MEPIDS: Multi-Expression Programming for Intrusion Detection System

Crina Groşan*, Ajith Abraham and Sang Yong Han

*Department of Computer Science
Babeş-Bolyai University, Kogălniceanu 1
Cluj-Napoca, 3400, Romania
School of Computer Science and Engineering
Chung-Ang University, Seoul, South Korea
cgrosan@cs.ubbcluj.ro, ajith.abraham@ieee.org, hansy@cau.ac.kr

Abstract. An Intrusion Detection System (IDS) is a program that analyzes what happens or has happened during an execution and tries to find indications that the computer has been misused. An IDS does not eliminate the use of preventive mechanism but it works as the last defensive mechanism in securing the system. This paper evaluates the performances of Multi-Expression Programming (MEP) to detect intrusions in a network. Results are then compared with Linear Genetic Programming (LGP) approach. Empirical results clearly show that genetic programming could play an important role in designing light weight, real time intrusion detection systems.

1 Introduction

An intrusion is defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource. Intrusion detection is classified into two types: misuse intrusion detection and anomaly intrusion detection [13]. Misuse intrusion detection uses well-defined patterns of the attack that exploit weaknesses in system and application software to identify the intrusions. These patterns are encoded in advance and used to match against the user behavior to detect intrusion. Anomaly intrusion detection uses the normal usage behavior patterns to identify the intrusion. The normal usage patterns are constructed from the statistical measures of the system features. The behavior of the user is observed and any deviation from the constructed normal behavior is detected as intrusion [7], [15]. Data mining approaches for intrusion detection were first implemented in mining audit data for automated models for intrusion detection [2], [6], [9]. Several data mining algorithms are applied to audit data to compute models that accurately capture the actual behavior of intrusions as well as normal activities. Audit data analysis and mining combine the association rules and classification algorithm to discover attacks in audit data. Soft Computing (SC) is an innovative approach to construct computationally intelligent systems consisting of artificial neural networks, fuzzy inference systems,

approximate reasoning and derivative free optimization methods such as evolutionary computation etc. [14]. This paper compares a Genetic Programming (GP) technique performance – Multi-Expression Programming (MEP) – with Linear Genetic Programming (LGP) [3], Support Vector Machines (SVM) [16] and Decision Trees (DT) [5]. Rest of the paper is organized as follows. Section 2 provides the technical details of MEP. In Section 3, a description of the intelligent paradigms used in experiments is given. Experiment results are presented in Section 4 and some conclusions are also provided towards the end.

2 Multi Expression Programming (MEP)

A GP chromosome generally encodes a single expression (computer program). By contrast, Multi Expression Programming (MEP)[11], [12] chromosome encodes several expressions. The best of the encoded solution is chosen to represent the chromosome (by supplying the fitness of the individual).

The MEP chromosome has some advantages over the single-expression chromosome especially when the complexity of the target expression is not known. This feature also acts as a provider of variable-length expressions. Other techniques (such as Gramatical Evolution (GE) [14] or Linear Genetic Programming (LGP) [4]) employ special genetic operators (which insert or remove chromosome parts) to achieve such a complex functionality.

2.1 Solution Representation

MEP genes are (represented by) substrings of a variable length. The number of genes per chromosome is constant. This number defines the length of the chromosome. Each gene encodes a terminal or a function symbol. A gene that encodes a function includes pointers towards the function arguments. Function arguments always have indices of lower values than the position of the function itself in the chromosome.

The proposed representation ensures that no cycle arises while the chromosome is decoded (phenotypically transcripted). According to the proposed representation scheme, the first symbol of the chromosome must be a terminal symbol. In this way, only syntactically correct programs (MEP individuals) are obtained.

An example of chromosome using the sets F= {+, *} and T= {a, b, c, d} is given below:

1: a
2: b
3: + 1, 2
4: c
5: d
6: + 4, 5
7: * 3, 6

The maximum number of symbols in MEP chromosome is given by the formula:

Number_of_Symbols = (n+1) * (*Number_of_Genes* – 1) + 1,

where n is the number of arguments of the function with the greatest number of arguments.

The maximum number of effective symbols is achieved when each gene (excepting the first one) encodes a function symbol with the highest number of arguments. The minimum number of effective symbols is equal to the number of genes and it is achieved when all genes encode terminal symbols only.

