CHAOS MULTIOBJECTIVE EVOLUTIONARY BASED TECHNIQUE TO OBTAIN ACCURATE SOLUTIONS IN INTRUSION DETECTION SYSTEMS

Sunitha Guruprasad *
Department of Computer Science and Engineering,
St Joseph Engineering College, Mangaluru, Karnataka, 575028, India.
sunithag@sjec.ac.in

Rio D'Souza G. L.
Department of Computer Science and Engineering,
St Joseph Engineering College, Mangaluru, Karnataka, 575028, India.
riod@sjec.ac.in

Abstract - Due to the rapid increase in technology, it becomes necessary to safeguard the system against any sort of external attacks. Intrusion detection system (IDS) plays a vital role in protecting the system against any sort of anomalous behavior. Whenever the system is prone to attack, it becomes difficult to identify the type of attack, due to different security policies of the network. In our research, we have developed a chaos multiobjective framework that can be trained for different objectives and obtain solutions for each combination of objectives. This allows the end user to select the best solution among the broad range of solutions. Experiments were conducted on NSL-KDD, ISCX-2012 and CICIDS2017 datasets. The results obtained shows that Chaos-Nondominated Sorting Genetic Programming (chaos-NSGP-II) algorithm produces better spread of solutions compared to the existing frameworks, Nondominated Sorting Genetic Programming (NSGP-II) and Multiobjective Evolutionary Algorithms Based on Decomposition (MOEA/D). Chaos theory was used to control the population size, improve the convergence speed and to avoid falling into local optima.

***Keywords*:** Intrusion detection; Genetic programming; NSL-KDD; ISCX-2012; CICIDS2017.

1. Introduction

Computer systems and networks are more prone to attacks from anomalous users. Ordinary firewalls may not be able to detect all the attacks. IDS is used to detect such attacks and inform the users about any malicious activity. The main task of the IDS is to discriminate between the legitimate and non-legitimate events on the system. Various machine learning techniques like fuzzy system, soft computing, evolutionary algorithm and computational intelligent systems have been used by researchers to find solutions to the problem. Among these techniques, evolutionary computing based techniques are applied successfully in the areas of optimization, bioinformatics, autonomous programming and many more, due to their nature of scaling well to higher dimensional problems.

The major limitation of any evolutionary based techniques is that it takes a very long time to reach the optimum solution. Also, since the population size is kept constant throughout the evolution, there are chances of occurrence of inbreeding among the individuals. Different variants of evolutionary methods have been proposed by some researchers to control the population size. In our work, we have used a technique similar to the work of [Nelson et al. (2010)] to control the population size. By using different population sizes at each generation, the probability of generating better solutions can be increased to a greater extent.

In real world, there exists many problems that contains several competing objectives and involves simultaneous optimization. In single objective optimization, an attempt is made to obtain a best solution. In multiple objective optimizations, a set of solution exists where every solution will be different from the other. Users will have the option of selecting the best solution based on the requirements. In recent years, Multiobjective Evolutionary Algorithms (MOEAs) are the most widely used technique in various fields due to their power in solving difficult problems with relative ease especially in the field of intrusion detection. An attempt has been made by many researchers to tackle the problems having multiple conflicting objectives. [Mukhopadhyay et al. (2013a)] and [Mukhopadhyay et al. (2013b)] have surveyed several multiobjective evolutionary algorithms for different data mining tasks. They have listed different combination of objectives that represent trade-off between multiple conflicting objectives. A pareto-based multiobjective evolutionary based algorithm has been proposed by

[Gómez et al. (2013)] to optimize automatic rule generation in IDS. The main aim of their research was to minimize the false positive rate and false negative rate of the classifiers. [Wang et al. (2014)] have proposed a method for maximizing the ROC curve. Four different multiobjective algorithms were used in their work for experimentation purpose. They have shown that NSGA-II algorithm performs better than the other three algorithms. An ensemble of Artificial Neural Network based IDS was created by [Kumar et al. (2012)] to increase the detection rate of the classifiers. They have used a new metric called CID (Intrusion detection Capability) which considers the true positive rate, false positive rate and base rate collectively. The main aim of [Wang et al. (2014)] and [Kumar et al. (2012)] was to maximize the true positive rate and minimize the false positive rates. [Elhag et al. (2019)] have proposed an algorithm in which the multiobjective evolutionary algorithm is integrated within a linguistic fuzzy association rule mining. They have conducted various studies to select the best combination of objectives from various combinations. They have shown that the combination of minimum FMeasure and false alarm rate gives best results compared to all other combinations.

