Programming Risk Assessment Models for Online Security Evaluation Systems

Ajith Abraham[1], Crina Grosan[1 2], Vaclav Snasel [1 3]
[1]Machine Intelligence Research Labs, MIR Labs, http://www.mirlabs.org
[2]Babes-Bolyai University, Cluj-Napoca, Romania
[3]VSB-Technical University of Ostrava, Czech Republic

Abstract

Risk assessment is often done by human experts, because there is no exact and mathematical solution to the problem. Usually the human reasoning and perception process cannot be expressed precisely. This paper propose a genetic programming approach for risk assessment. Preliminary results indicate that genetic programming methods are robust and suitable for this problem when compared to other risk assessment models.

1 Introduction

Risk analysis [8][11][12]is fundamentally all about establishing probabilities. Different people have different opinions about risk and the association of its dependent variables and in our previous research we have used fuzzy logic to model risk [6]. In the DIPS framework [6], the risk analysis was done using *threat levels*, *vulnerability* and *asset value* [5]. We consider that all components within a network scenario falls into one of these categories, and each has attributes, or derived factors, that contribute positively or negatively to risk.

Haslum et al. [6] used a fuzzy logic approach to model the risk assessment problem in intrusion detection systems. Further the authors [7] used a neuro-fuzzy learning method to optimize the performance of the fuzzy risk models.

Genetic programming techniques have been successfully applied to various security problems [1], [2], [4]. In this paper, we approach the risk assessment problem using a variant of genetic programming named multi-expression programming, which is presented in Section 3. In order to test the performances of genetic programming on risk assessment problem we performed some experiments as illustrated in Section 4. Results obtained by the genetic programming approach are compared with a hybrid fuzzy controller approach [6]. A set of conclusions are drawn towards the end of the paper.

2 Risk Assessment

Different people have different opinions about risk and the association of its dependent variables, and various soft computing tools provide an excellent framework to model risk. In our previous research [6], the risk analysis was modeled using *threat levels*, *vulnerability* and *asset value* [5].

Threat level is modeled as the frequency of attacks/intrusions, the probability that an intruder is being successful in overcoming protective controls and gains access to act against the organization or assets and the type and severity of attacks.

Vulnerability may be defined as the probability that an asset will be unable to resist the actions of an intruder. Vulnerability exists when this probability exceeds a given threshold. This may be because of weaknesses in software or hardware, missing software patches and so on. Vulnerability is modeled as (1) threat capability and (2) system threat resistance.

Asset may be defined as any data, device, or other component of the environment that supports information-related activities, which can be affected in a manner that result in loss. To determine asset loss could be one of the hardest tasks of analyzing risk. It is very difficult to put a precise value on the various types of assets, and there may be more than one value or liability characteristic. Complex relationships might exist between the different forms of loss and many factors determine loss magnitude. Asset value/loss is modeled using four variables: (1) cost (2) criticality (3) sensitivity and (4) recovery.

The overall architecture for asset risk management is summarized in Figure 1 [6].

3 Multi-Expression programming

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

978-0-7695-3593-7/09 $25.00 © 2009 IEEE
DOI 10.1109/UKSIM.2009.75

Authorized licensed use limited to: Wikipedia. Downloaded on June 22,2024 at 15:19:12 UTC from IEEE Xplore. Restrictions apply.

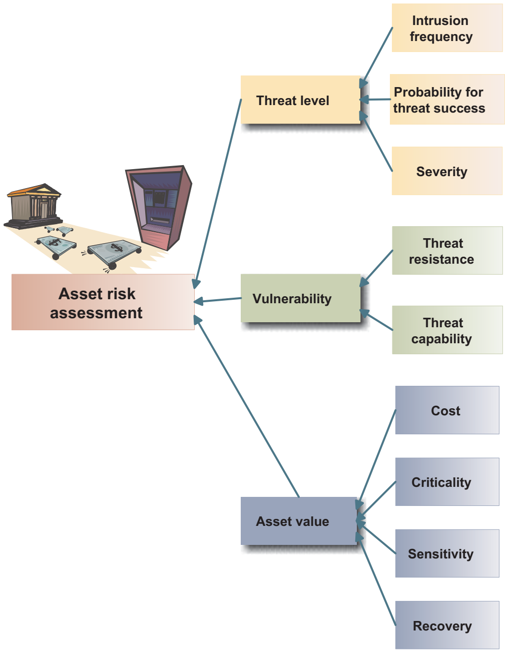


Figure 1. Generic structure of the risk assessment model

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 Grammatical Evolution (GE) [13] or Linear Genetic Programming (LGP) [3]) employ special genetic operators (which insert or remove chromosome parts) to achieve such a complex functionality. Multi Expression Programming (MEP) technique ([9], [10]) description and features are presented in what follows.

