A Refined Differential Evolution Algorithm Based Fuzzy Classifier for Intrusion Detection

T. Amalraj Victoire  
PhD Research Scholar, Anna University of Technology, Coimbatore  
E-mail: amalrajvictoire@gmail.com

M. Sakthivel  
Associate Professor & Head, Dept of Mechanical Engg  
Anna University of Technology, Coimbatore  
E-mail: sakthi_vel_m@yahoo.com

Abstract

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. 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. 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 paper proposes a refined differential evolution (RDE) 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 paper for real applications of intrusion detection system.

Keywords: Intrusion Detection, Differential Evolution, Local Search, KDD Cup data set

1. Introduction

An intrusion is defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource (Heady, Luger, Maccabe, & Servilla, 1990). 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. 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 & Shanmugam, 2005; Murali & Rao, 2005; Ye, Vilbert, & Chen, 2003, 2004), and has received a lot of attention in machine learning and data mining (Cho, 2002; Cho & 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 & 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, Yang, & Wang, 2005; Ozyer, Alhajj, & Barker, 2007; Saniee Abadeh, Habibi, & Lucas, 2007).

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 (Awad & Traore, 2005; Feng, Wu, Wu, Xiong, & Zhou, 2005; Kruegel & Vigna, 2003). Misuse detection is an effective approach to handle attacks that are known by the system. However, it is 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; 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, Nozaki, and Tanaka (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, Gomide, Herrera, Hofmann, & Magdalena, 2004; Hu, Chen, & Tzeng, 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 GMML methods are categorized as the Pittsburgh approach (Lee, Salvatore, & Mok, 1998) where a set of fuzzy if-then rules is coded as an individual (Cervantes, Galvan, & Isasi, 2005; Ishibuchi & Yamamoto, 2004; Ishibuchi, Nakashima, & Murata, 2001; Rouwhorst & 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, Cioppa, & Tarantino, 2002; Ishibuchi, Yamamoto, & Nakashima, 2005; Saniee Abadeh et al., 2007; Tan, Yu, Heng, & Lee, 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.’

We have empowered our algorithm using a heuristic local search procedure. This procedure aims to improve the quality of derived fuzzy rules by searching their neighborhoods according to some restrictions. Section 3 will discuss about the details of this procedure clearly.

This paper is organized as follows: a brief introduction to related work in the field of intrusion detection is given in Section 2. In Section 3, we propose our refined differential evolution search algorithm. Section 4 will discuss about the experimental results which we have obtained. The proposed approaches have been tested using the public KDD CUP’99 intrusion detection data set available at the University of California, Irvine web site (KDD-Cup data set).Section 5 derives some conclusions.
2. 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 and Fawcett, 1998). Some other applied techniques on intrusion detection problem are genetic algorithms (Ozyer et al., 2007; SaniedAbadeh et al., 2007), Bayesian parameter estimation (Cho and Cha, 2004) and clustering (Xu et al., 2005; Oh &Lee, 2003; Leon et al., 2004; Guan et al., 2003).

The above 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 difficulties 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.

3. Proposed Algorithm
This section presents a refined differential evolution search algorithm, which the paper 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 et al, 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 refined differential evolution search algorithm is presented next.

3.1. Fuzzy Rule Base for Pattern Classification
Let us assume that our pattern classification problem is a c-class problem in the n-dimensional pattern space with continuous attributes. We also assume that m real vectors \( x_p = (x_{p1}, x_{p2}, \ldots, x_{pn}) \), \( p = 1, 2, \ldots, m \), 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 \( p = 1, 2, \ldots, m \) and \( i = 1, 2, \ldots, n \). 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. 

\[ \text{Rule } R_j: \text{If } x_{j1} \text{ is } A_{j1}, \ldots, y \text{ and } x_{jn} \text{ is } A_{jn}, \text{ then Class } C_j \text{ with } CF_{j} = CF_{j1} \text{ where } R_j \text{ is the label of the } j\text{th fuzzy if-then rule, } A_{j1}, \ldots, A_{jn} \text{ are antecedent fuzzy sets on the unit interval } [0,1], \text{ and } C_j \text{ is the consequent class (i.e., one of the given } c \text{ classes), and } CF_j \text{ is the grade of certainty of the fuzzy if-then rule } R_j. \] 

The membership function of each linguistic value is specified by homogeneously partitioning the domain of each attribute into symmetric triangular fuzzy sets. We 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 our 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 \( n = 41 \)). Our 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; Nozaki et al., 1996; Ishibuchi and Nakashima, 1999), the task of our 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., more than ten billion in the case of \( n = 13 \)). In our 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 and Nakashima (1999).