When there are multiple conflicting objectives in a problem, it becomes difficult for the end-users to select the solutions required for their work, from the pool of solutions. In such cases, it becomes necessary to create a framework that gives a broad set of solutions that satisfies the requirements of all the users. Our work is similar to the method proposed by [Elhag et al. (2019)]. But we have used different combination of metric and algorithm since the datasets and the methods used in our algorithm are different. In our research, we have used a chaos-NSGP-II algorithm that is first trained for different objectives. This increases the search space for the generation of varied range of solutions for each combination of objectives. The combination that gives the best results are later used to compare the proposed method with the state-of-the-art methods, NSGP-II and MOEA/D. The organization of the paper is as follows: Section 2 provides the steps of the proposed framework, information about fast nondominated sorting method, chaos function and the metrics used in the work. Section 3 gives the description of the datasets and the parameters used. Section 4 shows the analysis of the best combination of objectives and the comparison with the existing methods and Section 5 concludes the paper.

2. Methodology

2.1. *Proposed Framework*

Chaos-NSGAII, just like all evolutionary based algorithms, focuses on creating a population of individuals and then perform evolution for evolving better individuals. The main parameters of any GP algorithms are Crossover, Mutation and Reproduction. During the crossover operation, changes are made to the individuals such that it creates new individuals that includes the best features of both the parents. During mutation operation, a slight modification is done to any feature of the individual by randomly replacing any node of the tree with a new one. During reproduction, the best individuals of one generation are copied to the next generation.

The procedure of the proposed framework is given below:

Step 1: Set the size of the population S, parameter for growth r, initial variable x0, and crossover and mutation rate.

Step 2: Generate the initial population randomly and evaluate the fitness of each individual. Step 3: Apply crossover and mutation operations on the individuals and create new offsprings. Step 4: If the termination criteria is reached, return the best population

Step 5: Combine the parent population and child populations and create a combined population R of size 2S.
Step 6: Identify the nondominated fronts in R using fast nondominated sorting algorithms.

Step 7: Calculate the crowding distance of the solutions and choose the best solutions of size S.

Step 8: Apply crossover, mutation and selection operators on the best solutions obtained. Step 9: Perform Chaos operation:

Step 9.1: Apply logistic map function using:

$$x_{i+1} = r * x_i * (1 - x_i)$$

Step 9.2: Find the population size of the next generation:

Newpopsize = round ($x_{i+1} * S$)

Step 10: Increment the generation count and go to step 4.

2.2. *Algorithm details*

2.2.1. *Population Initialization*

The initial population in any genetic programming consists of individuals/trees which can be created using either grow, full or ramped-half-and-half methods. The terminal nodes in grow method will have the same depth whereas in full method the depth might change. Both full and grow methods are used in ramped-half-and-half method. In our research, we have used the ramped-half-and-half method to create the initial population. The

trees generated consists of nodes with different depth.

The trees are generated using a set of terminals (t1, t2, …, tn) and set of functions (f1, f2, …, fn). The terminal nodes are leaf nodes which consists of named variables or constants and the functions are internal nodes and consists of fixed operands or arguments. The different operations that can be applied on the trees can be either arithmetic, logical, boolean, or any function specific to the domain. In our work, we have used arithmetic operators to create the expressions of the individuals. Real numbers are generated as the output. The output which generates a positive value is taken as a normal class and the negative value output is taken as an attack class.

