SECURING NETWORK TRAFFIC USING GENETICALLY EVOLVED TRANSFORMATIONS

***Kamel Mohamed Faraoun*[1], *Aoued Boukelif*[2]**
[1]Evolutionary Engineering and Distributed Information Systems Laboratory, EEDIS, Djillali Liabès University, Sidi Bel Abbès, Algeria. Email: kamel_mh@yahoo.fr
[2]Laboratoire des télécommunications et du traitement numérique de signal, Équipe de recherche des techniques vidées, Djillali Liabès University, Sidi Bel Abbés, Alegria

ABSTRACT

The paper describes a new approach of classification using genetic programming. The proposed technique consists of genetically coevolving a population of non-linear transformations on the input data to be classified, and map them to a new space with a reduced dimension, in order to get maximum inter-classes discrimination. The classification of new samples is then performed on the transformed data, and so becomes much easier. Contrary to the existing GP-classification techniques, the proposed one uses a dynamic repartition of the transformed data in separated intervals, the efficacy of a given interval repartition is handled by the fitness criterion, with maximum classes discrimination. Experiments were first performed using the Fisher's Iris dataset, and the KDD'99 Cup dataset was used to study the intrusion detection and classification problem. Obtained results demonstrate that the proposed genetic approach outperforms the existing GP-classification methods, and gives accepted results compared to other existing techniques.

Keywords: Genetic programming, patterns classification, intrusion detection.

1.0 INTRODUCTION

Pattern classification concepts are important in the design of computerized information processing systems for many applications, such as remote sensing, medical diagnosis, sonar, radar etc. Pattern classification involves the development of theory and techniques for the categorization of input data into identifiable classes [25]. A pattern class is a category determined by some common attributes. A pattern is the description of any member of a category representing a pattern class. The application determines the measurement of features. Classification typically involves the mapping of an N-dimensional feature vector to one of multiple classes. The N-dimensional feature vector is like a point in the N-dimensional feature space. Samples belonging to a particular class give rise to a data distribution of that class in some region of the feature space.

It is possible for data distributions of two classes to be either overlapping or non-overlapping in the feature space. A pattern classifier determines the decision boundaries between different classes. The complexity of these boundaries may range from linear to non-linear surfaces. The significance of decision boundaries lies in the fact that they can usually be generated by utilizing representative patterns from each class. The pattern classifier uses these decision boundaries and determines the class of a new pattern. In the present work, we consider the problem of classifying real number vectors form R^N, where N is the number of features of a given pattern.

The basic problem in pattern classification is to develop decision functions that divide the feature space into regions each of which contains sample patterns belonging to a class. Intrusions in computer networks can be traced and detected by collecting information about the traffic in and out of the network. From a pattern classification point of view, the network intrusion detection problem can be formulated as follows: given the information about network connections between pairs of hosts, assign each connection to one of N data classes representing normal traffic or different categories of intrusions (e.g., Denial of Service, access to root privileges). It is worth noting that various definitions of data classes are possible. The term "connection" refers to a sequence of data packets related to a particular service, e.g., the transfer of an image via the ftp protocol. The intrusion detection problem can then be viewed as a multi-category pattern classification problem, when each connection feature constitutes one pattern to be assigned to one of the N existing class (depending on the number of intrusion types taken into account).

In this paper, an attempt is made to show the use of a new GP-classification approach to perform network intrusion detection. Section 1 gives some background theory about genetic programming approaches and related works. Section 2 explains the method developed in the present work with its different elements and parameters. In Section 3, a description is given of the two datasets used for experiments and the codification of the different data elements. Section 4 summarizes the different results obtained and gives comparison with other approaches with discussion.

Further enhancements of the proposed method are explained in Section 5. The paper is finally concluded with a summary of the most important points and future works.

2.0 BACKGROUND

2.1 Genetic Programming Paradigm

Genetic programming (GP) is an extension of genetic algorithms (GA) [9]. It is a general search method that uses analogies from natural selection and evolution. In contrast to GA, GP encodes multi-potential solutions for specific problems as a population of programs or functions. The programs can be represented as parse trees. Usually, parse trees are composed of internal nodes and leaf nodes. Internal nodes are called primitive functions, and leaf nodes are called terminals. The terminals can be viewed as the inputs to the specific problem. They might include the independent variables and the set of constants. The primitive functions are combined with the terminals or simpler function calls to form more complex function calls.

GP randomly generates an initial population of solutions. Then, the initial population is manipulated using various genetic operators to produce new populations. These operators include reproduction, crossover, mutation, dropping condition and others. The whole process of evolving from one population to the next is called a generation. A high-level description of GP algorithm can be divided into a number of sequential steps:

- Create a random population of programs or rules using the symbolic expressions provided as the initial population.
- Evaluate each program or rule by assigning a fitness value according to a predefined fitness function that can measure the capability of the rule or program to solve the problem.
- Use reproduction operator to copy existing programs into the new generation.
- Generate the new population with crossover, mutation, or other operators from a randomly chosen set of parents.
- Repeat steps 2 onwards for the new population until a predefined termination criterion has been satisfied, or a fixed number of generations have been completed.
- The solution to the problem is the genetic program with the best fitness within all the generations.

In GP, crossover operation is achieved firstly by reproduction of two parent trees; two crossover points are then randomly selected in the two offspring trees. Exchanging sub-trees, which are selected according to the crossover point in the parent trees, generates the final offspring trees. The obtained offspring trees are usually different from their parents in size and shape.

Mutation operation is also considered in GP. A single parental tree is firstly reproduced. Then a mutation point is randomly selected from the reproduction, which can be either a leaf node or a sub-tree. Finally, the leaf node or the sub-tree is replaced by a new leaf node or sub-tree generated randomly. Fitness functions ensure that the evolution goes towards optimization by calculating the fitness value for each individual in the population. The fitness value evaluates the performance of each individual in the population.

2.2 Genetic Programming and Classification Task

Generally, GP trees can perform classification by returning numeric (real) values and then translating these values into class labels [10]. For binary classification problems, the division between negative and non-negative numbers acts as a natural boundary for a division between two classes. This means that genetic programs can easily represent binary class problems. While evaluating the GP expression for an input data, if the result of the GP-expression is ≥ 0, the input data is assigned to one class; else, it is assigned to the other class. Thus, the desired output D is +1 for one class and is -1 for the other one in the training set. Hence, the output of a GP-expression is either +1 (indicating that the input data belongs to that class) or -1 (indicating that the input sample does not belong to that class). During the genetic evolution of individuals, the best individual is that who correctly classifies the maximum of training samples, the positive samples must give a value of +1 for the output, and negative samples must give -1.

Given a set of training data $D_{Train}=\{X_1,X_2,.....X_p\}\subset R^N$, a binary classifier is a GP-expression T, so that:

$$\begin{aligned} &T(X_i) \le 0 \ \text{ if } \ X_i \in \text{Class 1} \ \ (D = +1) \\ &T(X_i) > 0 \quad \text{otherwise} \quad (D = -1) \end{aligned} \tag{1}$$

GP is guided by the fitness function to search for the most efficient computer program to solve a given problem. A simple measure of fitness has been adopted for the binary classification problem:

$$\text{Fitness (T)} = \frac{\text{Number of samples classified correctly}}{\text{Number of samples used for training during evolution}} \tag{2}$$

Each evolved genetic expression maps the samples space of the X_i's to the real numbers set: R, and attributes the interval $]-\infty, 0]$ to the class 1 and the interval $]0,+\infty[$ to the class 2. This mapping is static, but it can achieve good results for 2-category classification problems. Unfortunately, when more than two classes are involved (n-classes problem), finding meaningful division points over the set of real numbers returned by the genetic programs becomes more difficult. If boundary regions are chosen at arbitrary points over the set of real numbers, genetic programs face the problem of not only containing the necessary elements to distinguish between classes, but must also perform a translation task to provide output in the necessary range pre-specified for a given class. Many alternatives were proposed by many authors to solve this problem. In [11], if there are n classes in a classification task, these classes are sequentially assigned to n regions along the numeric output value space from some negative numbers to positive numbers by (n-1)*thresholds/boundaries. Class 1 is allocated to the region with all numbers less than the first boundary; class 2 is allocated to all numbers between the first and the second boundaries and class n to the region with all numbers greater than the last boundary n-1, as shown in the following:

$$\text{Classe}(X_i) = \begin{cases} \text{classe 1} & \text{if } \ T(X_i) \le b_1 \\ \text{classe 2} & \text{if } \ b_1 \le T(X_i) \le b_2 \\ ..... & \\ \text{classe n -1} & \text{if } \ b_{n-3} \le T(X_i) \le b_{n-2} \\ \text{classe n} & \text{if } \ b_{n-2} \le T(X_i) \le b_{n-1} \end{cases} \tag{3}$$

In this equation, n refers to the number of object classes, T is the GP-expression evolved, $T(X_i)$ is the output value, and b_1, b_2, b_{n-1}, b_n are static pre-defined class's boundaries.