### 3.1.1. Determination of \( C_j \) and \( CF_j \)

**Step 1:** Calculate the compatibility of each training pattern \( x_p = (x_{p1}, x_{p2}, \ldots, x_{pn}) \) with the fuzzy if-then rule \( R_j \) by the following product operation:

\[
\mu_j(x_p) = \mu_{j1}(x_{p1}) \times \cdots \times \mu_{jn}(x_{pn}) \quad p = 1, 2, \ldots, m
\]

where \( \mu_{j(x_p)} \) 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 \( R_j \):

\[
\beta_{\text{Class } h}(R_j) = \sum_{x_{p \in \text{Class } h}} \mu_j(x_p) / N_{\text{Class } h}, \quad h = 1, 2, \ldots, c,
\]

Where \( \beta_{\text{Class } h}(R_j) \) is the sum of the compatibility grades of the training patterns in Class \( h \) with the fuzzy if-then rule \( R_j \) and \( N_{\text{Class } h} \) 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 in Ishibuchi and Nakashima (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 and Nakashima (1999), the consequent class of \( R_j \) might be specified incorrectly.

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

\[
\beta_{\text{Class } h}(R_j) = \max \{ \beta_{\text{Class } 1}(R_j), \ldots, \beta_{\text{Class } c}(R_j) \}
\]

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 \( \varphi \). If a single class takes the maximum value, let \( C_j \) be Class \( h \). If there is no training pattern compatible with the fuzzy if-then rule \( R_j \) (i.e., if \( P_{\text{Class } h}(R_j) = 0 \) for \( h = 1, 2, \ldots, c \)) the consequent Class \( C_j \) is also specified as \( \varphi \).

**Step 4:** If the consequent Class \( C_j \) is \( \varphi \), 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:

\[
CF_j = \left( \beta_{\text{Class } h}(R_j) - \bar{B} \right) / \sum_{h \neq h_j} \beta_{\text{Class } h}(R_j)
\]

Where

\[
\bar{B} = \sum_{h \neq h_j} \beta_{\text{Class } h}(R_j) / (c - 1)
\]

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 our 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 \( x_p = (x_{p1}, x_{p2}, \ldots, x_{pm}) \) is classified by a single winner rule \( R_j \in S \), which is determined as follows (Ishibuchi and Nakashima, 1999):

\[
\mu_j(x_p) \cdot CF_j = \max \{ \mu_j(x_p) \cdot CF_j | R_j \in S \}
\]

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 \( \forall R_j \in S \)).

3.2. 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 RDE 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 RDE algorithm, which is used to evolve each of the subpopulations.

3.3. Refined Differential Evolution (RDE) 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”:

\[ \text{If } x_1 \text{ is small and } x_2 \text{ is medium and } x_3 \text{ is medium large and } x_4 \text{ is medium small, then Class } c_j \text{ with } CF = CF_j. \]

4. 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.

4.1. Population Structure

The current population, symbolized by \( P_c \), is composed of those D-dimensional vectors \( X_i^g = \{x_{i,1}^g, x_{i,2}^g, \ldots, x_{i,D}^g\} \) 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.

4.2. 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.
4.3. Mutation

For each vector (for example, a vector which represents the interval of the membership functions) $V_i^g = \{v_{i,1}^g, v_{i,2}^g, ..., v_{i,D}^g\}$, a mutant vector is as in following formulation:

$$v_{i,j}^g = x_{i,j}^{g-1} + F \times rand_{i,j} \times (x_{i,j}^{g-1} - x_{i,j}^{g-2})$$

(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 greater than "1"; $rand_{i,j}$ is a random number between 0 and 1.

4.4. Crossing

The relative vector is mixed with the transferred vector to produce a test vector $T_{i,j}^g$:

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

(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 $v_{i,j}^g$; otherwise, the parameter is copied from the vector $x_{i,j}^g$. 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 $x_{i,j}^g$. Because of this additional demand, $Cr$ only approximates the true probability.

4.5. 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), we must represent our optimization problem in the form of vectors. We face a coding problem of the membership functions.

