International Journal of Soft Computing 6 (5-6): 158-167, 2011
ISSN: 1816-9503

A Local Search Guided Differential Evolution Algorithm Based Fuzzy Classifier for Intrusion Detection in Computer Networks

T. Amalraj Victoire and M. Sakthivel
Department of Mechanical Engineering, Anna University of Technology, Coimbatore, India

Abstract: The security of networked computers plays a strategic role in modern computer systems. The most important reason is the difficulties in obtaining adequate attack data for the supervised classifiers to model the attack patterns and the data acquisition task is always time-consuming and greatly relies on the domain experts. The growing prevalence of network attacks is a well-known problem which can impact the availability, confidentiality and integrity of critical information for both individuals and enterprises. This task is so complicated because the determination of normal and abnormal behaviors in computer networks is hard as the boundaries cannot be well defined. One of the difficulties in such a prediction process is the generation of false alarms in many anomaly based intrusion detection systems. This study proposes a Local Search guided Differential Evolution (LSDE) search algorithm to generate fuzzy rules capable of detecting intrusive behaviors. In the presented algorithm the global population is divided into subpopulations, each assigned to a distinct processor. Each subpopulation consists of the same class fuzzy rules. These rules evolve independently in the proposed parallel manner. A series of experimental results on the well-known KDD Cup 1999 data set demonstrate that the proposed method is more robust and effective than the state-of-the-art previous intrusion detection methods as well as can be further optimized as discussed in this study for real applications of intrusion detection system.

Key words: Differential evolution, local search intrusion detection, KDD cup data set, optimized, fuzzy rules, network

INTRODUCTION

An Intrusion Detection System (IDS) monitors and restricts user access to the computer system by applying certain rules. The rules are based on expert knowledge extracted from skilled administrators who construct attack scenarios and apply them to find system exploits. An intrusion is defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource (Heady *et al*., 1990). The system identifies all intrusions by users and takes or recommends necessary action to stop an attack on the database.

The problem of intrusion detection has been studied extensively in computer security (Axelsson, 2000; Idris and Shanmugam, 2005; Murali and Rao, 2005; Ye *et al*., 2003) and has received a lot of attention in machine learning and data mining (Cho, 2002; Cho and Cha, 2004; Tian *et al*., 2005). As discussed in Idris and Shanmugam (2005), misuse detection and anomaly detection are two approaches of the intrusion detection system (Murali and Rao, 2005). Misuse detection is the ability to identify intrusions based on a known pattern for the malicious activity. Manufacturers of IDS (using misuse detection) ensure that users have the most up to date patterns also known as signatures by sending them updates to protect against new vulnerabilities (Gao *et al*., 2005; Ozyer *et al*., 2007; Abadeh *et al*., 2007b).

In anomaly detection, the system administrator defines the baseline or normal state of the network's traffic load, breakdown, protocol and typical packet size. The anomaly detector monitors network segments to compare their state to the normal baseline and looks for anomalies (Ahmed and Traore, 2005; Feng *et al*., 2005; Kruegel and Vigna, 2003). Misuse detection is an effective approach to handle attacks that are known by the system. However, it is of the main difficulties is the high production of false alarms unpredictable in a dynamic system to know different various attacks. Besides, such an attempt would be time consuming and too complex. Anomaly detection is the complementary to the misuse detection. One disadvantage of anomaly detection is the likelihood of false alarms raised by the system.

Fuzzy systems based on fuzzy if-rules have been successfully used in many applications areas (Lee, 1990;

Corresponding Author: T. Amalraj Victoire, Department of Mechanical Engineering, Anna University of Technology, Coimbatore, India

Sugeno, 1985). Fuzzy if then rules were traditionally gained from human experts. Recently, various methods have been suggested for automatically generating and adjusting fuzzy if then rules without using the aid of human experts. Wangm and Mendel (1992), Ishibuchi *et al.* (1992), Abe and Lan (1995) and Mitra and Pal (1994).

Genetic algorithms have been used as rule generation and optimization tools in the design of fuzzy rule-based systems (Cordon *et al.*, 2004; Hu *et al.*, 2003). Those GA-based studies on the design of fuzzy rule-based systems are usually referred to as fuzzy genetics-based machine learning methods (fuzzy GBML methods), each of which can be classified into the Michigan, Pittsburgh or Iterative Rule Learning (IRL) approaches (Cordon *et al.*, 2004). Many fuzzy GMBL methods are categorized as the Pittsburgh approach (Lee *et al.*, 1998) where a set of fuzzy if-then rules is coded as an individual (Cervantes *et al.*, 2005; Ishibuchi and Yamamoto, 2004; Ishibuchi *et al.*, 2001; Rouwhorst and Engelbrecht, 2000). Some studies are categorized as the Michigan approach where a single fuzzy if-then rule is coded as an individual (Cervantes *et al.*, 2005; De Falco *et al.*, 2002; Ishibuchi *et al.*, 2005; Abadeh *et al.*, 2007a; Tan *et al.*, 2003). In the third approach, the iterative one, chromosomes code individual rules and a new rule is adapted and added to the rule set in an iterative fashion in every run of the GA (Hofmann, 2004; Ozyer *et al.*, 2007).

The proposed algorithm is a new parallel model based on the Michigan approach. In this algorithm, the global population is divided into some subpopulations, each assigned to a distinct processor. The number of subpopulations is equal to the number of classes in the classification problem. For each subpopulation, all of the individuals represent rules with the same class in the prediction part of the rule according to the fact that each individual in the presented algorithm represents a rule which has the form ˙if condition then prediction.