An alternative approach to static range selection, where ranges are arbitrarily chosen to correspond to class boundaries that all programs for the run must adhere to, is to allow each program to use a separate set of ranges for class boundaries that are dynamically determined for each individual program. Given a classification problem with many training examples and an individual from a GP population it is possible to use a subset of the training examples and record the values that are returned when attributes for specific classes are used as inputs. Based on these outputs, the effectively infinite range of the real numbers can then be segmented into regions. Those regions correspond to class boundaries based on areas whose values were returned by the program. This method was implemented in [11].

However, the GP employed for classification tasks have a requirement for long training times when compared to many other classification methods. It is also often quite difficult to extract a meaningful reason justifying the selection of a given class. Because of these factors the GP method is preferably applicable to tasks where accuracy is the most important factor in classification, and training times and understanding ability are seen as relatively unimportant. The major considerations in applying GP to pattern classification are:

- GP-based techniques are data distribution-free, so no a priori knowledge is needed about statistical distribution of the data;
- GP can directly operate on the data in its original form;
- GP can detect the underlying but unknown relationship that exists among data and express it as a mathematical expression;
- GP can discover the most important discriminating features of a class during the training phase;

The generated expression can be easily used in the application environment.

2.3 Related Works

The use of genetic programming to solve the multi-category classification and the intrusion detection problems has been attempted in many researches in different ways. In [12], Loveard et al. proposed five methodologies for multi-category classification problems. Of these five methodologies, they have shown that dynamic range selection method is more suitable for multi-class problems. In this dynamic range selection scheme, they record the real valued output returned by a classifier (tree or program) for a subset of training samples. The range of the recorded values is then segmented into regions to represent class boundaries. If the output of the classifier for a pattern belongs to a given region, then the class is assigned to this region. Once the segmentation of the output range has been performed, the remaining training samples can then be used to determine the fitness of an individual (or classifier). Chien et al. [13] used GP to generate discriminator functions using arithmetic operations with fuzzy attributes for a classification problem. In [14], Mendes et al. used GP to evolve a population of fuzzy rule sets and a simple evolutionary algorithm to evolve the membership function definitions. These two populations are allowed to co-evolve so that both rule sets and membership functions can adapt to each other. For a C-class problem, the system is run C-times. Kishore et al. [3] proposed an interesting method which considers a class problem as a set of two-class problems. When a GP classifier expression (GPCE) is designed for a particular class, that class is viewed as the desired class and the remaining classes taken together are treated as a single undesired class. So, with GP runs, all GPCEs are evolved and can be used together to get the final classifier for the C-class problem. They have experimented with different function sets and incremental learning. In [15], Durga and Nikhil R. proposed a method to design classifiers for a C-class pattern classification problem using a single run of GP. For a class problem, a multi-tree classifier consisting of C-trees is evolved, where each tree represents a classifier for a particular class. The performance of a multi-tree classifier depends on the performance of its constituent trees. A new concept of unfitness of a tree was exploited in order to improve genetic evolution. Weak trees having poor performance are given more chance to participate in the genetic operations so that they get more chance to improve themselves.

In [10], Mengjie and Will proposed two new approaches to ameliorate the performances of genetic classification algorithms. Rather than using fixed static thresholds as boundaries to distinguish between different classes, this approach introduces two methods of classification where the boundaries between different classes can be dynamically determined during the evolutionary process. The two methods are centred dynamic class boundary determination and slotted dynamic class boundary determination. Their obtained results suggest that, while the static class boundary method works well on relatively easy object classification problems, the two dynamic classes boundary determination methods outperform the static method for more difficult, multiple class object classification problems.

The mentioned approaches were tested on different dataset publicly available, like the IRIS dataset, the Cancer dataset, the Australian Credit Card and the Fisher's Iris data or the Heart Disease datasets, which are relatively very small and limited compared to the intrusion detection problem ones. The most important work on GP-classification for intrusion detection is the one presented in [1] by Dong Song, where a Page-based Linear Genetic Programming is implemented with a two-layer Subset Selection scheme to address only the two-class intrusion detection classification problem. The same author introduced a hierarchical RSS-DSS algorithm for dynamically filtering large datasets to enhance the system performances in [2]. Less important works can be found in [16], [17] with the Chimera model, and [18].

3.0 PROPOSED GP-CLASSIFICATION APPROACH

The present work proposes a new approach of a dynamic GP-based classifier which consists of genetically coevolving a population of non-linear transformations on the input data to be classified, and map them to a new space with a reduced dimension (1-D) in order to get a maximum inter-classes discrimination. Let $D_{Train}=\{X_1,X_2,\ldots..X_p\}\subset R^N$ be the set of training data. Since the proposed approach belongs to the supervised learning category, each sample X_i can be labelled with its class identifier j and become X_i^j. The set D_{Train} can then be subdivided into n sub-set corresponding to n learned classes, such that:

$$D_{Train} = \bigcup_{j \le n} D^j_{Train} \quad , D^j_{Train} = \left\{ X^j_i \in D_{Train} \,/\, class(X^j_i) = j \right\} \tag{4}$$

The output value for each sample from each training sub-set is computed using the GP-expression T, this allows the computation of a transformed map $T(D_{Train}^{i})$ for each sub-set D_{train}^{I} like the following:

$$T(D_{Train}^{j}) = \left\{ Y = T(X_i^j) \;\; / \;\; X_i^j \in D_{Train}^{j} \right\} \tag{5}$$

The classification approach assigned to each class j, the region covered by the set $T(D_{Train}^{j})$. When a new sample Y is presented to the classifier, the corresponding class is deduced according to the following:

$$\text{Classe}(Y) = \begin{cases} \text{classe 1} & \text{if } T(Y) \in T(D_{Train}^{1}) \\ \text{classe 2} & \text{if } T(Y) \in T(D_{Train}^{2}) \\ \ldots\ldots & \\ \text{classe n -1} & \text{if } T(Y) \in T(D_{Train}^{n-1}) \\ \text{classe n} & \text{if } T(Y) \in T(D_{Train}^{n}) \end{cases}$$

If the value of T(Y) does not appear in any set $T(D_{Train}^{j})$, we assign Y to the nearest class using the algorithm presented in Fig. 3.

We can see that the proposed classification method transform the problem from an N-dimensional vectors classification to a 1-dimentional values classification. The classification of the transformed vectors becomes much easier, but this is assured if a maximum discrimination exists between the sets $T(D_{Train}^{i})$. It is role of the genetic programming system to assure such criteria, the fitness of each transformation T depend on its ability to give a maximum discrimination between the $T(D_{Train}^{i})$'s.

There is a trade-off between the generality and power of this classification approach search. To perform a relatively unbiased search and allow the saliencies of the problem to emerge, the proposed approach has many degrees of freedom in its representation of the solution. Rather than evolve the class predictors directly and further encumber the genetic program, features are evolved which are then passed to a simple classifier. This hybrid approach assists the global search of the genetic program with the local search of the simple classifier. The classifier, with its malleable decision boundaries, performs local tuning of the solution to compensate for the genetic program's difficulty with evolving constants. In the following, we present the different steps of the classification approach: the learning phase, which consists of searching for the best transformation of the training data D_{Train}, and the test phase that classify each test sample from a set of new vectors D_{Test}.

3.1 The Learning Phase

3.1.1Terminals and Functions

The GP-transformations are built using a terminal set Tr and a function set Fn. The terminals are the fields of the used training dataset: $Tr=\{V_1,V_2,\ldots,V_N\}$ and we also used constants as terminals. These constants are randomly generated using a uniform distribution. To be consistent with the feature terminals, we also set the range of the constants as [-100, 100]. The functions set include:

- Arithmetic operators: +, -, /, *, ^;
- Non-linear functions: Sin ,Cos ,Ln , Log ,Exp ,Tan;

The +, - , and * operators have their usual meanings: addition, subtraction and multiplication, while / represents "protected" division which is the usual division operator except that a divide by zero gives a result of zero. Each of these functions takes two arguments. The transformations T_i are represented by hierarchical S-expressions trees, as proposed in the standard Koza implementation.