2.2.2 *Fast nondominated sorting*

In nondominated sorting, each solution is compared with every solution in the population to check for dominance. The time taken to find the nondominated fronts is $O(MN^3)$. In order to reduce the computation, a fast nondominated sorting is used which requires only $O(MN^2)$ computations. Here, every solution is checked only with the partial population that is filled. For example, suppose the first solution in a population is kept in a set P. Each remaining solution, p is compared with all the members of P. If a member q of P is dominated by p, then q is removed from the set. If any member of P dominates p, then p is ignored. If p is not dominated by any member of P, it will be entered into P. After all the solutions are checked, the remaining members of P forms the nondominated set. The process is repeated until all the fronts are found.

2.2.3 *Chaos function*

Chaos is a theory that mainly focuses on nonlinear dynamic behavior. It is highly sensitive to initial conditions and avoids getting trapped into local optima and maintains population diversity. The main properties of chaos systems like stochastic, regularity and ergodicity, helps in getting accurate solutions after few iterations. The properties are expressed by chaotic maps. The different examples of chaotic maps are, Tent map, Henon map, Gaussian map, Logistic map, etc.

One of the simplest logistic equations is a logistic Map. The expression of the logistic map is shown in equation (1). x_i is a chaotic variable which contains values between 0 and 1 and represents the ratio of the maximum population and the current population.

$$x_{i+1} = r * x_i * (1 - x_i) \quad (1)$$

2.2.4 *Metrics for IDS*

Performance metrics plays a major role in classifying the performance of the IDS as normal or intrusive. Many such metrics exists, but no single metric can effectively measure the performance of the IDS. Few of the metrics available are Precision (Prec), FMeasure (F1), False Negative Rate (FNR), True Positive Rate (TPR) and False Positive Rate (FPR). It is very difficult to determine the best IDS using only a single metric. Most of the objectives exhibit conflicting behavior. Researchers have used many such metrics to compare the behavior of their work with the state-of-the-art. In our work we have analyzed the behavior of the metrics used by these researchers to select the best combination of metrics that exhibit good overall performance.

The different metrics used in our research are:

(1) Precision: stands for the number of correctly classified positive instances over the total number of positive instances.

$$Prec = TP/(TP+FP) \quad (2)$$

(2) FMeasure: stands for the harmonic mean or tradeoff between recall and precision.

$$F1 = 2.\ Recall.Prec\ /(Recall+Prec) \quad (3)$$

(3) False Negative Rate: stands for the false negatives divided by the total number of true negatives and false negatives.

$$FNR = FN/(TN+FN) \quad (4)$$

(4) True Positive Rate: stands for the number of correctly classified positive instances over the total number of correct instances.

$$TPR = TP/(TP+FN) \quad (5)$$

(5) False Positive Rate: stands for the number of incorrectly classified positive instances over the total number of negative instances.

$$FPR = FP/(FP+TN) \quad (6)$$

where, TP, TN, FP and FN stands for the number of true positives, true negatives, false positives and false negatives of a class.

3. Experimental Settings

In this section we describe the different datasets and the parameters used in our research. The algorithm of the Chaos-NSGAII method was implemented in Python 2.7. Windows 7 operating system and i3 Intel core CPU were used to conduct the experiments.

3.1. *Description of Datasets*

In our work, NSL-KDD, ISCX-2012 and CICIDS2017 datasets were used to conduct the experiments.

NSL-KDD dataset by [Tavallaee et al. (2009)] generated by MIT's Lincoln laboratory was mainly created to solve some of the problems of the existing KDD cup 1999 dataset. The dataset may not give the exact representation of the actual working of the network, but can be used for comparison with the methods used by many researchers. The dataset contains the necessary data required for training and testing. In our work, we have used 18 features out of 41 features and two classes, Normal and Attack out of the actual five classes.

The ISCX-IDS-2012 (Information Security Center of excellence) dataset by [Shiravi et al. (2012)] contains data similar to the real network traffic. It contains seven days of network traces and 19 features and 2 classes. In our work, we have used the data of Monday's dataflow and used 11 features and two classes, Normal and Attack.