4.6. 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:

$$T_{1r} = P_{1r} F_{\text{max}}$$
Where $F_{\text{max}}$ 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 rules. In our fuzzy classifier system, $P_{\text{rep}}$ percent of the worst rules with the smallest fitness values are removed from the current population and $(100 - P_{\text{rep}})$ percent of the newly generated fuzzy if-then rules are added ($P_{\text{rep}}$ is the replacement percentage).

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

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

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

**4.7. Re-initialization**

In order to improve the classification rate of the current subpopulation, we 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:

$$T_{\text{weak}} = P_{\text{weak}} F_{\text{max}},$$

where $P_{\text{weak}}$ is a percentage for the threshold. By decreasing $P_{\text{weak}}$, 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 our fuzzy classifier system.

### 4.8. 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 RDE based fuzzy classifier is as follows,

**Step 1:** Generate an initial population of $N_p$ individuals

```
for i = 1 to $N_p$
    for j = 0 to 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
            $x_{i,j,k}^0 = r_x$; // and the minimal value of the input $j$
        end for
    end
end
```

Adjust $s^0_i$ to ensure that $x_{i,j}^0 = x_{i,j-1}^0$ for each $j = 1$ to $D$;

```
for j = 1 to l
    for k = 1 to D
        generate a random integer value $r_y$ from 1 to D;
        $y_{i,j,k}^0 = r_y$
    end for
end
```

Adjust $r^0_i$ to ensure that $Y_{i,j}^0 \geq Y_{i,j-1}^0$ or $Y_{i,j}^0 \leq Y_{i,j-1}^0$ for each $j = 1$ to $D$;

Evaluate the new solution using $\text{RMSE}(X_i^0, Y_i^0)$; //Evaluate each individual based on the numerical output of the fuzzy

**Step 2:** apply Differential Evolution Algorithm

Let $g = 0$;

Repeat

```
g = g + 1;
```

```
for i = 0 to $N_p$
    for j = 0 to l
        select at randomly $r1 \neq r2 \neq r3$ from 1 to $N_p$ and $j_1$ and $j_2$ from 1 to D;
        for k = 1 to D
            $y_{i,j,k}^g = x_{r1,j,k}^{g-1} + F \ast \text{rand} \ast x_{r2,j,k}^{g-1} - x_{r3,j,k}^{g-1}$; \( \ll y_{i,j,k}^g, X_{i,j,k}^{g-1} \) indicate the mutated
            vector and the crossed vector
```

end for // system using RMSE (the root-mean squared error)
if \((r_{X,i,j}^g \leq Cr \ or \ j = j_j^1)\) \(T_{X,i,j}^g = v_{X,i,j,k}^g\); \(\forall\) of membership functions of the input \(j\)

Else \(T_{X,i,j}^g = x_{i,j}^{g-1}\)

end for

Adjust \(\tau_{X,i,j}^g\) to ensure that \(T_{X,i,j}^g \geq T_{X,i,j-1}^g\) for each \(j = 1\) to \(D\);

for \(j = 1\) to \(I\)

select at randomly \(r \neq r_2 \neq r_3\) from \(1\) to \(N_p\) and \(j_{1}^j\) and \(j_{2}^i\) from \(1\) to \(D\);

for \(k = l\) to \(D\)

\(v_{Y_i,j,k}^g = x_{Y_i,j,k}^{g-1} + F \ast rand \ast (y_{r_1,j,k}^g - y_{r_2,j,k}^g)\) \(\forall\) \(V_{Y_i,j}^g\), \(T_{Y_i,j}^g\) indicate the mutated vector and the crossed vector

round down; \(\lfloor v_{Y_i,j,k}^g \rfloor\);

if \(\lfloor r_{Y_i,j,k}^g \leq Cr \ or \ j = j_j^2\) \(T_{Y_i,j,k}^g = v_{Y_i,j,k}^g\);

else \(T_{Y_i,j,k}^g = y_{i,j}^{g-1}\)

end for

end for

Adjust \(\tau_{Y_i,j}^g\) to ensure that \(T_{Y_i,j}^g \geq T_{Y_i,j-1}^g\) or \(T_{Y_i,j}^g \leq T_{Y_i,j-1}^g\) for each \(j = 1\) to \(D\);

Evaluate the new solution using \(\text{RMSE}(X_{X,i,j}^g, Y_{Y_i,j}^g)\);

If \((\text{RMSE}(X_{X,i,j}^g, Y_{Y_i,j}^g) < \text{RMSE}(X_{X,i,j}^{g-1}, Y_{Y_i,j}^{g-1}))\) then

\(X_i^g = T_{X,i,j}^g\) and \(Y_i^g = T_{Y_i,j}^g\);

else

\(X_i^g = X_i^{g-1}\) and \(Y_i^g = Y_i^{g-1}\);

End if

end for

until (a stopping criterion is found)

In the next section, a detailed analysis of the proposed approach on the standard test data is shown in general.