3.1.2The Fitness Function

For a given training set $D_{Train} \subset R^N$, the genetic programming system evolves a population of transformations T. In order to compute the fitness of each one, we need to define a distance between the mapped sets $T(D_{Train}^{i})$. The value of the fitness must express the inter-classes discrimination and separation. During our experiments, we have tested many fitness measurements, such as the maximum distance between gravity centres of the mapped classes and the

inter-classes and intra-classes variance criterion. But theses functions assume that the transformed sets $T(D_{Train}^{i})$ must be homogeneous and linearly separable. This condition is not always easy to achieve, so it is better to give the classification system the ability to generate separate but alternate transformed sets. Fig. 1 illustrates the two situations: (a) represent two pointes sets linearly separable (in a one dimensional space), and (b) show two separated pointes sets but in an alternated situation.

For this reason, we have proposed a new fitness function formula, which tries to minimize the total intersection between point sets, and search for a minimum number of common points between the mapped classes. The fitness function is inversely proportional to the computed number of common points between transformed sets $T(D_{Train}^{i})$. Height values of the fitness signify that the transformed sets have a very small intersection region, and then the discrimination between each set elements becomes easier. The fitness value is computed by:

$$\text{Fitness}(T) = \frac{\text{Card}(\bigcap_{i \le n} T(D_{Train}^{i}))}{\text{Card}(T(D_{Train}))}$$

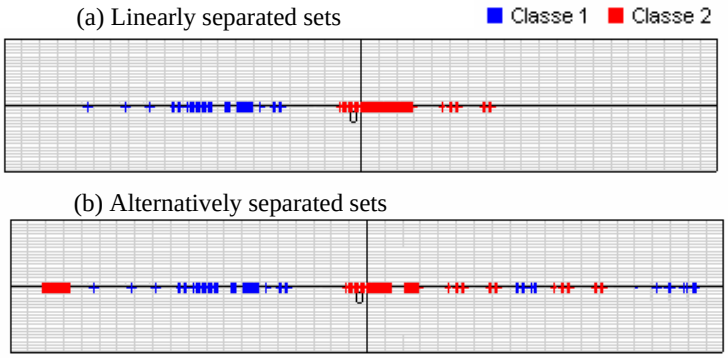

Fig.1: The two possible separation situations between two point sets.

When the function Card(X) gives the cardinality of a given set X. The performed experiments show that this function give the best classification rates with respect to other fitness functions mentioned above. The classification system become more flexible, and explores new solution unexploited by the other fitness functions.

3.1.3Genetic Operators and Parameters

The standard crossover and mutation operators presented in Section 1.1 are used in this implementation. Each transformation T is represented by a binary tree and the genetic operators always produce valid binary expressions. To control the maximum depth of the generated expressions, we use a modifiable parameter to control the length of the generated expressions. The genetic evolution process stops when it reaches a given generation count (termination criteria). Table 1 gives an overview of the parameters used in our implementation and the default value used for each one. During the evolution process, the result of each transformation T_i is bounded in a fixed interval ([-100,100] by default), to avoid having scatter sets in R.

Table 1: Set of parameters used to control the genetic evolution process

Parameter	**Value**
Generating constant probability	5%
Generating functions probability	70%
Crossover rate P_c	70%
Mutation rate P_m	10%
Population size	100
Maximum generations count	1000
Maximum individual's length	350
Minimum individual's length	30
Selection strategy	Roulette selection
Functions set	{+, - ,/ ,*, sin, cos, log, ln, tan, exp }
Terminals set	[-100,100]∪ {input variables}

The result of the genetic evolution during the training phase is the best generated transformation T, with the transformed sets $T(D_{Train}^{i})$. This output is used in the test phase to classify new samples.

3.2 The Test Phase: Classification of Unseen Samples

Let $D_{Test}=\{Y_1,Y_2,\ldots,Y_k\}$ be a new set of samples to be classified. Each vector $Y_i \in D_{Test}$ must be assigned to one of the n involved classes. To accomplish this task, the classification system operates as the following: First, a post-treatment algorithm is added to the classification system to compute a density array for the points of $T(D_{Train})$. This array is used with the transformation T during the test phase to deduce the class of each element Y_i form D_{Test}. This algorithm is presented as the following (Fig.2):

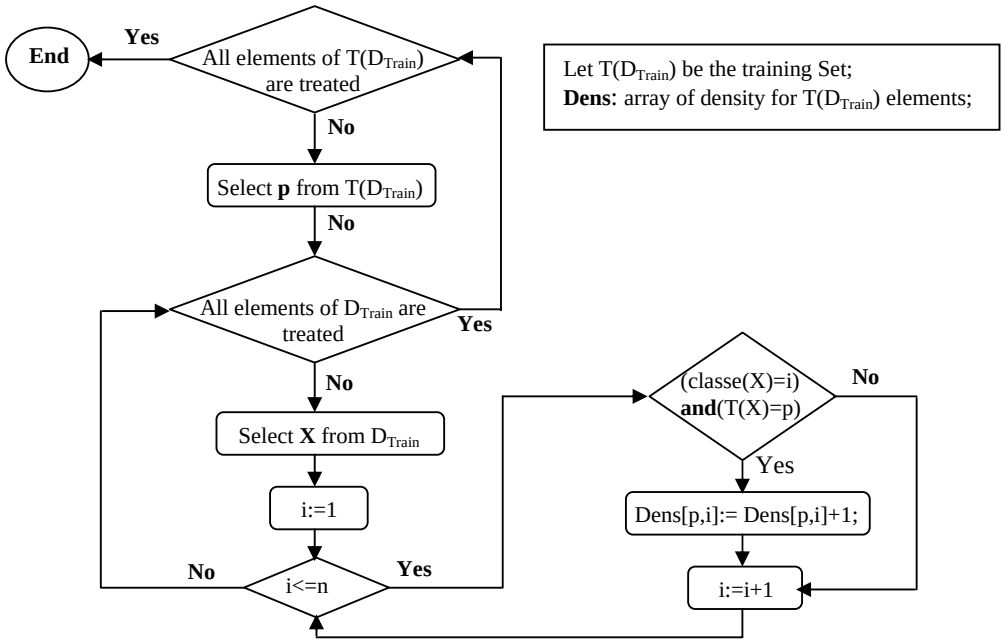

Fig.2: A post-treatment algorithm to generate density array, used during the testing phase

Then, for each new sample Y_i form D_{Test} , the corresponding class is determined using the following algorithm (Fig.3):

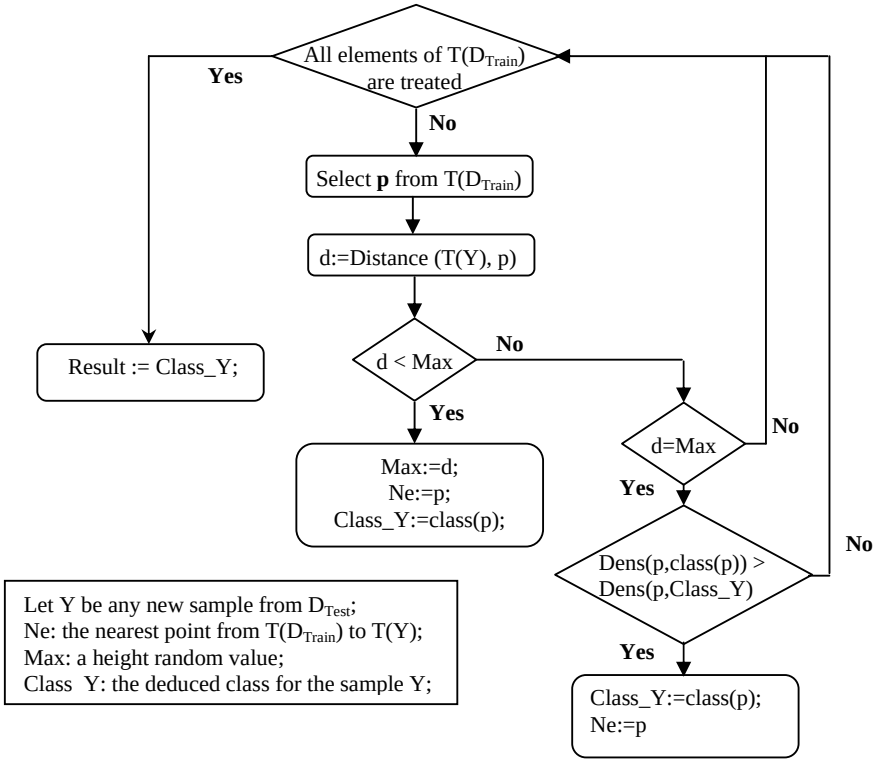

Fig.3: The proposed algorithm to deduce the class of new test samples Y_i, used during the testing phase

As shown by the experiments, these algorithms combined with the fitness function mentioned above, give better results than using classical fitness measurement, due to the flexibility of the classes distribution accorded to the genetic classification system.