RELATED WORKS

Soft computing techniques are being widely used by the IDS researchers due to their generalization capabilities that help to detect known and unknown intrusions or the attacks that their patterns are not described before. Earlier studies have used a rule-based technique for intrusion detection but they were unable to detect new attacks or attacks that had no previously describe patterns (Anderson *et al.*, 1995; Ilgun, 1993; Lunt *et al.*, 1992). Some recent researches have utilized Artificial Immune Systems (AIS) to detect intrusive behaviors in a computer network (Dasgupta and Gonzalez, 2002; Harmer *et al.*, 2002; Yang *et al.*, 2002; Provost *et al.*, 1998). Some other applied techniques on intrusion detection problem are genetic algorithms (Ozyer *et al.*, 2007; Abadeh *et al.*, 2007b), Bayesian parameter estimation (Cho and Cha, 2004) and clustering (Xu and Zhang, 2005; Oh and Lee, 2003; Leon *et al.*, 2004; Guan *et al.*, 2003).

The overview of related works indicates that pattern recognition techniques are suitable to provide a solution to some open issues in IDS development. In addition, the extensive evaluation of pattern classification techniques accomplished on a sample dataset of network traffic performed during the KDD_99 conference indicated the feasibility of the pattern recognition approach to detect intruders (KDD-cup dataset). However, it should be stated that for the deployment of IDSs using pattern recognition algorithms in operational environments, one of the main diffculties is the high production of false alarms. Except for the neural network hierarchy proposed in Lee and Heinbuch (2001) where features at different abstraction levels were processed by distinct networks, the classification is usually performed in the feature space composed of all the features needed to detect the considered attack classes.

PROPOSED ALGORITHM

This study presents a local search guided differential evolution search algorithm which the study proposes in three subsections. The sub-section that follows, details the fuzzy if-then rules and a fuzzy reasoning method for pattern classification problems with continuous attributes. A similar heuristic procedure of Mohammad is also described to determine the consequent class and the certainty grade of each fuzzy if-then rule from training patterns. Subsequent section discusses the learning framework of the fuzzy-rule generation algorithm. The local search guided differential evolution search algorithm is presented next.

Fuzzy rule base for pattern classification: Let us assume that the pattern classification problem is a c-class problem in the n-dimensional pattern space with continuous attributes. Researchers also assume that m real vectors $x_p = (x_{p1}, x_{p2}, ..., x_{pn})$, p = 1, 2, ..., m are given as training patterns from the c classes (c<<m). Because the pattern space is $[0, 1]^n$, attribute values of each pattern are $x_{pi} \in [0, 1]$ for p = 1, 2, ..., m and i = l, 2, ..., n. In computer simulations all attribute values of each data set into the unit interval [0, 1] normalized.

In the presented fuzzy classifier system, we use fuzzy if then rules of the following form.

Rule R_j: If x_1 is A_{j1} and $_y$ and x_n is A_{jn}, then Class C_j with CF- CF_j where R_j is the label of the jth fuzzy if-then rule, A_{j1}, $_y$, A_{jn} are antecedent fuzzy sets on the unit interval [0, 1], C_j is the consequent class (i.e., one of the given c classes) and Cf_j is the grade of certainty of the fuzzy if-then rule R_j. The membership function of each linguistic value is specified by homogeneously partitioning the domain of each attribute into symmetric triangular fuzzy sets. Reseachers use such a simple specification in computer simulations to show the high performance of our fuzzy classifier system, even if the membership function of each antecedent fuzzy set is not tailored. However, we can use any tailored membership functions in the fuzzy classifier system for a particular pattern classification problem.

The total number of fuzzy if then rules is 5^n in the case of the n-dimensional pattern classification problem. It is impossible to use all the 5^n fuzzy if-then rules in a single fuzzy rule base when the number of attributes (i.e., n) is large (e.g., intrusion detection problem which n = 41). The fuzzy classifier system searches for a relatively small number of fuzzy if then rules (e.g., 100 rules) with high classification ability. Since, the consequent class and the certainty grade of each fuzzy if then rule can be determined from training patterns by a simple heuristic procedure (Ishibuchi *et al*., 1992, 1999; Nozaki *et al*., 1996), the task of the fuzzy classifier system is to generate combinations of antecedent fuzzy sets for a set of fuzzy if then rules. While this task seems to be simple at first glance in fact it is very difficult for high-dimensional pattern classification problems since, the search space involves 5^n combinations (e.g., >10 billion in the case of n = 13). In the fuzzy classifier system, the consequent Class C_j and the grade of certainty CF_j of each fuzzy if then rule are determined by a modified version of the heuristic procedure which is discussed in Ishibuchi *et al*. (1999).

Determination of C_j and CF_j

Step 1: Calculate the compatibility of each training pattern $x_p = (x_p1, x_{p2}, ..., x_{pn})$ with the fuzzy if-then rule R_j by the following product operation:

$$\mu_{j(x_p)} = \mu_{j1}(x_{p1}) \times ..... \times \mu_{jn}(x_{pn}), \quad p = 1, 2, ..., m \qquad (1)$$

Where, $\mu_j(x_{pi})$ is the membership function of A_{ji}.