3.1 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 transcribed). 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 example, 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: $E3$ = $a + b$. Gene 6 indicates the operation + on the operands located at positions 4 and 5. Therefore gene 6 encodes the expression: $E6$ = $c + d$. Gene 7 indicates the operation * on the operands located at position 3 and 6. Therefore gene

7 encodes the expression: $E7$ = $(a + b)$ * $(c + d)$. $E7$ 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:

$E_1 = a$,
$E_2 = b$,
$E_3 = a + b$,

Authorized licensed use limited to: Wikipedia. Downloaded on June 22,2024 at 15:19:12 UTC from IEEE Xplore. Restrictions apply.

$E_4 = c$,
$E_5 = d$,
$E_6 = c + d$,
$E_7 = (a + b) * (c + d)$.

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.

3.2 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.

4 Experimental Results

We used the same network model as illustrated in Figure 2 to have performance comparisons [6]. The sample network consists of four different assets; a router, a public web server, a file server, and a database. Five IDS Agents denoted by $IDS_1, \ldots, IDS_5$ are deployed in the network.The attack category used for the risk assessment is based on inputs from the IDS agents and this value is used to assign values to eight of the nine input variables. All the 9 input variable values and the output variable (risk assessment) are scaled between 0-1.

4.1 Results obtained by MEP

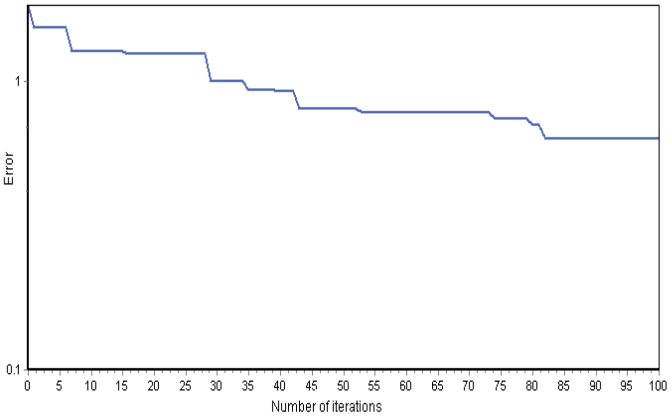
Parameters used by MEP are presented in Table 1. The best and worst results obtained in 20 independent runs are displayed in Table 2. In order to compare the MEP performance, we considered the results obtained by the Hierarchical Neuro-Fuzzy Learning approach (HiNFRA) [6].

Some more detailed explanations related to MEP results are provided as follows: the errors obtained by the best and worst individuals are depicted in Figures 3 and 4. Figures 5-8 depict the comparison between target and obtained results for both training and test data obtained in the best and in the worst runs.

Table 1. Parameters used by MEP

Parameter	Value
Number of iterations	100
Population size	20
Crossover probability	0.8
Number of mutations per chromosome	2
Functions used	+. -, *, /, sin, cos, tan, exp, ln, log, pow

Figure 3. The evolution of the error for the training data set obtained in the worse run.



The MEP program obtained by the best individual is as follows:

(cos(exp(1 / (log10(x[5])) + x[0]))) * (tan(1 / (exp(cos((tan(x[5])) * x[0])))));

The MEP program obtained by the worse individual is as follows:

sin((cos(x[7] - (sin(tan(x[0]) + fabs(x[0]))))) * (sin(pow(1 / tan(cos((fabs(x[0])) * x[5])), 2))));

It is to be noted that, not all 9 variables are required for building the MEP based risk assessment models. For example, the best individual used only 3 input variables, while the worst individual required just 2 input variables. So the MEP approach also helps in considerable feature reduction, which is very important for building online, light risk assessment models that can be implemented in mobile sensors etc.