4.0 DATASETS AND EXPERIMENTS

The proposed classification approach is benchmarked using two different datasets: the Fisher IRIS [19] dataset and the MIT KDD99 dataset [20]. The first one is used for comparison purpose, and to demonstrate the proposed method's capabilities. It is relatively very small and limited compared to the intrusion detection problem datasets. The second one concerns our problem of interest: the network intrusion detection. The KDD99 dataset is the most used for intrusion detection problems, collected at the Lincoln Laboratory of MIT, under DARPA sponsorship, which consists of about 5,000,000 connection records, with 41 data fields. The most important work on GP-classification using the KDD99 dataset is the one presented in [1, 2] by Dong Song, where a Page-based Linear Genetic Programming is implemented with a two-layer Subset Selection scheme to address only the two-class intrusion detection classification problem.

The first IRIS dataset was divided equally into a training set and a test validation set. The specific training sets for Iris setosa, versicolor and viginica are derived from the training set. To perform the experiments with the KDD99 dataset, the '10% KDD' set was sampled and only 24788 records were used to train our system. For test purposes, we use the whole 'Corrected (Test)' in almost all the implemented approaches. Table 2 lists the class's distributions of our used sets.

Table 2: Distribution of the normal and attack records in the used training and testing sets.

	Training Set		Testing Set	
Normal	11673	47.09 %	60593	19.48 %
DOS	7829	31.58 %	229853	73.90 %
PBR	4107	16.56 %	4166	1.34 %
R2L	1119	4.51 %	16347	5.25 %
U2R	52	0.24 %	70	0.02 %

Attributes in the KDD datasets had all forms :continuous, discrete, and symbolic, with significantly varying resolution and ranges. Most pattern classification methods are not able to process data in such a format. Hence, pre-processing was required before pattern classification models could be built. Pre-processing consisted of two steps: first step involved mapping symbolic-valued attributes to numeric-valued attributes and the second step involved scaling. In the present work, we have used the data codification and scaling presented in [21]. All the resulting scaled fields belong to the interval [0, 1].

Table 3 summarizes the 41 fields used in the KDD99 dataset regrouped into three mentioned categories. Each field is labelled with a symbolic notation (F_1, F_2,…, F_{41}) to be used as terminals during the genetic process.

All tests were performed on an Intel-Pentium 4 CPU 2.66 GHz with 256 Mb Ram size. The performances of intrusion detection for the classifier are computed using the following expressions:

$$\begin{aligned} &\text{Detection rate} && DR = 1 - \frac{\text{False negatives number}}{\text{Total Number of Attaks}} \\ &\text{False Positive Rate} && FP = \frac{\text{False Positives}}{\text{Total Number of normal connections}} \end{aligned} \tag{8}$$

The KDD99 dataset contains four attack categories; so the problem of intrusion detection is extended to an intrusion classification one. The generated classifier (GP expression) must learn to discriminate between the different types of attack, and associate each intrusion to the corresponding type. A classification rate is computed to evaluate the classification ability of each transformation using the expression (9) mentioned in Section 4.1.

Table 3: The KDD99 used features, grouped into 3 categories

Basic features of individual TCP connections		Content features suggested by domain knowledge		Traffic features computed using a two-second time window	
duration	**F1**	hot	**F10**	count	**F23**
protocol_type	**F2**	num_failed_logins	**F11**	srv_count	**F24**
service	**F3**	logged_in	**F12**	serror_rate	**F25**
flag	**F4**	num_compromised	**F13**	srv_serror_rate	**F26**
src_bytes	**F5**	root_shell	**F14**	rerror_rate	**F27**
dst_bytes	**F6**	su_attempted	**F15**	srv_rerror_rate	**F28**
land	**F7**	num_root	**F16**	same_srv_rate	**F29**
wrong_fragment	**F8**	num_file_creations	**F17**	diff_srv_rate	**F30**
urgent	**F9**	num_shells	**F18**	srv_diff_host_rate	**F31**
		num_access_files	**F19**	dst_host_count	**F32**
		num_outbound_cmds	**F20**	dst_host_srv_count	**F33**
		is_hot_login	**F21**	dst_host_same_srv_rate	**F34**
		is_guest_login	**F22**	dst_host_diff_srv_rate	**F35**
				dst_host_same_src_port_rate	**F36**
				dst_host_srv_diff_host_rate	**F37**
				dst_host_serror_rate	**F38**
				dst_host_srv_serror_rate	**F39**
				dst_host_rerror_rate	**F40**
				dst_host_srv_rerror_rate	**F41**

5.0 RESULTS AND COMPARISON

The results of the proposed GP classification approach for the 2 n-classes pattern classification problems described above, using the set of parameters presented in Table 1 is as following:

5.1 Fisher IRIS Classification Problem

The dataset was divided equally into a training set and a test validation set (75 samples in each set). The result of each test is a classification matrix C computed by the following algorithm (Fig.4):

```
Let T(D_Test) be the training Set;
n is the number of classe
i is the true class of the sample and k is the assigned class.
For i=1 to n do {
   For j=1 to n do C[i,j]:=0;
      For i=1 to Card(T(D_Test^i)) do
            {Apply the classifier and assign class k to input sample.
             C[i,k]:=C[i,k]+1;
             }
 }
```

Fig. 4: The algorithm used to compute the classification matrix

The classification rate is then computed using the following expression:

$$CR = \frac{\text{Number of samples classified correctly}}{\text{Number of samples used for training during evolution}} * 100 \qquad (9)$$

Table 4 gives the classification matrix obtained using the proposed approach. Tables 5 and 6 give the results of the classification process using a maximum likelihood classifier and a GP-based classification approach proposed in [3].

Table 4: Obtained classification matrix using the proposed approach

	Setosa	Versicolor	Viginica
Setosa	25	0	0
Versicolor	0	25	0
Viginica	0	1	24

Table 5: Classification matrix for Iris data set with maximum likelihood classifier [3].

	Setosa	Versicolor	Viginica
Setosa	25	0	0
Versicolor	0	24	1
Viginica	0	2	23

Table 6: Classification matrix for GPCE with interleaved training sets for Iris data [3].

	Setosa	Versicolor	Viginica
Setosa	25	0	0
Versicolor	0	24	1
Viginica	0	2	23

Table 7 gives a comparison between the detection rate obtained with different classifiers as presented in [24, 3], and our proposed classification approach. Table 8 gives the optimal transformation T (best individual) obtained after the GP running, with the corresponding classification rate. Fig. 5 shows the distribution of the transformed training set $T(D_{Train})$ obtained with this transformation.

Table 7: A summary of the classification rates obtained using different classifiers for the Fisher's Iris dataset

Method	**NN**	**Naive Bayse**	**BayseNet**	**C4.5**	**GPCE [3]**	**Maximum Likelihood**	**GP-classification**
Classification rate (DR)	96 %	96 %	94.667 %	94.67 %	96 %	97.3 %	**98.6 %**

Table 8: Best obtained individual using the proposed approach for the Fisher's Iris dataset

Best individual	**Classification rate (DR)**
(((((ln((-(((log2(exp((((F3)^2)/(F4))*(F3))))^2)-(F4)))^2))+(F4))*(F4))+(F3)))	**98.6 %**

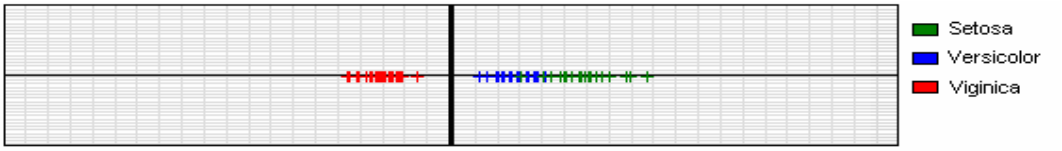

Fig.5: Distribution of the transformed training set $T(D_{Train})$ of the best obtained individual

From Table 7, we can see that our proposed approach gives the best classification rate compared to other proposed approaches. Only one sample from the "Viginica" set is misclassified.