Step 2: For each class, calculate the relative sum of the compatibility grades of the training patterns with the fuzzy if-then rule R_j:

$$\beta_{Class\ h}(R_j) = \sum_{x_p \in Class\ h} \mu_{j(x_p)} / N_{Class\ h}, \quad h = 1, 2, ..., c \qquad (2)$$

where, $\beta_{Class\ h}(R_j)$ is the sum of the compatibility grades of the training patterns in Class h with the fuzzy if then rule R_j and $N_{class\ h}$ is the number of training patterns which their corresponding class is Class h. The described modification of the heuristic procedure has occurred in this step since, in the procedure discussed by Ishibuchi *et al*. (1999) the sum of the compatibility grades is calculated instead of calculating the relative sum of the grades. This is because in intrusion detection problem some of the classes are very similar to each other. Moreover, the number of training patterns for each of the classes is significantly different. So if we use the traditional heuristic method of Ishibuchi *et al*. (1999), the consequent class of R_j might be specified incorrectly.

Step 3: Find Class h_j that has the maximum value of $\beta_{Class\ h}(R_j)$:

$$\beta_{Class\ \hat{h}_j}(R_j) = \max\{\beta_{Class_1}(R_j), ..., \beta_{Class_c}(R_j)\} \qquad (3)$$

If two or more classes take the maximum value, the consequent Class C_j of the fuzzy if-then rule R_j cannot be determined uniquely. In this case, let C_j be φ. If a single class takes the maximum value, let C_j be Class h_j. If there is no training pattern compatible with the fuzzy if-then rule R_j (i.e., if $P_{Classh}(R_j) = 0$ for h = 1, 2, ..., c) the consequent Class C_j is also specified as φ.

Step 4: If the consequent Class C_j is φ, let the grade of certainty Cf_j of the fuzzy if-then rule R_j be $Cf_j = 0$. Otherwise, the grade of certainty Cf_j is determined as follows:

$$CF_j = \left(\beta_{Class\ \hat{h}_j}(R_j) - \bar{\beta}\right) / \sum_{h=1}^{c} \beta_{Class\ h}(R_j) \qquad (4)$$

Where:

$$\bar{B} = \sum_{h \neq \hat{h}_j} \beta_{Class\ h}(R_j) / (c - 1) \qquad (5)$$

By the proposed heuristic procedure we can specify the consequent class and the certainty grade for any combination of antecedent fuzzy sets. Such a combination is generated by a fuzzy classifier system which its construction steps will be explained in the next subsections. The task of the fuzzy classifier system is to generate combinations of antecedent fuzzy sets for generating a rule set S with high classification ability. When a rule set S is given, an input pattern x_p-(x_{p1}, x_{p2},..., x_{pn}) is classified by a single winner rule R_j in S which is determined as follows (Ishibuchi *et al*., 1999):

$$\mu_j(x_p)CF_j = \max\left\{\mu_j(x_p)CF_j \mid R_j \in S\right\} \quad (6)$$

That is the winner rule has the maximum product of the compatibility and the certainty grade CF_j. The classification is rejected if no fuzzy if then rule is compatible with the input pattern x_p (i.e., $\mu_j(x_p) = 0$ for $\forall R_j \in S$).

Parallel learning framework: The parallel learning framework, indicates that the whole population is divided into c subpopulations which c is the number of classes in the classification problem. Then each of these subpopulations presents to a distinct processor and evolves using a heuristic LSDE algorithm. The training dataset for each of these subpopulations is different from others. The main training data set is divided into c sub datasets. Each of these sub datasets consists of the same class patterns. The result of this evolution which is a fuzzy rule set is then act as a source knowledge for a classifier. This classifier is capable of classifying patterns which have the class similar to their corresponding rule set class.

The main classifier consists of c classifiers which each of them develops using the above parallel learning framework separately. The following subsection will present the LSDE algorithm which is used to evolve each of the subpopulations.

Local Search guided Differential Evolution (LSDE) algorithm: First let us explain about the method of coding fuzzy rules. Each fuzzy if then rule is coded as a string. The following symbols are used for denoting the five linguistic values:

- 1 = Small
- 2 = Medium small
- 3 = Medium
- 4 = Medium large
- 5 = Large

For example, the following fuzzy if-then rules is coded as 1342: If x_1 is small and x_2 is medium and x_3 is medium large and x_4 is medium small then Class c_j with CF = CF_j.

DIFFERENTIAL EVOLUTION

Differential Evolution (DE) is one of the evolutionary algorithms. Every specimen from jth generation creates just one descent which is a member of the j+1 generation. Differential evolution can be used also for exactly analytical defined functions and localization of their global extremes. Testing the performance of the evolutionary algorithms is performed on simple functions on functions with very indistinctive extremums as well as on functions where the difference between global and local extremums is very small.

Population structure: The current population, symbolized by P_c is composed of those D-dimensional vectors:

$$X_i^g = \left\{x_{i,1}^g, x_{i,2}^g, \ldots, x_{i,D}^g\right\}$$

the index g indicates the generation to which a vector belongs. In addition, each vector is assigned a population index, i which varies from 1 to N_P and N_P is the population size.

Initialization: This stage consists in forming the initial population. The optimization of the membership functions, the initialization step consists in arbitrarily choosing the interval of this function.

Mutation: For each vector (for example, a vector which represents the interval of the membership functions):

$$V_i^g = \left\{v_{i,1}^g, v_{i,2}^g, \ldots, v_{i,D}^g\right\}$$

A mutant vector is as in following formulation:

$$v_{i,j}^g = x_{r1,j}^{g-1} + F * rand_{i,j}^g(x_{r1,j}^{g-1} - x_{r2,j}^{g-1}) \quad (7)$$

The scale factor F is a positive real number that controls the rate at which the population evolves. While there is no upper limit on F, effective values seldom are >1 is a random number between 0 and 1.