Compared to the Neuro-fuzzy model, the developed GP model also has a comparable Root Mean Squared Error (RMSE) for the test data set. The best program obtained

Authorized licensed use limited to: Wikipedia. Downloaded on June 22,2024 at 15:19:12 UTC from IEEE Xplore. Restrictions apply.

Figure 2. Experimental network for risk assessment

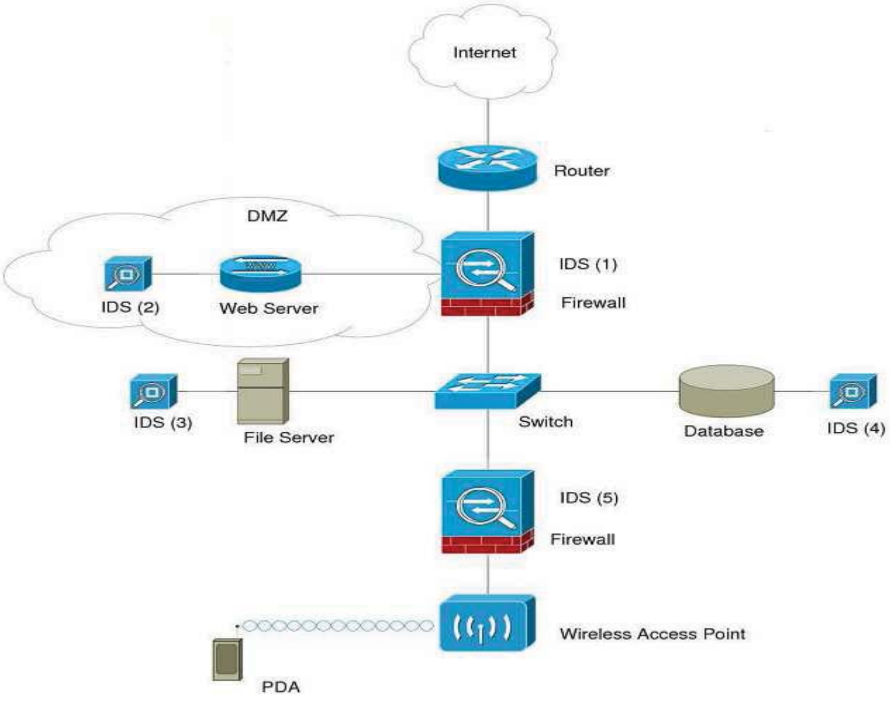


Figure 4. The evolution of the error for the training data set obtained in the best run.

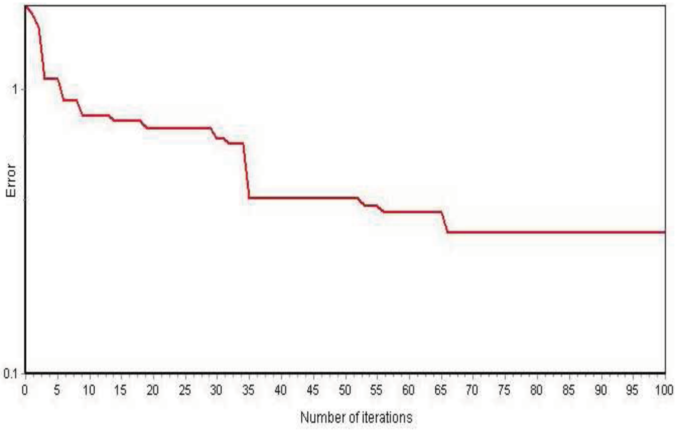
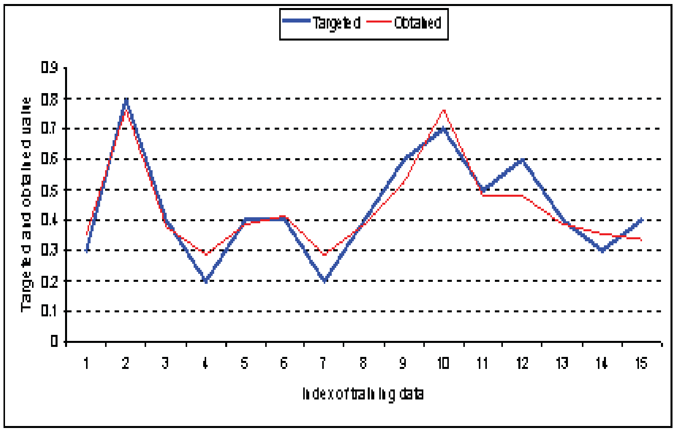


Figure 5. The results (targeted and obtained) for the training data obtained in the worse run.