5. Data sets for Validation of the RDE 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 (KDD data set, 1999). 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 five million connection records as training data and about two 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: (1) 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. (2) the content features within a connection suggested by domain knowledge are used to assess the payload of the original TCP packets, such as the number of failed login attempts. (3) 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. (4) 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: (1) Denial of Service (DoS): making
some computing or memory resources too busy to accept legitimate users access these resources. (2)
Probe (PRB): host and port scans to gather information or find known vulnerabilities. (3) Remote to
Local (R2L): unauthorized access from a remote machine in order to exploit machine’s vulnerabilities.
(4) 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, we randomly select 18,285 records, similar to prior research (Beghdad, 2008). 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.

### 6. 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). We 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 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 presents 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 (Gomez & Dasgupta, 2001). As it is showed 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 we have used a subset of this large
data set as our 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.

We normalized the train and test data sets, where each numerical value in the data set is
normalized between 0 and 1. Table 2 exhibits features of the preprocessed KDD-Cup 99 data set. The
distribution of different classes in the train and test datasets is presented in Table 3.

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

<table>
<thead>
<tr>
<th>Class</th>
<th>Sub-classes</th>
<th>Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td></td>
<td>97,278</td>
</tr>
<tr>
<td>PRB</td>
<td>Ipsweep, nmap, portsweep, satan</td>
<td>4107</td>
</tr>
<tr>
<td>DOS</td>
<td>Back, land, neptune, pod, smurf, teardrop</td>
<td>391,458</td>
</tr>
<tr>
<td>U2R</td>
<td>Buffer_overflow, loadmodule, multihop, perl, rootkit</td>
<td>52</td>
</tr>
<tr>
<td>R2L</td>
<td>ftp, write, guess_passwd, imap, phf, spy, warezclient, warezmaster</td>
<td>1126</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>#</th>
<th>Feature name</th>
<th>Type</th>
<th>#</th>
<th>Feature name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Duration</td>
<td>C</td>
<td>22</td>
<td>is_guest_login</td>
<td>S</td>
</tr>
<tr>
<td>2</td>
<td>Protocol_type</td>
<td>S</td>
<td>23</td>
<td>Count</td>
<td>C</td>
</tr>
<tr>
<td>3</td>
<td>Service</td>
<td>S</td>
<td>24</td>
<td>srv_count</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>Flag</td>
<td>S</td>
<td>25</td>
<td>serror_rate</td>
<td>C</td>
</tr>
<tr>
<td>5</td>
<td>src_byte</td>
<td>C</td>
<td>26</td>
<td>srv_error_rate</td>
<td>C</td>
</tr>
<tr>
<td>6</td>
<td>dst_bytes</td>
<td>C</td>
<td>27</td>
<td>reror_rate</td>
<td>C</td>
</tr>
<tr>
<td>7</td>
<td>Land</td>
<td>C</td>
<td>28</td>
<td>srv_rerror_rate</td>
<td>C</td>
</tr>
<tr>
<td>8</td>
<td>wrong_fragment</td>
<td>C</td>
<td>29</td>
<td>same_srv_rate</td>
<td>C</td>
</tr>
<tr>
<td>9</td>
<td>Urgent</td>
<td>C</td>
<td>30</td>
<td>diff_srv_rate</td>
<td>C</td>
</tr>
<tr>
<td>10</td>
<td>Hot</td>
<td>C</td>
<td>31</td>
<td>srv_diff_host_rate</td>
<td>C</td>
</tr>
<tr>
<td>11</td>
<td>num_failed_logins</td>
<td>C</td>
<td>32</td>
<td>dst_host_count</td>
<td>C</td>
</tr>
<tr>
<td>12</td>
<td>logged_in</td>
<td>S</td>
<td>33</td>
<td>dst_host_srv_count</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>num_compromised</td>
<td>C</td>
<td>34</td>
<td>dst_host_same_srv_rate</td>
<td>C</td>
</tr>
<tr>
<td>14</td>
<td>root_shell</td>
<td>C</td>
<td>35</td>
<td>dst_host_diff_srv_rate</td>
<td>C</td>
</tr>
<tr>
<td>15</td>
<td>su_attempted</td>
<td>C</td>
<td>36</td>
<td>dst_host_same_scp_rate</td>
<td>C</td>
</tr>
<tr>
<td>16</td>
<td>num_root</td>
<td>C</td>
<td>37</td>
<td>dst_host_scp_srv_rate</td>
<td>C</td>
</tr>
<tr>
<td>17</td>
<td>num_file_creations</td>
<td>C</td>
<td>38</td>
<td>dst_host_scp_scp_rate</td>
<td>C</td>
</tr>
<tr>
<td>18</td>
<td>num_shells</td>
<td>C</td>
<td>39</td>
<td>dst_host_scp_scp_rate</td>
<td>C</td>
</tr>
<tr>
<td>19</td>
<td>num_access_files</td>
<td>C</td>
<td>40</td>
<td>dst_host_scp_scp_rate</td>
<td>C</td>
</tr>
<tr>
<td>20</td>
<td>num_outbounds_cmds</td>
<td>C</td>
<td>41</td>
<td>dst_host_scp_scp_rate</td>
<td>C</td>
</tr>
<tr>
<td>21</td>
<td>is_host_login</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>100</td>
<td>1000</td>
</tr>
<tr>
<td>U2R</td>
<td>50</td>
<td>59</td>
</tr>
<tr>
<td>R2L</td>
<td>100</td>
<td>1000</td>
</tr>
<tr>
<td>DOS</td>
<td>300</td>
<td>6500</td>
</tr>
<tr>
<td>PRB</td>
<td>100</td>
<td>1000</td>
</tr>
</tbody>
</table>