The datasets of ISCX-2012 dataset does not have traces of HTTPS protocol. A new dataset CICIDS2017 dataset by [Sharafaldin et al. (2018)] contains the pcap format of the real network flow and also the up-to-date attacks with full packet payloads. It contains the data collected for a week and has 15 classes and 84 features. In our work, we used the data of Wednesday's dataflow and used 25 features and two classes, Normal and Attack.

3.2. *GP Parameters*

The performance of GP algorithms depends on the parameters set for the system. Table 1 represents the parameters used in the proposed system. The algorithm was run many times with different set of parameters and the one which gave the optimum solution was selected.

Table 1. Parameters set for GP.

Format symbol	Descriptio
Population size	500
Generations	50 or 100% accuracy
Method of population	Ramped half-and-half
Depth of the tree	5
Functions	/, *, -, +
Terminals	Variables and Constants
Crossover	0.9
Mutation	0.05
Reproduction	Tournament selection
Size of tournament	2

4. Experiments conducted

The most difficult task in any multiobjective evolutionary based algorithms is to find the combination of objectives that determine the tradeoff between the objectives. Our experiments are composed of two parts. First, analysis of the best combination of objectives that shows the trade-off between objectives. Second, the combination that gives the best results are compared with the existing methods.

4.1. *Analysis of the best combination of objectives*

The first step of our work was to identify the objectives that conflict with each other, analyze them and check which combination gives the best results. The experimental results of the different combination of objectives are given in tables 2 to 5. The digits are printed in bold-face if the results obtained are significantly better than that of the other variants of the same datasets.

The first combination of objectives shown in Table 2 are false positive rate and false negative rate [Gómez et al. (2013)] with the aim of minimizing the number of false positives and false negatives respectively. The second combination of objectives shown in Table 3 are true positive rate and false positive rate [Wang et al. (2014)] and [Kumar et al. (2012)] where the aim was to maximize the true positive rate and minimize the false positive rate. The third combination of objectives shown in Table 4 are FMeasure and false positive rate [Elhag et al. (2019)] where the aim was to maximize the FMeasure and minimize the false positive rate. The fourth combination of objectives shown in Table 5 are Precision and true positive rate where the aim was to maximize the Precision and true positive rate. Standard accuracy was not used in our analysis since it is not considered as a suitable metric for most of the classification problems [Wang et al. (2014)], [Kumar et al. (2012)].

Table 2. Results of FNR vs FPR.

Metrics/Datasets			Prec	F1	FNR	TPR	FPR
NSL-KDD	Best FNR	Train	96.05	95.21	0.35	93.46	0.312
		Test	95.36	94.77	**0.53**	**92.54**	0.356
	Best FPR	Train	97.32	96.78	0.43	89.72	0.054
		Test	**96.11**	**95.31**	0.54	88.13	**0.071**
ISCX-2012	Best FNR	Train	96.33	95.32	0.31	93.79	0.178
		Test	**95.42**	94.13	**0.45**	92.11	0.211
	Best FPR	Train	95.72	96.54	0.47	94.07	0.105
		Test	94.03	**95.31**	0.50	**93.82**	**0.132**
CICIDS2017	Best FNR	Train	94.78	96.50	0.30	94.85	0.141
		Test	94.51	**96.11**	**0.35**	93.72	0.172
	Best FPR	Train	95.65	95.91	0.39	95.00	0.082
		Test	**95.00**	95.60	0.43	**94.23**	**0.099**

Table 3. Results of F1 vs FPR.

Metrics/Datasets			Prec	F1	FNR	TPR	FPR
NSL-KDD	Best F1	Train	95.43	96.21	0.37	92.35	0.212
		Test	94.81	**95.32**	**0.42**	**91.47**	0.273
	Best FPR	Train	96.28	95.10	0.48	88.55	0.096
		Test	**95.71**	94.27	0.51	85.72	**0.132**
ISCX-2012	Best F1	Train	96.12	96.25	0.43	93.82	0.295
		Test	**95.38**	**95.70**	**0.51**	92.71	0.310
	Best FPR	Train	94.62	95.82	0.58	94.22	0.111
		Test	93.17	94.11	0.62	**93.91**	**0.215**
CICIDS2017	Best F1	Train	94.88	95.84	0.44	95.89	0.091
		Test	**95.92**	**95.41**	0.51	**94.37**	0.123
	Best FPR	Train	95.13	95.72	0.33	94.22	0.052
		Test	94.72	94.99	**0.43**	93.78	**0.083**

Table 4. Results of TPR vs FPR.