Authorized licensed use limited to: Wikipedia. Downloaded on June 22,2024 at 15:19:12 UTC from IEEE Xplore. Restrictions apply.

Figure 6. The results (targeted and obtained) for the test data obtained in the worse run.

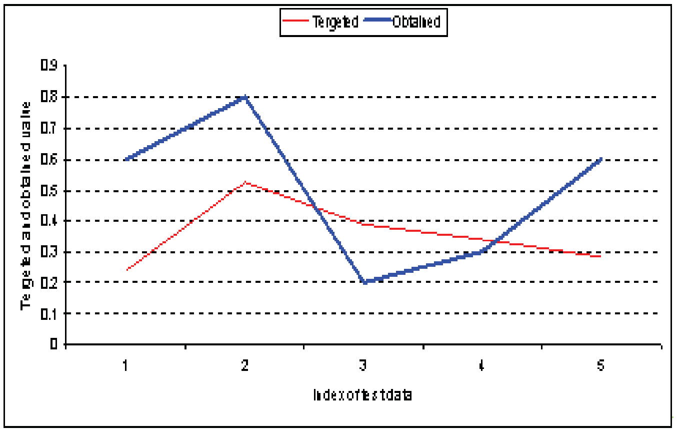


Figure 7. The results (targeted and obtained) for the training data obtained in the best run.

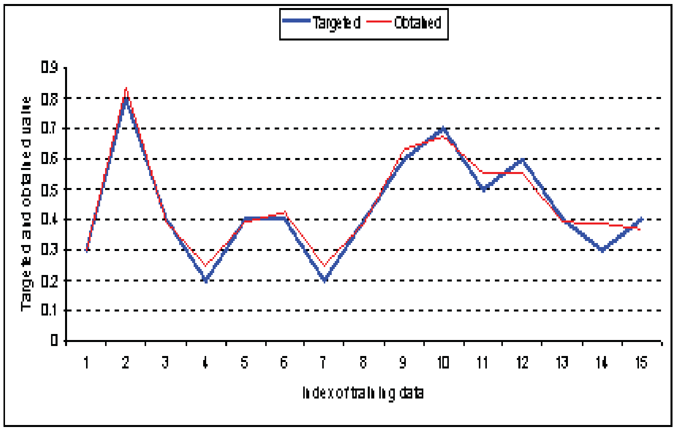
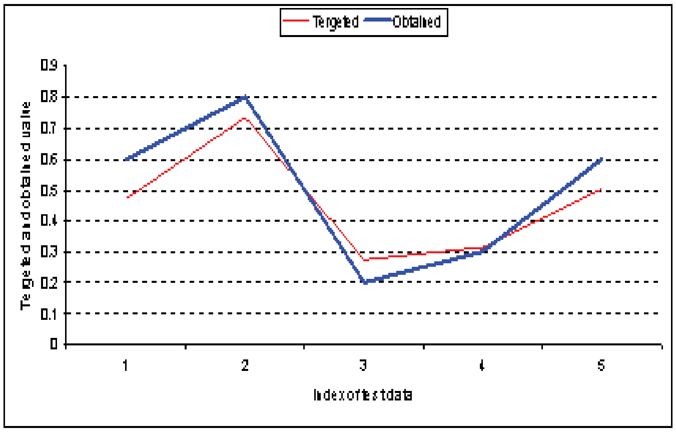


Table 2. The values of error and root mean square error (RMSE) obtained in the best and worse runs for training and test data. Results obtained by HiNFRA.