Crossing: The relative vector is mixed with the transferred vector to produce a test vector T_i^g:

$$T_{i,j}^g = \begin{cases} v_{i,j}^g \text{ if } (r_{i,j}^g \le Cr \text{ or } j = j_r \\ x_{i,j}^{g-1} \text{ otherwise} \end{cases} \quad (8)$$

The crossover probability $Cr \in [0, 1]$ is a user-defined value that controls the fraction of parameter values that are copied from the mutant. To determine which source contributes, a given uniform crossover parameter compares Cr to the output of a uniform random number generator $r_{i,j}^g$. If the random number is less than or equal to Cr, the trial parameter is inherited from the mutant V_i^g; otherwise, the parameter is copied from the vector x_i^g. In

addition, the trial parameter with randomly chosen index j_r is taken from the mutant to ensure that the trial vector does not duplicate x_i^g. Because of this additional demand, Cr only approximates the true probability.

Selection: All the solutions in the population have the same chance that the parents of being selected, regardless of their fitness function value f_t. The child produced (new vector) after the crossing operations is evaluated. Then, the performances of the child vector and its relative are compared and the best one is selected. If the relative is still better, it is maintained within the population.

With the aim of using this technique to optimize the membership functions of the explanatory factors, related to the dynamics of the prices (input) and the dynamics of the prices itself (output), researchers must represent the optimization problem in the form of vectors. We face a coding problem of the membership functions.

Local search: The local search operator searches the local area of an input individual to find better individuals. However, this task is done on individuals that their fitness is higher than a pre-specified threshold. This search is done by increasing or decreasing an antecedent part of the rule. Of course, those searches which produce a rule with a different consequent class (comparing to the class of the rule before performing the local search operation) must be repeated. This search must also be repeated if the fitness of the rule decreases after doing the local search operation. Therefore, this search performs the following steps:

- The local search operation is done only for those individuals that their fitness is higher than a threshold. This threshold is determined as follows:

$$T_{lt} = P_{lt} F_{max}$$

- Where, F_{max} is a number which indicates the maximum possible fitness for the input individual and P_{lt} is a percentage for the threshold. By decreasing P_{lt}, the local search operation performs for more individuals (line 1)
- Increase or decrease a random if-part of the input fuzzy rule. If the result is 0 or c+1, change it to 2 or c- 1, respectively (lines 2-8)

The result rule is accepted only if its consequent class is the same as the class of the input rule and its fitness is better than the fitness of the input rule otherwise the result rule is rejected. If the result rule is rejected the local search operation repeats until K iterations (lines 9-11).

The output of this procedure would be a rule which is very similar to the input rule while its fitness is improved. After performing the above local search procedure for all of the individuals in the subpopulation, the fitness value of each of them is evaluated.

Replacement: A pre-specified number of fuzzy if then rules in the current population are replaced with the newly generated:

```
begin
1    if (individual.fitness> = Tlt) then
2    for j = 1 to K
3    n1 = random_choose(-1, 1);
4    n2 = random_choose(1.. 41);
5    temp = individual.if part (n2);
6    individual.if part (n2) = individual.if part (n2) +n1;
7    if (individual.if part (n2) = 0) then individual.if part (n2) = 2;
8    if (individual.ifpart (n2) = C+1) then individual.if part (n2) = C- 1;
9    if (Detclass(individual) <>individual.thenpart) or
     (Detfitness(individual) <individual.fitness) then individual.if part (n2)
     = temp; j = j-1
10   end-if
11   end-for j
12   end-if
13   end
```

rules. In the fuzzy classifier system, P_{rep} percent of the worst rules with the smallest fitness values are removed from the current population and (100-P_{rep}) percent of the newly generated fuzzy if-then rules are added (P_{rep} is the replacement percentage).

After performing the above replacement procedure the fitness value of each of the individuals is evaluated according to Eq. 7.

Re-initialization: In oLSDEr to improve the classification rate of the current subpopulation, researchers replace some of the weak fuzzy if then rules of the current subpopulation with some new fuzzy if then rules which are generated according to rejected training patterns. The number of inserted new rules depends on the number of weak individuals in the current subpopulation. We call an individual weak if its fitness is less than a threshold. This threshold is defined as follows:

$$T_{weak} = P_{weak} F_{max} \quad (9)$$

where, P_{weak} is a percentage for the threshold. By decreasing P_{weak}, the reinitialization operation performs for more individuals or the weakness definition matches for more individuals. The task of generating a new fuzzy if-then rule is similar to the one which is explained in the initialization step. After performing the above reinitialization procedure the fitness value of each of the individuals is evaluated according to Eq. 7.

Internal termination test: The total number of generations is taken as a stopping condition for terminating the internal cycle of the fuzzy classifier system.

Saving the learned fuzzy rule set: The whole subpopulation is stored as a rule set in the Fuzzy Rule Set Pool (FRSP). This pool is updated until a pre-specified number of iteration for the external cycle of the genetic local search algorithm is achieved.