The translation of a MEP chromosome into a computer program represents the phenotypic transcription of the MEP chromosomes. Phenotypic translation is obtained by parsing the chromosome top-down. A terminal symbol specifies a simple expression. A function symbol specifies a complex expression obtained by connecting the operands specified by the argument positions with the current function symbol.

For instance, genes 1, 2, 4 and 5 in the previous example encode simple expressions formed by a single terminal symbol. These expressions are:

$E_1 = a$,

$E_2 = b$,

$E_4 = c$,

$E_5 = d$,

Gene 3 indicates the operation + on the operands located at positions 1 and 2 of the chromosome. Therefore gene 3 encodes the expression: $E_3 = a + b$. Gene 6 indicates the operation + on the operands located at positions 4 and 5. Therefore gene 6 encodes the expression: $E_6 = c + d$. Gene 7 indicates the operation * on the operands located at position 3 and 6. Therefore gene 7 encodes the expression: $E_7 = (a + b)$ * $(c + d)$. E_7 is the expression encoded by the whole chromosome.

There is neither practical nor theoretical evidence that one of these expressions is better than the others. This is why each MEP chromosome is allowed to encode a number of expressions equal to the chromosome length (number of genes). The chromosome described above encodes the following expressions:

$E_1 = a$,

$E_2 = b$,

$E_3 = a + b$,

$E_4 = c$,

$E_5 = d$,

$E_6 = c + d$,

$E_7 = (a + b)$ * $(c + d)$.

The value of these expressions may be computed by reading the chromosome top down. Partial results are computed by dynamic programming and are stored in a conventional manner.

Due to its multi expression representation, each MEP chromosome may be viewed as a forest of trees rather than as a single tree, which is the case of Genetic Programming.

2.2 Fitness assignment

As MEP chromosome encodes more than one problem solution, it is interesting to see how the fitness is assigned.

The chromosome fitness is usually defined as the fitness of the best expression encoded by that chromosome.

For instance, if we want to solve symbolic regression problems, the fitness of each sub-expression E_i may be computed using the formula:

$$f(E_i) = \sum_{k=1}^{n} |o_{k,i} - w_k|,$$

where $o_{k,i}$ is the result obtained by the expression E_i for the fitness case k and w_k is the targeted result for the fitness case k. In this case the fitness needs to be minimized.

The fitness of an individual is set to be equal to the lowest fitness of the expressions encoded in the chromosome:

When we have to deal with other problems, we compute the fitness of each sub-expression encoded in the MEP chromosome. Thus, the fitness of the entire individual is supplied by the fitness of the best expression encoded in that chromosome.

3 Intelligent Paradigms

3.1 Linear Genetic Programming (LGP)

Linear genetic programming is a variant of the GP technique that acts on linear genomes [4]. Its main characteristics in comparison to tree-based GP lies in that the evolvable units are not the expressions of a functional programming language (like LISP), but the programs of an imperative language (like c/c ++). An alternate approach is to evolve a computer program at the machine code level, using lower level representations for the individuals. This can tremendously hasten the evolution process as, no matter how an individual is initially represented, finally it always has to be represented as a piece of machine code, as fitness evaluation requires physical execution of the individuals.

3.2 Support Vector Machines

Support Vector Machines [16] have been proposed as a novel technique for intrusion detection. A Support Vector Machine (SVM) maps input (real-valued) feature vectors into a higher dimensional feature space through some nonlinear mapping. These are developed on the principle of structural risk minimization. SVM uses a feature called kernel to solve this problem. Kernel transforms linear algorithms into nonlinear ones via a map into feature spaces.

3.3 Decision Trees

Decision tree induction is one of the classification algorithms in data mining [5]. Each data item is defined by values of the attributes. The Decision tree classifies the given data item using the values of its attributes. The main approach is to select the attributes, which best divides the data items into their classes. According to the values of these attributes the data items are partitioned. This process is recursively applied to each partitioned subset of the data items. The process terminates when all the data items in current subset belongs to the same class.

4 Experiment Setup and Results

The data for our experiments was prepared by the 1998 DARPA intrusion detection evaluation program by MIT Lincoln Labs [10]. The data set has 41 attributes for each connection record plus one class label as given in Table 1. The data set contains 24 attack types that could be classified into four main categories Attack types fall into four main categories:

DoS: Denial of Service

Denial of Service (DoS) is a class of attack where an attacker makes a computing or memory resource too busy or too full to handle legitimate requests, thus denying legitimate users access to a machine.

