics data set, *Proc. of the 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, Washington, USA, pp.35-42, 2006.

[39] G. J. McLachlan, K-A Do and C. Ambroise, *Analyzing Microarray Gene Expression Data*, John Wiley & Sons, 2004.

[40] A. Zainal, M. Maarof and S. Shamsuddin, Feature selection using rough-DPSO in anomaly intrusion detection, *Computational Science and Its Applications*, pp.512-524, 2007.

[41] B. Akay and D. Karaboga, Parameter tuning for the artificial bee colony algorithm, *LNCS*, vol.5796, pp.608-619, 2009.

[42] S. Parsa and S. Naree, A new semantic kernel function for online anomaly detection of software, *ETRI Journal*, vol.34, no.2, pp.288-291, 2012.