External termination test: As most of the operations in the mentioned genetic local search algorithm deal with random parameters, the algorithm repeats the previous steps until a pre-specified number of cycles. This external cycle enables the genetic local search algorithm to obtain high-quality fuzzy sets. The detailed pseudo-code for LSDE based fuzzy classifier is as follows:

stepl: Generate an initial population of N_p individuals
for i = 1 to N_p
for j = 0-l
for k = 1 to D
generate a random value r_x in the range [$x_{j,min}$, $x_{j,max}$]; // $x_{j,min}$ and $x_{j,max}$ indicate respectively the maximal and the minimal value of the input j
$x^0_{i,j,k} = r_x$;
end for
end for
Adjust x^0_i to ensure that $x^0_{i,j} \geq x^0_{i,j-1}$ for each] = 1 to D;
for j = l to I
for k = l to D
generate a random integer value r_y from 1 to D;
$y^0_{i,j,k} = r_y$
End for
End for
Adjust Y^0_i to ensure that $Y^0_{i,j} \geq Y^0_{i,j-1}$ or $Y^0_{i,j} \leq Y^0_{i,j-1}$ for each j = 1 to D;
Evaluate the new solution using RMSE (X^0_i, Y^0_i) ; //Evaluate each individual based on the numerical output of the fuzzy
end for // system using RMSE (the root-mean squared error)

step 2: apply Differential Evolution Algorithm
Let g = 0;
Repeat
g = g+1;
for i = 0 to N_p
for j = 0 to I
select at randomly rl≠r2≠r3 from 1 to N_p and j^1_r and j^2_r from 1 to D;
for k = l to D

$v^g_{X_i,j,k} = x^{g-1}_{r1,j,k} + F * rand^g_{Xj,k}(x^{g-1}_{r1,j,k} - x^{g-1}_{r2,j,k})$; \\

$v^g_{X_i,j}, T^{g-1}_{X_i,j}$, indicate the mutated vector and the crossed vector

if $(r^g_{X_i,i,j} \leq Cr$ or $j = j^1_r)\ T^g_{X_i,i,j} = v^g_{Xi,j,k}$; \\

of membership functions of the input j

Else $T^g_{X_i,ij} = x^{g-1}_{i,j}$

end for
end for
Adjust $T^g_{X_i,i}$ to ensure that $T^g_{X_i,i,j} \geq T^g_{X_i,i,j-1}$ for each j = 1 to D;

for j = l to I
select at randomly rl≠r2≠r3 from 1 to N_p and j^1_r and j^2_r) from 1 to D;

for k = l to D

$v^g_{Y_i,j,k} = x^{g-1}_{r1,j,k} + F * rand^g_{Yj,k}(y^{g-1}_{r1,j,k} - y^{g-1}_{r2,j,k})$ \\

$V^g_{Y_i,j}, T^g_{Y_i,j}$, indicate the mutated vector

and the crossed vector
round down; $v^g_{Y_i,j,k}$; \\ of rules of the input j
if $(r^g_{Y_i,j,} \leq Cr$ or $j = j^2r)T^g_{Y_i,j,} = v^g_{Y_i,j,}$;

else $T^g_{Y_i,i,j} = y^{g-1}_{i,j}$

end for
end for
Adjust $T^g_{Y_i,i}$ to ensure that $T^g_{Y_i,i,j} \geq T^g_{Y_i,i,j-1}$ or $T^g_{Y_i,i,j} \leq T^g_{Y_i,i,j-1}$

for each j = 1 to D;
Evaluate the new solution using RMSE $(T^g_{X_i,i,j-1}, T^g_{Y_i,i,j-1})$;

If (RMSE $(T^g_{X_i,i}, T^g_{Y_i,i})$ <RMSE (X^{g-1}_i, Y^{g-1}_i)) then

$X^g_i = T^g_{X_i,i}$ and $Y^g_i = T^g_{Y_i,i}$;

else
$X^g_i = X^{g-1}_i$ and $Y^g_i = T^{g-1}_i$;

End if
End for
until (a stopping criterion is found)

Data sets for validation of the LSDE based fuzzy classifier: To evaluate the performance of the proposed approach, a series of experiments on KDD CUP 1999 dataset were conducted.

In the experiments, KDD CUP 1999 dataset is used. The KDD CUP 1999 dataset is a version of the original 1998 DARPA intrusion detection evaluation program which is prepared and managed by the MIT Lincoln Laboratory.

The dataset contains about 5 million connection records as training data and about 2 million connection records as test data. And the dataset includes a set of 41 features derived from each connection and a label which specifies the status of connection records as either normal or specific attack type.

These features have all forms of continuous, discrete and symbolic variables with significantly varying ranges falling in four categories: the first category consists of the intrinsic features of a connection which include the basic features of individual TCP connections.

The duration of the connection, the type of the protocol (TCP, UDP, etc.) and network service (http, telnet, etc.) are some of the features. The content features within a connection suggested by domain knowledge are

Table 1: Classes in the 10% of the KDD-Cup 99 data set

Classes	Sub-classes	Samples
Normal		97.278 (19.6911%)
PRB	Ipsweep, nmap, portsweep, satan	4107 (0.8313%)
DOS	Back, land, neptune, pod, smurf, teardrop	391.458 (79.2391%)
U2R	Buffer_overflow, loadmodule, multihop, perl, rootkit	52 (0.0105%)
R2L	Ftp, write, guess_passwd, imap, phf, spy, warezclient, warezmaster	1126 (0.2279%)

used to assess the payload of the original TCP packets such as the number of failed login attempts. The same host features examine established connections in the past two seconds that have the same destination host as the current connection and calculate the statistics related to the protocol behavior, service, etc. The similar same service features inspect the connections in the past two seconds that have the same service as the current connection. Likewise, attacks fall into four categories: Denial of Service (DoS): making some computing or memory resources too busy to accept legitimate users access these resources. Probe (PRB): host and port scans to gather information or find known vulnerabilities. Remote to Local (R2L): unauthorized access from a remote machine in oLSDEr to exploit machine's vulnerabilities. User to Root (U2R): unauthorized access to local super user (root) privileges using system's susceptibility. Random selection has been used in many applications to reduce the size of the dataset.