R2L: Unauthorized Access from a Remote Machine

A remote to user (R2L) attack is a class of attack where an attacker sends packets to a machine over a network, then exploits the machine's vulnerability to illegally gain local access as a user.

U2Su: Unauthorized Access to Local Super User (root)

User to root (U2Su) exploits are a class of attacks where an attacker starts out with access to a normal user account on the system and is able to exploit vulnerability to gain root access to the system.

Probing: Surveillance and Other Probing

Probing is a class of attack where an attacker scans a network to gather information or find known vulnerabilities. An attacker with a map of machines and services that are available on a network can use the information to look for exploits.

Our experiments have two phases namely training and testing phases. In the training phase, MEP models were constructed using the training data to give maximum generalization accuracy on the unseen data. The test data is then passed through the saved trained model to detect intrusions in the testing phase. The 41 features are labeled as shown in Table 1 and the class label is named as *AP*.

This data set has five different attack types (classes) namely *Normal, DoS, R2L, U2R* and *Probes.* The training and test data comprises of 5,092 and 6,890 records respectively [8]. All the training data were scaled to (0-1). Using the data set, we performed a 5-class classification.

Table 1. Variables for intrusion detection data set

Variable No.	Variable name	Variable type	Variable label
1	duration	continuous	A
2	protocol_type	discrete	B
3	service	discrete	C
4	flag	discrete	D
5	src_bytes	continuous	E
6	dst_bytes	continuous	F
7	land	discrete	G
8	wrong_fragment	continuous	H
9	urgent	continuous	I
10	hot	continuous	J
11	num_failed_logins	continuous	K
12	logged_in	discrete	L
13	num_compromised	continuous	M
14	root_shell	continuous	N
15	su_attempted	continuous	O
16	num_root	continuous	P
17	num_file_creations	continuous	Q
18	num_shells	continuous	R
19	num_access_files	continuous	S
20	num_outbound_cmds	continuous	T
21	is_host_login	discrete	U
22	is_guest_login	discrete	V
23	count	continuous	W
24	srv_count	continuous	X
25	serror_rate	continuous	Y
26	srv_serror_rate	continuous	X
27	rerror_rate	continuous	AA
28	srv_rerror_rate	continuous	AB
29	same_srv_rate	continuous	AC
30	diff_srv_rate	continuous	AD
31	srv_diff_host_rate	continuous	AE
32	dst_host_count	continuous	AF
33	dst_host_srv_count	continuous	AG
34	dst_host_same_srv_rate	continuous	AH
35	dst_host_diff_srv_rate	continuous	AI
36	dst_host_same_src_port_rate	continuous	AJ
37	dst_host_srv_diff_host_rate	continuous	AK
38	dst_host_serror_rate	continuous	AL
39	dst_host_srv_serror_rate	continuous	AM
40	dst_host_rerror_rate	continuous	AN
41	dst_host_srv_rerror_rate	continuous	AO

The settings of various linear genetic programming system parameters are of utmost importance for successful performance of the system [1]. The population size was set at 120,000 and a tournament size of 8 is used for all the 5 classes. Crossover and mutation probability is set at 65-75% and 75-86% respectively for the different classes. Our trial experiments with SVM revealed that the polynomial kernel option often performs well on most of the datasets. We also constructed decision trees using the training data and then testing data was passed through the constructed classifier to classify the attacks [13]. Parameters used by MEP are presented in Table 2. We made use of +, - , *, /, sin, cos, sqrt, ln, lg, $\log_2$, min, max, and abs as function sets.

Table 2. Parameters used by MEP

Attack type	Parameter value				
	Pop. Size	Generations	Crossover (%)	No. of mutations	Chromosome length
Normal	100	100	0.9	3	30
Probe	200	200	0.9	4	40
DOS	500	200	0.8	5	40
U2R	20	100	0.9	3	30
R2L	800	200	0.9	4	40

Table 3. Performance comparison

Attack type	Classification accuracy on test data set (%)			
	MEP	DT	SVM	LGP
Normal	**99.82**	99.64	99.64	99.73
Probe	95.52	99.86	98.57	**99.89**
DOS	98.91	96.83	99.92	**99.95**
U2R	**99.75**	68.00	40.00	64.00
R2L	**99.72**	84.19	33.92	99.47

In Table 4 the variable combinations evolved by MEP are presented. Results presented in Table 3 are based on these evolved functions.