5.2 KDD99 Dataset: The Intrusion Detection Problem

As we see in Table 1, the KDD99 dataset is more voluminous than the Iris Fisher's one, and contains more classes (5 classes). Discrimination is also very difficult in the intrusion detection case because the classes are not clearly separable, so the classification task becomes harder. To evolve the GP classification system, the same parameters set presented in Table 1 is used. In Table 9, we present the classification matrix obtained. Fig. 6 and 7 illustrate the transformed training set T (DTrain) repartition and the fitness value evolution during the GP evolution. The best individual T is presented by the following expression:

T: (((((log2(tan(-(F3))))*(cos((tan((F5)+(((log2(tan(-(F3))))*(((log2(tan(-(F3))))*(F30))+(cos(F5))))*
((tan(F13))+(F30)))))+((tan(log2(F2)))+(F30)))))*(cos(F5)))*((18)+(cos((tan(log2(F13)))+(F30)))))

Table 9: Classification matrix obtained using the proposed approach

	Normal	Prob	Dos	U2R	R2L	% Correct
Normal	59769	500	112	49	163	**98.64 %**
Probe	562	3443	113	3	45	**82.65 %**
Dos	8411	768	220662	0	11	**96.00 %**
U2R	25	11	6	19	9	**09.82 %**
R2L	10612	2107	8	2059	1611	**27.14 %**
% Correct	**75.29 %**	**50.42%**	**99.89%**	**0.89%**	**87.60%**	

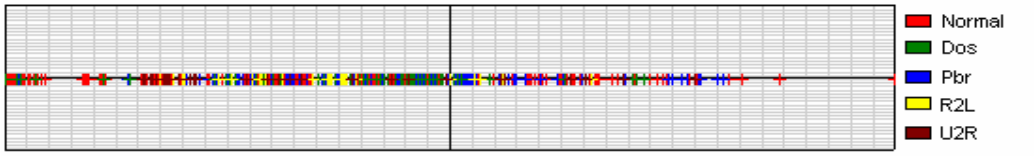

Fig.6: Distribution of the transformed training set $T(D_{Train})$ of the best individual

The following values of detection rate and the false positive rates were computed for the best obtained individual T:

Detection rate: DR = 0.925 (92.5 %)

False positive rate FP = 0.0135 (1.35 %)

Classification rate = 91.7 %

Fig.7: Evolution of the fitness value during the genetic process

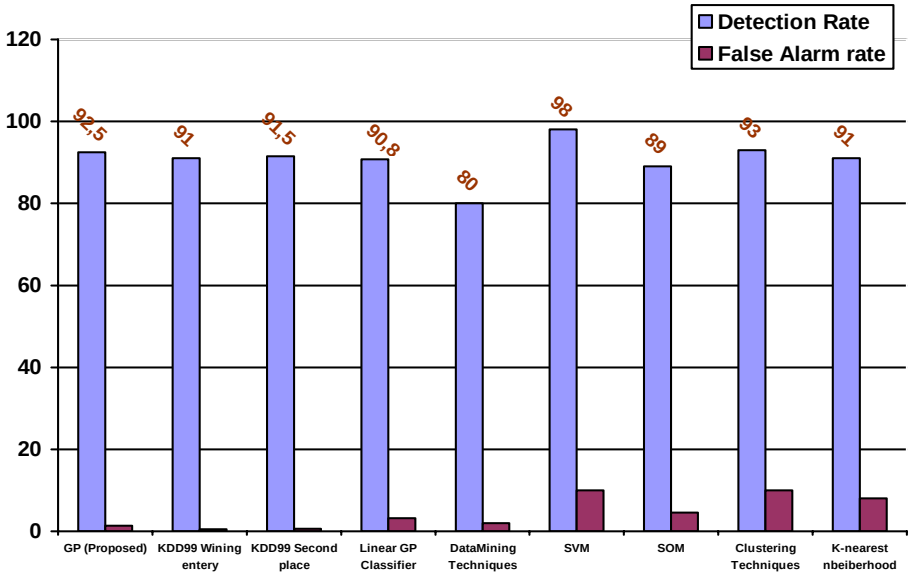

Fig.8: Comparison of detection rate and false positive rate results

Fig.8 summarizes and compares the detection rates and false positive rates obtained using the approaches mentioned above, and some recent results on KDD benchmark presented in [7] and [8]. All the mentioned approaches were tested using the 'Corrected (Test)' set of the KDD99 cup competition. The reported rates present the intrusion detection rate and false alarm rate for all the attack classes without considering the correct classification of each attack to its corresponding type.

We can see from the presented results that the proposed classification approach gives acceptable results compared to the other techniques. The highest detection rate is obtained using the support vectors machine technique implemented in [7], but with a height false positive rate (10 %) compared to 1.35% obtained with our proposed GP-classification approach.

It is reasonable to state that the set of pattern recognition and machine learning algorithms mentioned above offer an acceptable level of misuse detection performance for only two attack categories, namely Probing and DoS when tested on the KDD datasets, and failed to demonstrate an acceptable level of detection performance for the remaining two attack categories, U2R and R2L. To enhance the detection capabilities of our classification system, especially for categories R2L and U2R, we propose in the following an improvement of the proposed classification approach using multi-transformation approaches. The obtained results demonstrate that the capabilities can be highly ameliorated compared to the standard approach.

6.0 GP-CLASSIFIER ENHANCEMENT: THE MULTI-TRANSFORMATIONS CLASSIFICATION SYSTEM

6.1 Method Description

As explained in Section 2, the classification system use a single transformation (the best obtained individual) to transform each new sample, and then deduce the corresponding class using the algorithm presented in Fig. 4. The main idea of the multi-transformation system is to use a set of multiple transformations $TR_{set} = \{T_1, T_2, \ldots, T_p\}$ obtained genetically (the best ones) on the sample to be classified. Each transformation will output a corresponding class with a confidence factor for each sample Y from the testing dataset computed using the following expression:

$$\text{Confidence}(Y, T) = \frac{\text{Dens}(Ne, \text{Class\_Y})}{\text{Card}(T(D_{\text{Train}}^{\text{Class\_Y}}))} * \text{Fitness}(T) \qquad (10)$$

when :

- Ne is the nearest point from the $T(D_{Train})$ set;
- Class_Y is the deduced class for the sample Y
- $\text{Card}(T(D_{\text{Train}}^{\text{Class\_Y}}))$ is the number of the samples from the training set belonging to Class_Y.
- Fitness(T) is the fitness value of the transformation T
- Dens(Ne, Class_Y) is the density value computed by the algorithm of Fig.3

It is clear from the equation (10) that the confidence factor of a given sample in relation to a transformation T_i range in the interval [0, 1]. All the mentioned parameters are taken from the algorithm of Fig. 4. The equation (10) was introduced in the algorithm shown in Fig.9.

This algorithm returned for each sample Y, its corresponding class Ret_Class(Y, T), with the corresponding confidence factor Confidence(Y,T). The new classification system take the best individuals collected during the genetic evolution to construct a transformations set $TR_{set} = \{T_1, T_2, \ldots, T_p\}$. All these transformations are applied on each test sample Y during the testing phase to obtain p possible class and p corresponding confidence factor. These obtained outputs are combined to compute the membership factor of Y to each class c from the existing n classes (see Fig.10)