In this study, researchers randomly select 18,285 records, similar to prior research. The PRB, R2L and U2R attack classes were totally selected because of their low portion in the KDD dataset. Three-thousand normal connections (records) and 10,000 DoS connections were randomly selected. For the testing step, the KDD testing set was used. Table 1 shows detailed information about the number of all records. It is important to note that the test data includes specific attack types not present in the training data. This makes the intrusion detection task more realistic.

EXPERIMENTAL RESULT

Experiments were carried out on a subset of the database created by DARPA in the framework of the 1998 Intrusion Detection Evaluation Program (Lincoln Laboratory MIT). Researchers used the subset that was pre-processed by the Columbia University and distributed as part of the UCI KDD Archive (KDD-Cup data set). The available database is made up of a large number of network connections related to normal and malicious traffic. Each connection is represented with a 41-dimensional feature vector. Connections are also labeled as belonging to one out of five classes. One of these

Table 2: Feature set of the preprocessed KDD-Cup 99 data set. Feature type: S for symbolic feature and C for contineous

Feature name	Type
Duration	C
protocol_type	S
Service	S
Flag	S
src_byte	C
dst_bytes	C
Land	C
wrong_fragment	C
Urgent	C
Hot	C
num_failed_logins	C
logged_in	S
num_compromised	C
root_shell	C
su_attempted	C
num_root	C
num_file_creations	C
num_shells	C
num_access_files	C
num_outbounds_cmds	C
is_host_login	C
is_guest_login	S
Count	C
srv_count	C
serror_rate	C
srv_error_rate	C
rerror_rate	C
srv_rerror_rate	C
same_srv_rate	C
diff_srv_rate	C
srv_diff_host_rate	C
dst_host_count	C
dst_host_srv_count	C
dst_host_same_srv_rate	C
dst_host_diff_srv_rate	C
dst_host_same_src_port_rate	C
dst_host_srv_diff_host_rate	C
dst_host_serror_rate	C
dst_host_srv_serror_rate	C
dst_host_rerror_rate	C
dst_host_srv_rerror_rate	C
	C

classes is the normal class and the rest indicates four different intrusion classes. These intrusion classes are a classification of 22 different types of attacks in a computer network. Table 1 shows the classification of different types of attacks in the four different intrusion classes and the distribution of each class in the 10% of the KDD-Cup 99 data set.

As it is shown in Table 1 the number of records in the 10% data set is very large (494, 021). Also, the proportion of samples per class is not uniform, for example from class U2R the number of samples in the training data set is 52 while from class DOS the number of samples is 391,458. According to this fact researchers have used a subset of this large dataset as the train and test datasets hence, the training data set contains 650 randomly generated samples. A different randomly selected set of 6500 instances is used for testing different genetic fuzzy systems. Researchers normalized the train and test data sets where each numerical value in the data set is normalized between 0 and 1. Table 2 shows features of

Table 3: Distribution of different classes in the train and test datasets

Algorithm	Train	Test
Normal	100	1000
U2R	50	59
R2L	100	1000
DOS	300	6500
PRB	100	1000

Table 4: Parameter specification in computer simulations for the Michigan approach based IDS

Parameters	Values
Populatin size (N_{pop})	50.00
Crossover probablity (P_c)	0.70
Mutation probability (P_m)	0.01
Mutation attempts (M_{repeat})	30.00
Replacement percentage (P_{repR})	20.00
Maximum number of generations	100.00

Table 5: Confusion matrix for the LSDE based IDS

	Detected class					
Real class	Normal	U2R	R2L	DOS	PRB	Recall (%)
Normal	990.0	7.0	1	0	1	99.2
U2R	26.0	26.0	7	0	0	44.0
R2L	206.0	0.0	794	0	0	79.4
DOS	459.0	0.0	0	6041	0	92.9
PRB	315.0	3.0	0	0	682	68.2
Precision (%)	49.6	72.2	99	100	100	-

the preprocessed KDD-Cup 99 data set. The distribution of different classes in the train and test datasets is shown in Table 3. This study consists of 2 subsections. First experiments of applying each of the three mentioned genetic fuzzy systems to the intrusion detection classification problem. In the next subsection researchers compare the performance of proposed LSDE fuzzy systems to some of other intrusion detection algorithms. Table 4 shows the parameter specifications that researchers have used in the computer simulations for each of the mentioned fuzzy systems.

Table 5 is the confusion matrix of the fuzzy systems. The top-left entry of Table 5 shows that 992 of the actual Normal test set were detected to be normal; the last column indicates that 99.2% of the actual NORMAL data points were detected correctly. In the same way, for PRB 682 of the actual Attack test set were correctly detected; the last column indicates that 68.2% of the actual PRB data points were detected correctly. The bottom row shows that 49.6% of the test set said to be Normal indeed were Normal and 100% of the test set classified as PRB indeed belongs to PRB. The results of other classes in Table 5.

As per the tabulations shown in Table 6 and 7, the precision rate of Michigan based LSDE fuzzy system for R2L, DOS and PRB patterns outperforms other approaches. Also it is clear that the IDS which develops using LSDE fuzzy systems would be more reliable than other approaches as the false alarm rate in GFS based intrusion detection systems is considerably lower than other approaches.