As evident from Table 3, MEP gives the best results for detecting Normal patterns, U2R and R2L attacks. While DT, SVM and LGP did not perform well U2R attacks, MEP obtained the best results for this class (99.76% accuracy). Results for MEP presented in this table are obtained by applying the test data using the training function which gave the best results during training.

In Figure 1, the classification accuracy for the best results obtained for training data, results obtained for the test data using the best training function and

Fig. 1. Evolutionary learning of the different attack types

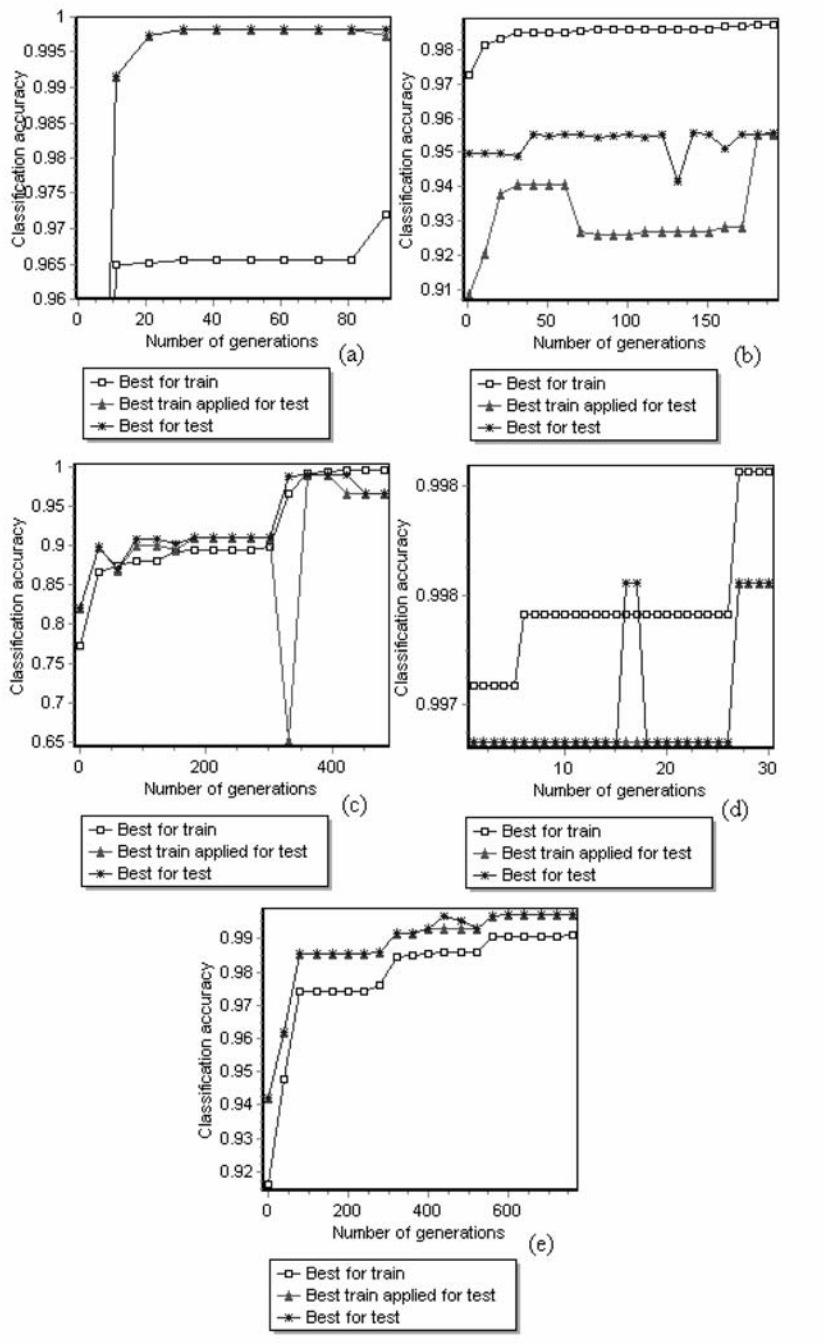

Table 4. MEP evolved functions for the attack classes

Attack type	Evolved Function
Normal	var12 * log_2(var10 + var3)
Probe	(log_2(var2) < (fabs((var36 * var27) > (var27 + var35 – var34) ? (var35 * var27) : (var27 + var35 – var34))) ? ($\log_2$(var2)) : (fabs((var36 * var27) > (var27 + var35 – var34) ? (var36 * var27) : (var27 + var35 – var34)));
DOS	$0.457*(var8+(ln(var6))*(lg(var41))--var40+var23+var8)$
U2R	$sin(var14) - -var33$
R2L	0.36 + (var11 < 0.086 ? var11 : 0.086 + 0.086) > (var6 > ($\log_2$($\log_2$(var12 * var3))) ? var6 : ($\log_2$($\log_2$(var12 * var3)))) ? (var11 < 0.086 ? var11 : 0.086 + 0.086) : (var6 > ($\log_2$($\log_2$(var12 * var3))) ? var6 : ($\log_2$($\log_2$(var12 * var3)))) + var6