This section 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 we compare the performance of proposed RDE fuzzy systems to some of other intrusion detection algorithms. Tables 4 show the parameter specifications that we have used in our computer simulations for each of the mentioned fuzzy systems.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size ($N_{pop}$)</td>
<td>50</td>
</tr>
<tr>
<td>Crossover probability ($P_{c}$)</td>
<td>0.7</td>
</tr>
<tr>
<td>Mutation probability ($P_{m}$)</td>
<td>0.01</td>
</tr>
<tr>
<td>Mutation attempts ($M_{repeat}$)</td>
<td>30</td>
</tr>
<tr>
<td>Replacement percentage ($P_{rep}$)</td>
<td>20</td>
</tr>
<tr>
<td>Maximum number of generations</td>
<td>100</td>
</tr>
</tbody>
</table>

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

Table 5: Confusion matrix for the RDE based IDS

<table>
<thead>
<tr>
<th>Real class</th>
<th>Detected class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
</tr>
<tr>
<td>Normal</td>
<td>990</td>
</tr>
<tr>
<td>U2R</td>
<td>26</td>
</tr>
<tr>
<td>R2L</td>
<td>206</td>
</tr>
<tr>
<td>DOS</td>
<td>459</td>
</tr>
<tr>
<td>PRB</td>
<td>315</td>
</tr>
<tr>
<td>Precision%</td>
<td>49.6</td>
</tr>
</tbody>
</table>

As per the tabulations shown in Table 6 and 7, the precision rate of Michigan based RDE fuzzy system for R2L, DOS, and PRB patterns outperforms other approaches. Also it is clear that the IDS which develops using RDE 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.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
</tr>
<tr>
<td>RDE</td>
<td>99.23</td>
</tr>
<tr>
<td>EFRID (Gomez &amp; Dasgupta, 2001)</td>
<td>92.78</td>
</tr>
<tr>
<td>Winner entry (Elkan, 2000)</td>
<td>94.5</td>
</tr>
</tbody>
</table>

Table 7: Performance comparison

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Detection rate</th>
<th>False alarm rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDE</td>
<td>89.17</td>
<td>0.13</td>
</tr>
<tr>
<td>EFRID</td>
<td>98.15</td>
<td>7.0</td>
</tr>
<tr>
<td>RIPPER-Artificial Anomalies (Fan et al., 2001)</td>
<td>94.26</td>
<td>2.02</td>
</tr>
<tr>
<td>SMARTSIFTER (Yamanishi et al., 2000)</td>
<td>82</td>
<td>-</td>
</tr>
</tbody>
</table>

7. Conclusion

A refined differential evolution (RDE) 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 refined 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.

Reference