Table 6: Classification rate comparison of testing for five-class classifications

	Class				
Algorithm	Normal	U2R	R2L	DOS	PRB
LSDE	99.23	45.00	79.70	92.70	68.50
EFRID (Gomez and Dasgupta, 2001)	92.78	88.13	7.41	98.91	50.35
Winner entry (Elkan, 2000)	94.50	13.20	8.40	97.10	83.30

Table 7: Performance comparison

Algorithm	Detection rate	False alarm rate
LSDE	89.17	0.13
EFRID	98.15	7.00
RIPPER-Artificial Anomalies (Fan *et al*., 2005)	94.26	2.02
SMARTSIFTER (Yamanishi *et al*., 2000)	82.00	-

CONCLUSION

A Local Search guided Differential Evolution (LSDE) algorithm is introduced in this article to generate fuzzy rules which are then used to construct a fuzzy classifier system. In the proposed algorithm the global population is divided as subpopulations and then each of them is assigned to a distinct processor. Each subpopulation consists of the same class fuzzy rules. These subpopulations are evolved independently using a local search guided differential evolution algorithm. The performance of the final classification system which was constructed according to the proposed parallel learning algorithm was compared to several classification algorithms. Results showed that the presented algorithm was capable of increasing the detection rate and decreasing the false alarm rate simultaneously. The training time of the proposed learning algorithm is decreased significantly using the suggested parallel learning framework. This advantage could be used to construct high-performance classifiers capable of solving complex classification problems in a considerable short computation time.

REFERENCES

Abadeh, M.S., J. Habibi and C. Lucas, 2007a. Intrusion detection using a fuzzy genetics-based learning algorithm. J. Network Comput. Appl., 30: 414-428.

Abadeh, M.S., J. Habibi, Z. Barzegar and M. Sergi, 2007b. A parallel genetic local search algorithm for intrusion detection in computer networks. Eng. Appl. Artif. Intell., 20: 1058-1069.

Abe, S. and M.S. Lan, 1995. A method for fuzzy rules extraction directly from numerical data and its application to pattern classification. IEEE Trans. Fuzzy Syst., 3: 18-28.

Ahmed, A.A.E. and L. Traore, 2005. Anomaly intrusion detection based on biometrics. Proceedings of the 2005 IEEE Workshop on Information Assurance and Security United States Military Academy, August 15 2005, West Point, New York, pp: 452-453.

Anderson, D., T.F. Lunt, H. Javitz, A. Tamaru and A. Valdes, 1995. Detecting unusual program behavior using the statistical component of the next-generation intrusion detection expert system (NIDES). Proceeding of the SRI International, Menlo Park, May 1995, Computer Science Laboratory SRI-CSL-95-06, CA.

Axelsson, S., 2000. Intrusion detection systems: A survey and taxonomy. Technical report no. 99-15. Department of Computer Engineering, Chalmers University of Technology, Sweden.

Cervantes, A., I. Galvan and P. Isasi, 2005. A comparison between the Pittsburgh and Michigan approaches for the binary PSO algorithm. Proceeding of the 2005 IEEE Congress on Evolutionary Computation, September 5, 2005, Computer Science Department University Carlos III de Madrid, pp: 290-297.

Cho, S. and S. Cha, 2004. SAD: Web session anomaly detection based on parameter estimation. Comput. Security, 23: 312-319.

Cho, S.B., 2002. Incorporating soft computing techniques into a probabilistic intrusion detection system. IEEE Trans. Syst. Man Cybernetics Part C, 32: 154-160.

Cordon, O., F. Gomide, F. Herrera, F. Hofmann and L. Magdalena, 2004. Ten years of genetic fuzzy systems current framework and new trends. Fuzzy Sets Syst., 141: 5-31.

Dasgupta, D. and F. Gonzalez, 2002. An immunity based technique to characterize intrusions in computer networks. IEEE Trans. Evolut. Comput., 6: 281-291.

De Falco, I., A.D. Cioppa and E. Tarantino, 2002. Discovering interesting classification rules with genetic programming. Applied Soft Computing, 1: 257-269.

Feng, Y., Z.F. Wu, K.G. Wu, W., Z.Y. Xiong and Y. Zhou, 2005. An unsupervised anomaly intrusion detection algorithm based on swarm intelligence. Proceedings of the fourth international conference on machine learning and cybernetics, August 18-21, 2005, Guangzhou, pp: 3965-3969.

Gao, H.H., H.H. Yang and X.Y. Wang, 2005. Ant colony optimization based network intrusion feature selection and detection. Proc. Int. Conf. Machine Learn. Cybernetics, 6: 3871-3875.

Guan, Y., A.A. Ghorbani and N. Belacel, 2003. Y-MEANS: A clustering method for intrusion detection. Proc. Can. Conf. Electr. Comput. Eng., 2: 1083-1086.

Harmer, P.K., P.D. Williams, G.H. Gunsch and G.B. Lamont, 2002. An artificial immune system architecture for computer security applications. IEEE Trans. Evol. Comput., 6: 252-280.

Heady, R., G. Luger, A. Maccabe and M. Servilla, 1990. The architecture of network level intrusion detection system. Technical Report, Department of Computer Science, University of New Mexico.

Hofmann, F., 2004. Combining boosting and evolutionary algorithms for learning of fuzzy classification rules. Fuzzy Sets Syst., 141: 47-58.