the best results obtained for the test data are depicted. Figure 1 (a) corresponds to Normal patterns, Figure 1 (b) corresponds to Probe, Figure 1 (c) corresponds to DOS Figure 1 (d) corresponds to U2R and Figure 1 (e) corresponds to R2L respectively.

In some classes the accuracy figures tend to be very small and may not be statistically significant, especially in view of the fact that the 5 classes of patterns differ in their sizes tremendously. For example only 27 data sets were available for training the U2R class. More definitive conclusions can only be made after analyzing more comprehensive sets of network traffic.

5 Conclusions

In this paper, we have illustrated the importance of genetic programming based techniques for modeling intrusion detection systems. MEP outperforms LGP for three of the considered classes and LGP outperform MEP for two of the classes. MEP classification accuracy is grater than 95% for all considered classes and for four of them is greater than 99.65%. It is to be noted that for real time intrusion detection systems MEP and LGP would be the ideal candidates because of its simplified implementation.

6 Acknowledgements

This research was supported by the MIC (Ministry of Information and Communication), Korea, under the Chung-Ang University HNRC-ITRC (Home Network

Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).

References

1. Abraham A., Evolutionary Computation in Intelligent Web Management, Evolutionary Computing in Data Mining, Ghosh A. and Jain L.C. (Eds.), Studies in Fuzziness and Soft Computing, Springer Verlag Germany, Chapter 8, pp. 189-210, 2004.
2. Barbara D., Couto J., Jajodia S. and Wu N., ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection. SIGMOD Record, 30(4), pp. 15-24, 2001.
3. Brameier. M. and Banzhaf. W., A comparison of linear genetic programming and neural networks in medical data mining,Evolutionary Computation," IEEE Transactions on, Volume: 5(1), pp. 17-26, 2001.
4. Brameier M. and Banzhaf W, Explicit control of diversity and effective variation distance in Linear Genetic Programming. In Proceedings of the fourth European Conference on Genetic Programming, Springer-Verlag Berlin, 2001.
5. Brieman L., Friedman J., Olshen R., and Stone C., Classification of Regression Trees. Wadsworth Inc., 1984.
6. Cohen W., Learning Trees and Rules with Set-Valued Features, American Association for Artificial Intelligence (AAAI), 1996.
7. Denning D., An Intrusion-Detection Model, IEEE Transactions on Software Engineering, Vol. SE-13, No. 2, pp.222-232, 1987.
8. KDD Cup 1999 Intrusion detection data set: http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10_percent.gz
9. Lee W. and Stolfo S. and Mok K., A Data Mining Framework for Building Intrusion Detection Models. In Proceedings of the IEEE Symposium on Security and Privacy, 1999.
10. MIT Lincoln Laboratory. http://www.ll.mit.edu/IST/ideval/
11. Oltean M. and Grosan C., A Comparison of Several Linear GP Techniques, *Complex Systems*, Vol. 14, No. 4, pp. 285-313, 2004.
12. Oltean M. and Grosan C., Evolving Evolutionary Algorithms using Multi Expression Programming. Proceedings of The 7^{th} European Conference on Artificial Life, Dortmund, Germany, pp. 651-658, 2003.
13. Peddabachigari S., Abraham A., Thomas J., Intrusion Detection Systems Using Decision Trees and Support Vector Machines, International Journal of Applied Science and Computations, USA, Vol.11, No.3, pp.118-134, 2004.
14. Ryan C. et al, 1998. Gramatical Evolution:Evolving programs for an arbitrary language. In Proceedings of the first European Workshop on Genetic Programming, Springer-Verlag, Berlin
15. Summers R.C., Secure Computing: Threats and Safeguards. McGraw Hill, New York, 1997.
16. Vapnik V.N., The Nature of Statistical Learning Theory. Springer, 1995.