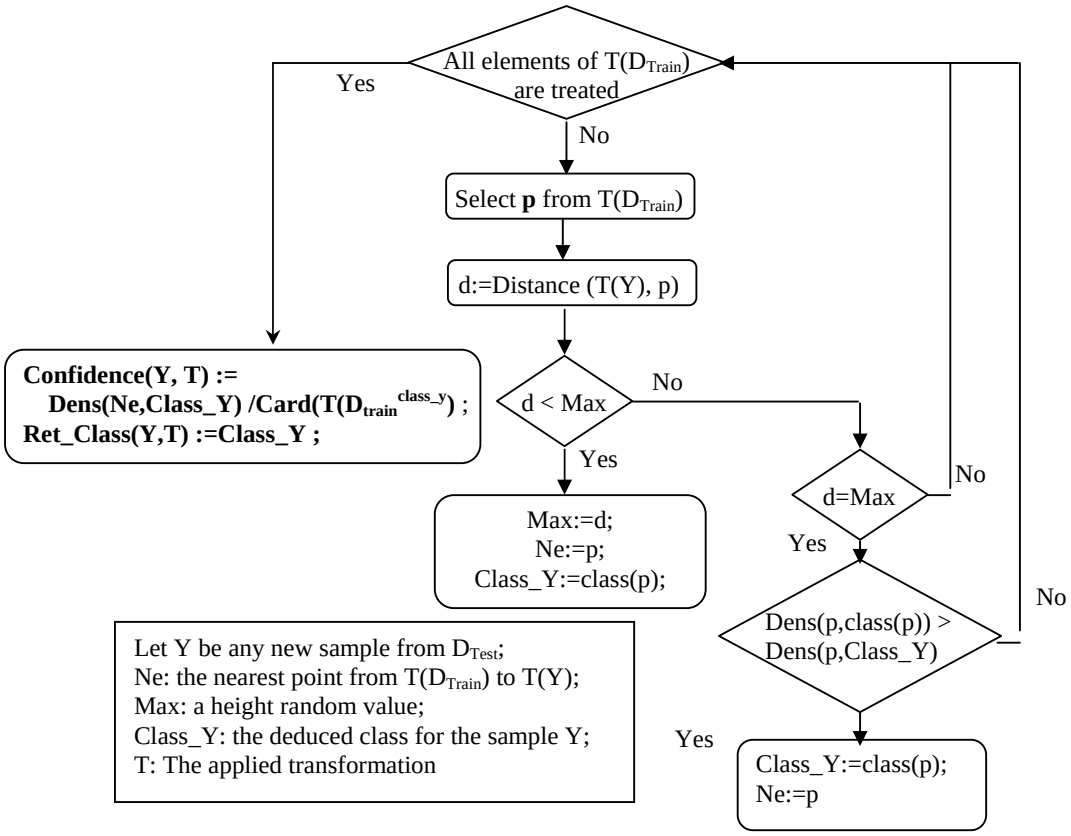

Fig.9: The modified version of the algorithm used to deduce the class of new test samples, and compute their confidence factor.

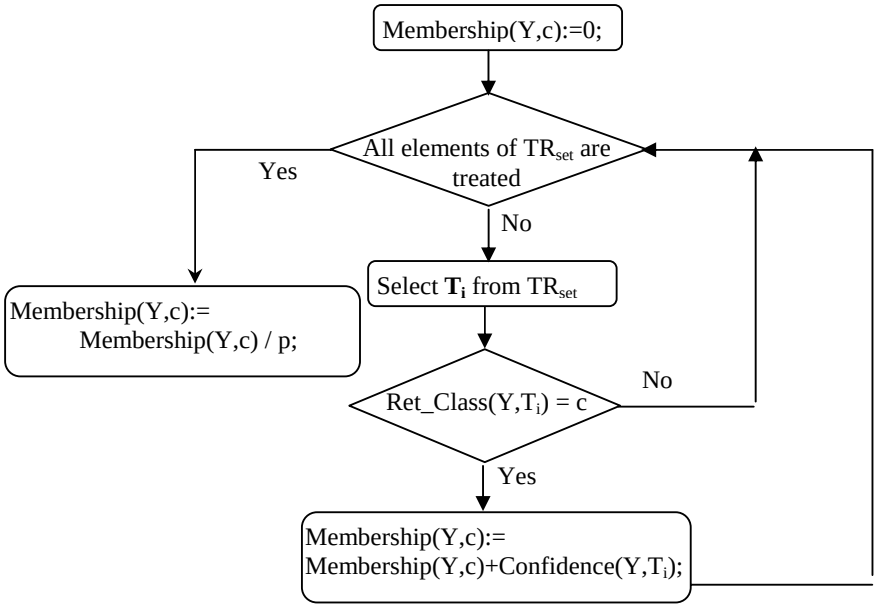

Fig.10: The proposed algorithm to compute the membership factor of a sample Y to a given class c.

It is clear from the formulas used above that the value of the membership factor range is always in the interval [0, 1]. The classification system assigned to Y, the class with the highest membership factor:

$$\text{Class}(Y) = c \quad \text{such that} \quad \text{Confidence}(Y, c) = \underset{1 \le i \le n}{\text{MAX}}(\text{confidence}(Y, i)) \tag{11}$$

This method benefits from the detection capabilities of each transformation T from the generated set TR_{set}, it acts like a rule system that averages the obtained decisions to elaborate the final one. The following results demonstrate the improvement achieved by this technique compared to the single transformation.

6.2 Results and Comparison

This section summarizes the results obtained using the multi-transformations classification system described above to detect and classify the intrusions in the KDD99 dataset. The test phase used the KDD99 'Corrected (Test)' set. The number of transformations p used in this experiment is fixed to 50 transformations collected during the learning phase realised by the genetic process. The following results give the average accuracy obtained for 40 GP trials conducted on the input training set. The classification, detection and false positive rates were computed in each GP trial.

Fig. 11 shows the variations of the detection rate for each class with respect to the number of used transformation. It is clear that better classification rates are allowed for the two classes R2L and U2R. The classification is ameliorated when augmenting the number of the used transformations. For the classes Normal and Dos, the maximum classification performances are reached starting from 6 or 7 transformations. Through the same way, it can be seen from Fig. 11 that the system reaches its maximum performance when the number of used transformations is maximum (a higher detection rate and a lower false positive rate).

Table 10 illustrates the classification matrix obtained with the best GP-trail using the multi-transformation method to classify the intrusions of the used KDD99 Test dataset with 50 collected transformations.

The performances rates obtained by the obtained solution are given by:

Detection rate: DR = 0.980 (98.0%)

False positive rate FP = 7E-4 (0.07%)

Classification rate = 99.05 %

Table 10: Classification matrix obtained using the multi-transformations classification method with 50 transformations

	Normal	**Prob**	**Dos**	**U2R**	**R2L**	**% Correct**
Normal	60550	21	10	4	8	**99.93%**
Probe	93	4053	15	0	5	**97.29%**
Dos	1792	911	227117	15	18	**98.81%**
U2R	21	6	2	38	3	**45.20%**
R2L	2973	154	21	85	13114	**80.22%**
% Correct	**92.54%**	**78.77%**	**99.97%**	**26.7%**	**99.74%**	

In Fig.12 and 13, obtained classification rates using the multi-transformations classification system are compared to the results presented in [23] using multiple classification systems such as Multilayer Perceptron (MLP), Gaussian classifier (GAU), nearest cluster algorithm (NEA), incremental radial basis function, K-means clustering (K-M), C4.5 decision tree and other techniques. The results show that classification rates obtained using the multi-transformations classification system for the classes R2L and U2R are very satisfactory with respect to the other techniques. The false positives detection rate of each attack class is unavailable for the SOM [8] and the linear GP [1, 2] techniques, since they are 2-category classifiers (normal and attack). Their false positive rates can be given only in terms of whole attacks classification.

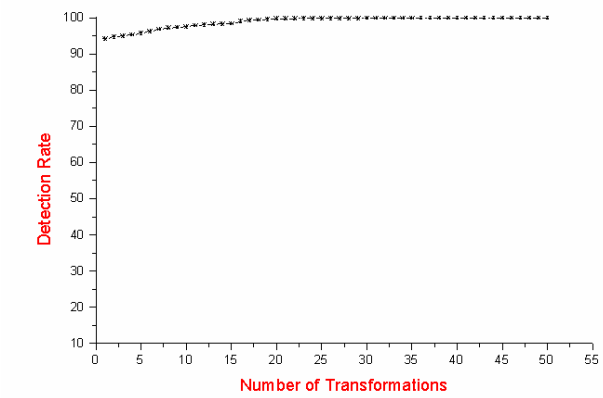

Fig.11.(a)

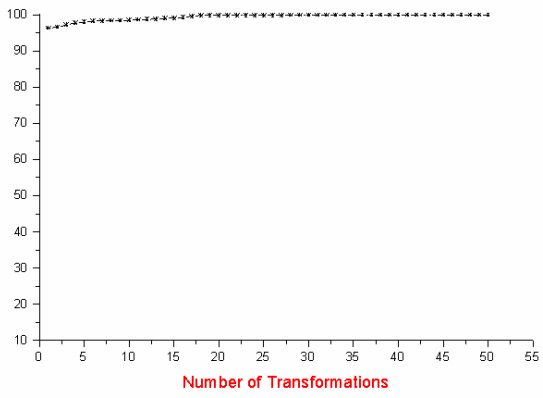

Fig.11. (b)