Hu, Y.C., R.S. Chen and G.H. Tzeng, 2003. Finding fuzzy classification rules using data mining techniques. Pattern Recognit. Lett., 24: 509-519.

Idris, N.B. and B. Shanmugam, 2005. Artificial intelligence techniques applied to intrusion detection. Proceedings of Annual IEEE Indicon, December 11-13, 2005, Institute of Electrical and Electronics Engineers, pp:52-55.

Ilgun, K., 1993. USTAT: A real-time intrusion detection system for UNIX. Proceedings of the 1993 Computer Society Symposium on Research in Security and Privacy, Oakland, May 24-26, 1993, IEEE Computer Society Press, Los Alamitos, pp: 16-28.

Ishibuchi, H. and T. Yamamoto, 2004. Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. Fuzzy Sets Syst., 141: 59-88.

Ishibuchi, H., K. Nozaki and H. Tanaka, 1992. Distributed representation of fuzzy rules and its application to pattern classification. Fuzzy Sets Syst., 52: 21-32.

Ishibuchi, H., T. Nakashima and T. Murata, 1999. Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems. IEEE Trans. Syst. Man Cybern. B Cybern., 19: 601-618.

Ishibuchi, H., T. Nakashima and T. Murata, 2001. Three-objective genetics-based machine learning for linguistic rule extraction. Info. Sci., 136: 109-133.

Ishibuchi, H., T. Yamamoto and T. Nakashima, 2005. Hybridization of fuzzy GBML approaches for pattern classification problems. IEEE Trans. Syst. Man Cybern. B Cybern., 35: 359-365.

Kruegel, C. and G. Vigna, 2003. Anomaly detection of web-based attacks. Proceedings of 10th ACM Conference on Computer and Communications Security, (CCS'03), ACM Press, pp: 251-261.

Lee, C.C., 1990. Fuzzy logic in control systems: Fuzzy logic controller. II. IEEE Trans. Syst. Man Cybernet., 20: 419-435.

Lee, S.C. and D.V. Heinbuch, 2001. Training a neural-network based intrusion detector to recognize novel attacks. IEEE Trans. Syst. Man Cybern. A, 31: 294-299.

Lee, W., S.J. Stolfo and K.W. Mok, 1998. Mining audit data to build intrusion detection models. Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, August 27-31, 1998, AAAI Press, New York, pp: 1-20.

Leon, E., O. Nasraoui and J. Gomez, 2004. Anomaly detection based on unsupervised niche clustering with application to network intrusion detection. IEEE Conf. Evol. Comput., 1: 502-508.

Lunt, T., A. Tamaru, F. Gilham, R. Jagannathan and C. Jalali *et al*., 1992. A real time intrusion detection expert system (IDES). Final Report.SRI Interna-tional, Menlo Park, CA.

Mitra, S. and S.K. Pal, 1994. Self-organizing neural network as a fuzzy classifier. IEEE Trans. Syst. Man Cybernet., 24: 385-399.

Murali, A. and M. Rao, 2005. A Survey on intrusion detection approaches. Proceedings of the 1st International Conference on Information and Communication Technologies, August 27-28, 2005, Computer Centre University of Hyderabad India, pp: 233-240.

Oh, S.H. and W.S. Lee, 2003. An anomaly intrusion detection method by clustering normal user behavior. Comput. Secur., 22: 596-612.

Ozyer, T., R. Alhajj and K. Barker, 2007. Intrusion detection by integrating boosting genetic fuzzy classifier and data mining criteria for rule pre-screening. J. Network Comput. Appl., 30: 99-113.

Provost, F., T. Fawcett and R. Kohavi, 1998. The case against accuracy estimation for comparing induction algorithms. Proceedings of the 15th International Conference on Machine Learning, July 24-27, 1998, Morgan Kaufmann, pp: 445-453.

Rouwhorst, S.E. and A.P. Engelbrecht, 2000. Searching the forest: Using decision trees as building blocks for evolutionary search in classification databases. IEEE Congr. Evol. Comput., 1: 633-638.

Sugeno, M., 1985. An introductory survey of fuzzy control. Inform. Sci., 36: 59-83.

Tan, K.C., Q. Yu, C.M. Heng and T.H. Lee, 2003. Evolutionary computing for knowledge discovery in medical diagnosis. Artif. Intell. Med., 27: 129-154.

Tian, J.F., X. Ying, F. Yue and J.L. Wang, 2005. Intrusion detection combining multiple decision trees by fuzzy logic. Proceedings of Sixth International Conference on Parallel and Distributed Computing Applications and Technologies, December 5-8, 2005, IEEE Computer Security, pp: 256-258.

Wangm, L.X. and J.M. Mendel, 1992. Generating fuzzy rules by learning from examples. IEEE Trans. Syst. Man Cybern., 22: 1414-1427.

Xu, B. and A. Zhang, 2005. Application of support vector clustering algorithm to network intrusion detection. Proceedings of International Conference on Neural Networks and Brain, October 13-15, 2005, IEEE Computer Security, USA., pp: 1036-1040.

Yang, X.R., J.Y. Shen and R. Wang, 2002. Artificial immune theory based network intrusion detection system and the algorithms design. Int. Conf. Mach. Learn. Cybern. Beijing, 1: 73-77.

Ye, N., S. Vilbert and Q. Chen, 2003. Computer intrusion detection through EWMA for autocorrelated and uncorrelated data. IEEE Trans. Reliab., 52: 75-82.