Metrics/Datasets			Prec	F1	FNR	TPR	FPR
NSL-KDD	Best TPR	Train	95.07	92.84	0.53	90.81	0.055
		Test	**93.50**	**91.90**	0.62	**90.70**	0.076
	Best FPR	Train	92.51	91.13	0.38	79.82	0.022
		Test	91.32	90.61	**0.42**	78.46	**0.051**
ISCX-2012	Best TPR	Train	96.71	96.00	0.45	94.90	0.072
		Test	**96.42**	96.78	**0.48**	**94.73**	0.088
	Best FPR	Train	93.21	96.89	0.49	93.87	0.054
		Test	93.00	**96.11**	0.53	92.36	**0.085**
CICIDS2017	Best TPR	Train	96.93	94.52	0.41	93.73	0.087
		Test	**96.20**	**94.15**	**0.45**	**93.52**	0.091
	Best FPR	Train	94.17	92.26	0.59	91.30	0.042
		Test	95.37	93.75	0.65	92.61	**0.064**

Table 5. Results of Prec vs TPR.

Metrics/Datasets			Prec	F1	FNR	TPR	FPR
NSL-KDD	Best Prec	Train	96.05	93.81	0.41	93.44	0.132
		Test	**95.72**	93.13	0.49	**93.68**	0.156
	Best TPR	Train	95.80	95.23	0.37	93.51	0.100
		Test	94.31	**94.72**	**0.44**	92.13	**0.112**
ISCX-2012	Best Prec	Train	95.95	95.68	0.60	94.17	0.230
		Test	**95.48**	**95.08**	0.63	**93.76**	0.257
	Best TPR	Train	94.88	94.66	0.51	93.94	0.199
		Test	94.61	94.21	**0.57**	93.19	**0.210**
CICIDS2017	Best Prec	Train	95.54	94.08	0.33	94.56	0.150
		Test	**95.10**	93.77	**0.41**	94.10	0.192
	Best TPR	Train	95.26	94.99	0.38	95.63	0.099
		Test	94.63	**94.76**	0.49	**95.25**	**0.101**

Among all the combinations that we experimented, we found that the combinations of maximizing the true positive rate and minimizing the false positive rate gives best results for most of the metric. The next combination of objectives that gave better results were FMeasure and false positive rate. It is also interesting to note that when false positive rate is used with most of the combinations, it gives varied results.

4.2. *Comparison with the existing methods*

The best combination of objectives selected by comparison of different objectives were used to analyze the effectiveness of the approach with the methods available in literature. The original NSGP-II and MOEA/D were used for comparison with the Chaos-NSGP-II methods. Tables 6 to 8 shows the performance of the three datasets with different metrics. It can be seen that the performance of NSGP-II was better than that of MOEA/D in almost all the cases. The main reason is that NSGP-II ranks the individuals and performs elitism using crowding distance at each generation. Also, the performance of Chaos-NSGP-II was seen to be better than the NSGP-II. The performance of Chaos-NSGP-II mainly depends on the new individuals that are created when the population size is reduced at some generation. When a new individual is created, we mainly concentrate on the maintaining those individuals that gives better performance. This in turn resulted in increasing the performance of the overall algorithm.