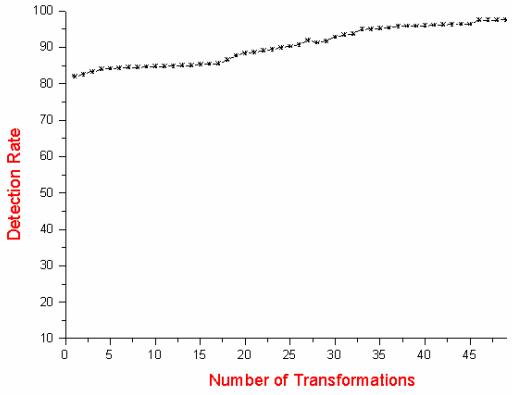

Fig.11. (c)

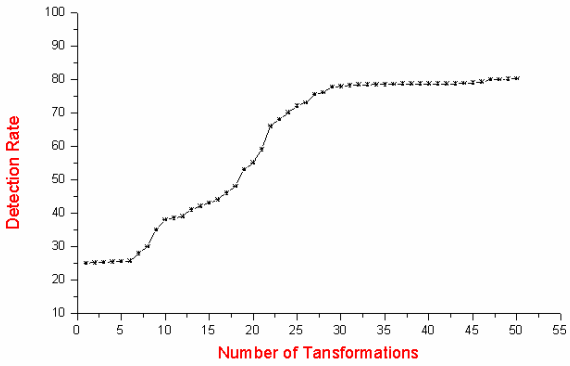

Fig.11. (d)

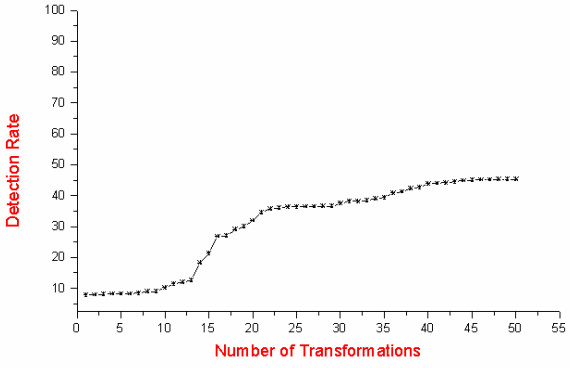

Fig.11. (e)

Fig.11: Evolution of the detection rate for each attack class with respect to the number of applied transformation: (a) Normal, (b) Dos, (c) Prob, (d) R2L and (d) U2R

In the present work, the multi-transformations classification system requires approximately 1 hour and 48 minutes to generate a set of 50 optimal transformations, when addressing the problem of intrusions classification using the mentioned KDD99 dataset. Compared to other existing solutions, the proposed classification approach has the potential to achieve best classification performances in shorter training time as illustrated in Table 11.

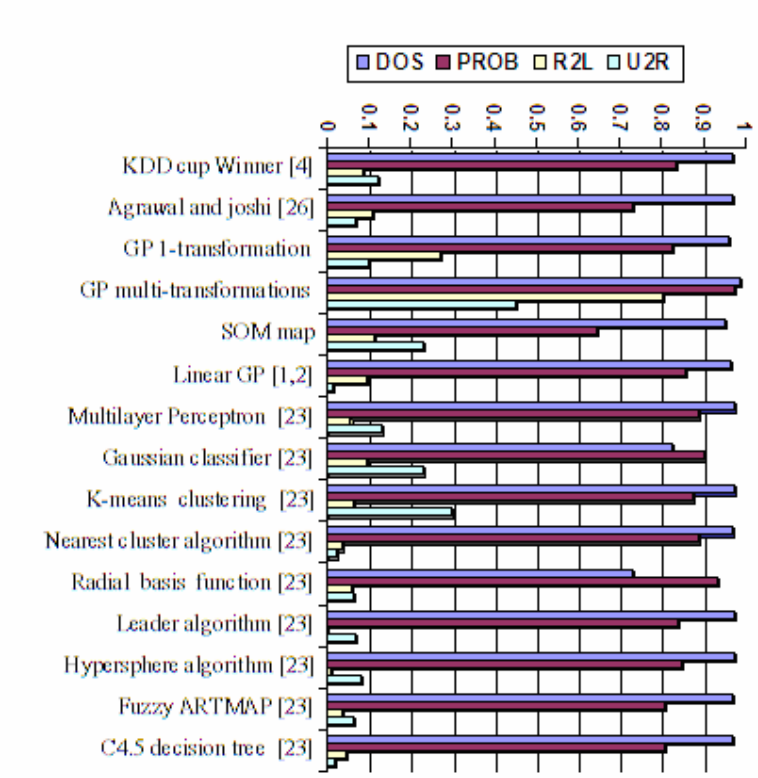

Fig.12: Comparison of the classification rates between different approaches

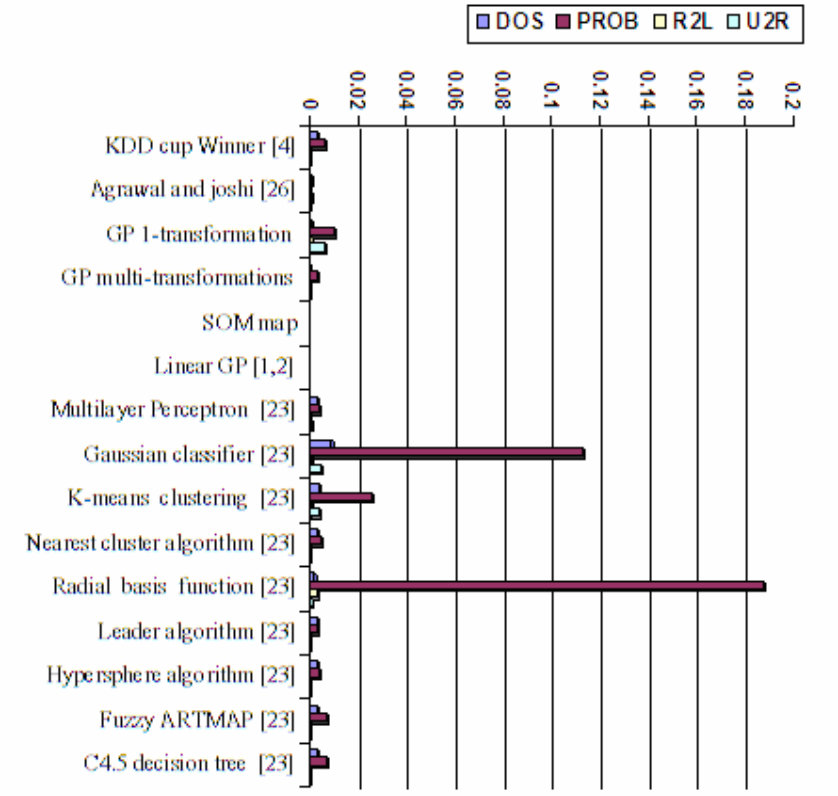

Fig.13: Comparison of the false positive classification rates between different approaches

Table 11: Comparison of the time and solution complexity of different classification methods

Classification method	Training time	Solution complexity
KDD wining entry	≈ 24 Hours	500 decision tree
KDD second place	≈ 22 Hours	755 decision tree
Linear GP [2]	≈ 7 Hours and 30 minute	A linear program with 86 instruction in 2 address format
GP with Multi-transformation	≈ 1 Hours and 48 minute	50 non-linear transformation with an average length of 150 character

7.0 RESULTS ANALYSIS

From the results, we can deduce that the proposed technique permits us to get better classification rates with respect to the other approaches, even if the detection rate is sometimes similar. The strength of this approach is in its ability to discriminate different attacks (attacks classification). The other approaches focus on the discrimination between the normal and intrusive behaviour. The complexity of the evolved solution is also very reasonable compared to decision trees, linear programmes or more complex neural networks. We can see from Fig. 13 and 14 that the classification rate of the multi-transformation approach is the best one, especially for categories R2L and U2R. It was proved through experiments that augmenting the number of the used transformations (beyond 50) enhanced the classification rates for these two classes without greatly decreasing the false positive rate. This is due to the limited number of used samples for training. Much better results should be obtained if more attacks samples are available.