Data	**MEP**		**HiNFRA**
	Error	RMSE	RMSE
	Worse run		
Training data	0.7438	0.0690	
Test data	0.4854	0.6794	
	Best run		
Training data	0.4616	0.0385	0.0593
Test data	0.3781	0.1896	0.3169

Figure 8. The results (targeted and obtained) for the test data obtained in the best run.



is obtaining better results than the neuro-fuzzy approach which makes it very suitable for the risk assessment problem.

5 Conclusions

This paper presented a detailed implementation of Genetic Programming based Risk Assessment Model (GePRA) to aid the decision making process. The proposed model has been illustrated in an intrusion detection system. Preliminary results indicate that genetic programming approach could work very well for the risk assessment problem. Compared to our previous [6] research, where the fuzzy model was mainly based on expert knowledge, the implementation of GePRA is fairly easy and obtains a more simple, light weight (just few lines of codes) and an adaptive risk assessment model.

References

[1] A. Abraham and C. Grosan, Evolving Intrusion Detection Systems, Genetic Systems Programming, Nadia Nedjah et al. (Eds.), Studies in Computational Intelligence, Springer Verlag Germany, ISBN: 3-540-29849-5, pp. 57-79, 2006

[2] A. Abraham, C. Grosan and C. Martin-Vide, Evolutionary Design of Intrusion Detection Programs, International Journal of Network Security, Vol.4, No.3, pp. 328-339, 2007

[3] Brameier M. and Banzhaf W, Explicit control of diversity and effective variation distance in Linear Genetic Programming. In Proceedings of the fourth European

Authorized licensed use limited to: Wikipedia. Downloaded on June 22,2024 at 15:19:12 UTC from IEEE Xplore. Restrictions apply.

Conference on Genetic Programming, Springer-Verlag Berlin, 2001.

[4] C. Grosan, A. Abraham and S. Y. Han, MEPIDS: Multi-Expression Programming for Intrusion Detection System, International Work-conference on the Interplay between Natural and Artificial Computation, (IWINAC'05), Spain, Lecture Notes in Computer Science, LNCS 3562, J. Mira and J.R. Alvarez (Eds.), Springer Verlag, Germany, pp. 163-172, 2005

[5] J. Jones. An introduction to factor analysis of information risk (fair). Norwich Journal of Information Assurance , 2(1):67, 2006.

[6] K. Haslum, A. Abraham and S. Knapskog, HiNFRA: Hierarchical Neuro-Fuzzy Learning for Online Risk Assessment, Second Asia International Conference on Modeling and Simulation, AMS 2008, IEEE Computer Society Press, USA, ISBN 978-0-7695-3136-6, pp. 631-636, 2008

[7] K. Haslum, A. Abraham and S. Knapskog, Fuzzy Online Risk Assessment for Distributed Intrusion Prediction and Prevention Systems, Tenth International Conference on Computer Modeling and Simulation, UKSiM/EUROSiM 2008, Cambridge, UK, IEEE Computer Society Press, USA, ISBN 0-7695-3114-8, pp. 216-223, 2008.

[8] D. J. Landoll, The Security Risk Assessment Handnook: A Complete Guide for Performing Security Risk Assessments, Taylor & Francis, 2006

[9] Oltean M. and Grosan C., A Comparison of Several Linear GP Techniques, Complex Systems, Vol. 14, No. 4, pp. 285-313, 2004.

[10] Oltean M. and Grosan C., Evolving Evolutionary Algorithms using Multi Expression Programming. Proceedings of The 7 thEuropean Conference on Artificial Life, Dortmund, Germany, pp. 651-658, 2003

[11] AI. Passori, Selecting the Risk Assessment Method of Choice, META Group, July 21, 2004. http://searchcio.techtarget.com/originalContent/0,289142, sid19_gci994851,00.html

[12] T.R. Peltier, Information Security Risk Analysis, pp. 2346. Auerbach Publications, 2001.

[13] C. Ryan C. J.J. Collins and M. ONeill. Gramatical Evolution: Evolving programs for an arbitrary language, In Proceedings of the first European Workshop on Genetic Programming, Springer-Verlag, Berlin, 1998

Authorized licensed use limited to: Wikipedia. Downloaded on June 22,2024 at 15:19:12 UTC from IEEE Xplore. Restrictions apply.