Table 6. Results of Chaos-NSGP-II (TPR vs FPR) with NSGP-II and MOEA/D for NSL-KDD dataset

Metrics/Methods		Prec	F1	FNR	TPR	FPR
Chaos-NSGP-II	Train	95.07	92.84	0.53	90.81	0.055
	Test	93.50	91.90	0.62	90.70	0.076
NSGP-II	Train	94.75	91.65	0.67	89.91	0.087
	Test	93.11	90.84	0.73	88.12	0.082
MOEA/D	Train	94.15	90.99	0.83	89.33	0.169
	Test	93.03	89.34	0.92	87.76	0.102

Table 7. Results of Chaos-NSGP-II (TPR vs FPR) with NSGP-II and MOEA/D for ISCX-2012 dataset

Metrics/Methods		Prec	F1	FNR	TPR	FPR
Chaos-NSGP-II	Train	96.71	96.00	0.45	94.90	0.072
	Test	96.42	96.78	0.48	94.73	0.088
NSGP-II	Train	95.62	95.76	0.53	94.62	0.101
	Test	95.75	95.24	0.69	94.20	0.152
MOEA/D	Train	95.10	95.11	0.46	94.00	0.128
	Test	94.83	94.58	0.41	93.62	0.156

Table 8. Results of Chaos-NSGP-II (TPR vs FPR) with NSGP-II and MOEA/D for CICIDS2017 dataset

Metrics/Methods		Prec	F1	FNR	TPR	FPR
Chaos-NSGP-II	Train	96.93	94.52	0.41	93.73	0.087
	Test	96.20	94.15	0.45	93.52	0.091
NSGP-II	Train	95.79	93.85	0.58	93.42	0.099
	Test	95.45	93.25	0.59	93.06	0.116
MOEA/D	Train	94.87	93.23	0.64	92.76	0.145
	Test	94.62	92.69	0.59	92.62	0.134

5. Conclusion

The requirement of the user might change based on the type of attack on the system. A good IDS should not only be able to identify the intrusive behavior, but also be able to exhibit optimized trade-offs between multiple conflicting objectives. In our research, we have proposed a Chaos-NSGP-II algorithm which generates a pool of solutions based on the different requirements of the user. Chaos theory helps in generating new strong individuals by controlling the population size at each generation. The contribution of our work is summarized below:

(1) We have conducted several experiments for combination of objectives during the training process.

(2) The best combination of objectives obtained was used for comparison with the existing methods.

(3) Chaos-theory was used to vary the population size in every generation to avoid in-breeding of the individuals.

We tested our algorithm using NSL-KDD, ISCX-2012 and CICIDS2017 datasets and obtained good trade-off between maximizing the true positive rate and minimizing the false positive rate for most of the experiments.

References

[1] Elhag, S., Fernández, A., Altalhi, A., Alshomrani, S. and Herrera, F., 2019. A multi-objective evolutionary fuzzy system to obtain a broad and accurate set of solutions in intrusion detection systems. Soft Computing, 23(4), pp.1321-1336.

[2] Gómez, J., Gil, C., Baños, R., Márquez, A.L., Montoya, F.G. and Montoya, M.G., 2013. A Pareto-based multi-objective evolutionary algorithm for automatic rule generation in network intrusion detection systems. Soft Computing, 17(2), pp.255-263.

[3] Kumar, G. and Kumar, K., 2012. The use of multi-objective genetic algorithm based approach to create ensemble of ann for intrusion detection.

[4] Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S. and Coello, C.A.C., 2013. A survey of multiobjective evolutionary algorithms for data mining: Part I. IEEE Transactions on Evolutionary Computation, 18(1), pp.4-19.

[5] Mukhopadhyay, A., Maulik, U., Bandyopadhyay, S. and Coello, C.A.C., 2013. Survey of multiobjective evolutionary algorithms for data mining: Part II. IEEE Transactions on Evolutionary Computation, 18(1), pp.20-35.

[6] Nelson, T.H., 2010. Genetic Algorithms with Chaotic Population Dynamics.

[7] Shiravi, A., Shiravi, H., Tavallaee, M. and Ghorbani, A.A., 2012. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. computers & security, 31(3), pp.357-374.

[8] Sharafaldin, I., Lashkari, A.H. and Ghorbani, A.A., 2018, January. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In ICISSP (pp. 108-116).

[9] Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A.A., 2009, July. A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE symposium on computational intelligence for security and defense applications (pp. 1-6). IEEE.

[10] Wang, P., Tang, K., Weise, T., Tsang, E.P.K. and Yao, X., 2014. Multiobjective genetic programming for maximizing ROC performance. Neurocomputing, 125, pp.102-118.