8.0 CONCLUSION AND FUTURE WORK

In this work, a new Genetic Programming classification system with a dynamic class projection was implemented and tested on both Fisher's Iris dataset and the KDD'99 benchmark dataset, a problem involving multi-category classification task. To do so, populations of non-linear transformations are evolved to transform the input training data to be classified to a new one-dimensional space with a maximum discrimination between the projected classes. The classification task becomes much easier with the transformed data and the new testing samples are then transformed with the generated transformation and assigned to their corresponding class using a simple search algorithm (Fig. 4). The technique is independent of the dataset and structure of GP employed. Moreover, the framework has no specific hardware requirements, making use of the generic classifiers design which is already widely supported in computing systems. The proposed system is shown to be capable of learning attack and normal behaviour from the training data, and makes accurate predictions on the test data that also contains new attacks that the system was not trained on.

In order to enhance the classification performances, especially for some bad handled categories, a multi-transformation system was implemented and tested to combine the classification decisions of a large transformations set. The obtained results show that the proposed system can achieve much better classification performances without significant increase of the learning and detection run time. The examination of the proposed method shows that increasing the number of combined transformations significantly enhances the system performances.

In comparison with artificial intelligence approaches currently proposed, this approach provides competitive performance whilst utilizing a relatively small set of training samples. The time complexity of the approach is independent from the number of used fields (41 in the case of the KDD dataset) and is very satisfactory when compared to the other approaches (Table 11). The complexity of the generated solution is reduced in comparison to the solutions of other techniques. Each transformation is represented as a string with 150 characters (byte) at maximum, and can be easily transformed to an assembly routine and evaluated using a stack base schema, to be integrated in a real time detection system.

In terms of future work, the proposed classification approach can be extended to map the classes to a higher dimensionality space (especially for the 2D and 3D spaces). That is to say, a population of combinations of transformations $<T_1,T_2,..,T_p>$ is evolved for the training dataset to get the optimal combination that projects the data

to the specified space of dimensionality p. For example, in the 2D case, each individual is a couple $<T_1,T_2>$ that project each sample X_i from R^N to R^2 as follows:

$$T(X_i) =< T_1, T_2 > (X_i) = (y_1, y_2) \quad \text{such that} \quad \begin{cases} y_1 = T_1(X_i) \\ y_2 = T_2(X_i) \end{cases}, (y_1, y_2) \in \Re^2 \tag{12}$$

The same principal can be used for any p-dimensional space. Such approach has the potential to reduce the information loss due to the transformation operation, since a higher dimension can handle more information and relationship between the different initial components. Another important advantage is the possibility to generate a graphical visualisation of the transformed data (in the 2D or 3D case) which allows having different possible profiles of the classes' distribution, and to give some interpretations such as inter-classes proximity and intersections.

REFERENCES

[1] S. Dong, I. Malcolm. Heywood, and A. N. Zincir-Heywood. "Training Genetic Programming on Half a Million Patterns: An Example from Anomaly Detection", *IEEE Transactions on Evolutionary Computation*, 2005, Vol. 9, No. 3, pp 225-240.

[2] S. Dong, I. Malcolm. Heywood, and A. N. Zincir-Heywood. "A Linear Genetic Programming Approach to Intrusion Detection". *GECCO 2003*, 2003, LNCS 2724, pp. 2325–2336.

[3] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal. "Application of Genetic Programming for Multicategory Pattern Classification," *IEEE Transactions on Evolutionary Computation*, September 2000, Vol. 4, pp. 242–258.

[4] B. Pfahringer, "Winning the KDD99 Classification Cup: Bagged Boosting". *SIGKDD Explorations. ACM SIGKDD*. 2000, Vol. 1, No. 2, pp. 65-66.

[5] I. Levin, "KDD-99 Classifier Learning Contest LLSoft's Results Overview". *SIGKDD Explorations. ACM SIGKDD*. 2000, Vol. 1, No. 2, pp. 67- 75.

[6] M. Vladimir, V. Alexei, and S. Ivan, "The MP13 Approach to the KDD'99 Classifier Learning Contest". *SIGKDD Explorations. ACM SIGKDD*. 2000, Vol. 1, No. 2, pp. 76-77.

[7] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data", in *Applications of Data Mining in Computer Security. Kluwer*, D. Barbara and S. Jajodia, ed., 2002.

[8] G. Kayacik, N. Zincir-Heywood, and M. Heywood, "On the Capability of an SOM based Intrusion Detection System", in *Proceedings of International Joint Conference on Neural Networks*, 2003.

[9] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*. The MIT Press. 1994.

[10] M. J. Zhang and V. Ciesielski. "Genetic programming for multiple class object detection", in *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence*, *Lecture Notes in Artificial Intelligence*, Heidelberg, December 1999, Vol. 1747, pp. 180–191.

[11] M. J. Zhang, and W. Smart. "Multiclass Object Classification Using Genetic Programming". *Technical Report CS-TR-04/2*, School of Mathematical and Computing Sciences, Victoria University, February 2004.

[12] T. Loveard, and V. Ciesielski. "Representing classification problems in genetic programming", in *Proceedings of the Congress on Evolutionary Computation*, Seoul, Korea, 2001, Vol. 2, pp. 1070-1077.

[13] B.-C. Chien, J. Y. Lin, and T. P. Hong, "Learning discriminant functions with fuzzy attributes for classification using genetic programming," *Expert Syst. Applicat.*, 2002, Vol. 23, pp. 31–37.

[14] R. R. F. Mendes, F. B.Voznika, A. A. Freitas, and J. C. Nievola, "Discovering fuzzy classification rules with genetic programming and co-evolution," in *Proc. 5th Eur. Conf. PKDD, Lecture Notes in Artificial Intelligence*, 2001, Vol. 2168, pp. 314–325.

[15] D. P. Muni, N.l R. Pal, and J. Das, "A Novel Approach to Design Classifiers Using Genetic Programming", *IEEE transactions on evolutionary computation*, April 2004, Vol. 8, No. 2, pp. 183-196.

[16] C. Mark and S. Gene, "Applying Genetic Programming Techniques to Intrusion Detection", in *Proceedings of the AAAI 1995 Fall Symposium*, November 1995.

[17] B. Adolf . *New Paradigms for Intrusion Detection Using Genetic Programming*. Technical report January 2004.

[18] C. M. G. Spafford. "Applying Genetic Programming to Intrusion Detection", in *Proceedings of the 18th NISSC Conference*, October 1998.

[19] R. A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Ann. Eugenics*, pt. II, 1936, Vol. 7, pp. 179–188.

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

[21] C. Elkan, "Results of the KDD'99 Classifier Learning", *SIGKDD Explorations, ACM* , January 2000, Vol.1, No. 2, pp. 63-64.

[22] W. Lee, and S. Stolfo. "A Framework for Constructing Features and Models for Intrusion Detection Systems", *Information and System Security*, 2000, Vol. 3, No. 4, pp. 227–261.

[23] M. Sabhnani, and G. Serpen, "Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context", in *Proceedings of the International Conference on Machine Learning, Models, Technologies and Applications (MLMTA 2003)*, Las Vegas, NV, June 2003, pp. 209-215.

[24] A. Küçükyılmaz, *Pattern Classification: A Survey and Comparison*. Department of Computer Engineering, Bilkent University, Ankara, Turkey. April 2005.

[25] A. K. Jain, R. P.W. Duin, and J. C. Mao, "Statistical pattern recognition: a review", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, January 2000, Vol. 22. No. 1, pp. 4-37.

[26] R. Agarwal, and M. V. Joshi, *PNrule: A New Framework for Learning Classifier Models in Data Mining*, Technical Report TR 00-015, Department of Computer Science, University of Minnesota, 2000.

BIOGRAPHY

Kamel Mohamed Faraoun earned a Master's degree in computer science at the Computer Science Department of Djilali Liabbes University- Sidi-Bel-abbes – Algeria in 2002. His current research areas include computer safety systems; genetic algorithms, fractal images compression evolutionary programming and grammatical inferences and physical materials structures modeling. He is currently a teacher at the Computer Sciences Institute of Djilali Liabess University, teaching operational research and human-machine interaction, and is preparing his Ph.D thesis in the field of computer security using artificial intelligence systems. He has published several papers in international journals.

Aoued Boukelif earned a Bachelor of Science in electrical engineering from the University of Pittsburgh and a Ph.D. (honours) degree in electrical engineering, image processing option. He is currently an assistant professor at the University of Sidi Bel Abbes and head of a research team dealing with information and communication technologies applied to distant learning. His main research areas include digital television, digital image compression, satellite communications information and communication technologies (ICTs). He is the author of several publications, including HDTV (Centre National des Etudes en Telecommunications, Paris, 1994), Digital Television Techniques (Masson and Paris, 1997), and Image Compression Techniques (Algiers, 2004).