International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-5, March 2020

Evolutionary Method of Detecting Intrusions using Different Population Dynamics

Sunitha Guruprasad, Rio D'Souza G. L.

Abstract*: Chaos theory plays a vital role in any evolutionary based algorithms for avoiding the local optima and to improve the convergence speed. Various researchers have used different methods to increase the detection rate and to speed up the convergence. Some researchers have used evolutionary algorithms for the same purpose and has proved that the application of those algorithms provide good results. Most of the researchers have used population sizes which remains constant throughout the evolution. It has been seen that small population size may result in premature convergence and large population size requires more computation time to find a solution. In this paper, a novel application of different population dynamics to the genetic programming (GP) algorithm has been applied to manage the population size. The main focus was to improve the accuracy of the normal GP algorithm by varying the population sizes at each generation. The experiments were conducted on the standard GP algorithm using static and dynamic population sizes. Different population dynamics has been used to check the effectiveness of the proposed algorithm. The results obtained has shown that dynamic population size gives better results compared to static population size and also solves the problem of local optima.

***Keywords*: Detection rate, Genetic programming, Intrusion detection, population dynamics, NSL-KDD, ISCX-2012, CICIDS2017.**

I. INTRODUCTION

With the emergence of Internet and the dependence of business applications on it, network security has become a major issue. Traditional techniques such as firewalls, anti-virus and spyware, cannot prevent unauthorized access to the data effectively. A more effective technique such as intrusion detection system (IDS) is needed to control the access of information at the network level.

The main challenge of any intrusion detection system is to distinguish the genuine activity from the abnormal activity. Various machine learning methods like evolutionary based algorithms, Bayes theorem, etc., has been used by different researchers to tackle the issue. Among the evolutionary based algorithms, the most common ones are Genetic algorithm (GA) and Genetic programming (GP). Both algorithms use the evolutionary operations like crossover, mutation and reproduction to get better solutions. But the main difference lies in the way the data is interpreted. In GA, every individual is interpreted as a string of characters. But in GP, every individual is interpreted as a tree and hence gives far better results compared to GA.

GP has been used successfully by many researchers for solving the classification problems of improving the accuracy of the classifiers. Jabeen et al. [1] has proposed a method that increases the accuracy of the classifiers with less number of function evaluations using a single GP classifier. It also optimizes the evolved intelligent structures using Particle Swarm Optimization technique. Dufourq et al. [2] proposed four GP models which hybridizes GA with GP algorithm. The results obtained, showed that the ensemble methods improved the performance of the classifiers.

In the field of Intrusion detection, researchers have used various methods to increase the accuracy of the classifiers and to minimize the false alarm rate. A classifier called GPSVM created by Pozi et al. [3] uses GP and support vector machine to create the classifier. Experiments were conducted on rare anomalous attacks and the classifier reported high detection rate. Wa'el et al. [4] presented a new hybrid model using Rough Set Classification and Parallel GP (RSC-PGP). The model was used to identify important features to build the intrusion detection system. It also decreased the training time and increased the convergence speed of RSC. The model was also able to produce robust and better set of rules. The features selected was able to detect new attack with great reduction in data, time and error. Another method proposed by Le, Thi, et al. [5], uses GP for detecting malwares collected form internet database. The results obtained from the method generated good results on both the imbalanced and balanced datasets.

Population size plays an important role in any evolutionary based algorithms. Most of the algorithms uses a population size which remains constant throughout the evolution. Researchers argue that small population size may result in premature convergence with increasing probability of obtaining a local optimum. A large population size involves more computation time in finding a solution with increased probability of generating a global optimum. Selection of appropriate population size is important which greatly improves the effectiveness of the algorithm.

In literature, researchers have proposed several variants of evolutionary methods for obtaining an optimal population size for different perspectives. The first successful one was GAVaPS, proposed by Arabas et al. [6], where a lifespan was assigned to each individual based on fitness. The lifetime notation was extended by Back et al.

Revised Manuscript Received on March 30, 2020.
***** Correspondence Author
Sunitha Guruprasad*, Department of Computer Science and Engineering, St Joseph Engineering College, Mangaluru, India. Email: sunithag@sjec.ac.in
Rio D'Souza G. L., Department of Computer Science and Engineering, St Joseph Engineering College, Mangaluru, India. Email: riod@sjec.ac.in

[7] to steady-state GA in Genetic Algorithm which uses adaptive population size (APGA). Various other methods like Evolutionary Algorithm for self-regulating the population size (SRP-EA) by Fernandes and Rosa [8], altering the population size using the selection error probability method by Smith [9], evolving many populations of various sizes in parallel by Harik and Lobo [10] were proposed.

Unlike these methods, in this paper we have used a method similar to the work done by Nelson [11]. Rather than keeping the population size constant, different population size can be used at each generation. This increases the possibility of generating better individuals at every generation. This is done using a chaotic population control method. A number of performance tests were done by comparing the static and chaotic methods. It was observed that chaotic method gives good results compared to the static population size method, even though both perform in the same way in certain situations.

The rest of the paper is organized as follows: an overview of chaotic system is given in section II, section III explains the proposed chaotic-GP model in detail, section IV shows the settings of the experiment and the datasets used and in section IV the results are discussed and compared.

II. CHAOTIC SYSTEM

Chaos is an interdisciplinary theory that focuses on deterministic dynamic behavior and is highly sensitive to initial conditions. A very small change may lead to different search paths which enhances the searching capability of any algorithm. Chaos systems have stochastic, regularity and ergodicity properties which results in getting arbitrary accuracy after few iterations. These properties can be expressed by chaotic maps, which maintains not only the population diversity, but also avoids getting trapped in local optimum. Chaotic maps can be one-dimensional or two-dimensional. We can implement different variants of the algorithm, by applying these maps to the algorithm. Few examples of chaotic maps are, Logistic map, Tent map, Gaussian map, Henon map, etc.

Logistic Map is a polynomial mapping of degree two. It is the simplest form of logistic equations. The mathematical expression of the map is shown in (1), where $x_i \in (0,1)$ is a chaotic variable which constitutes the ratio of the current population to the highest population.

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

The logistic map function is applied repeatedly for some initial value x_0, to get a new point x_{i+1}. When the initial variable $x_0 \neq \{0, 0.25, 0.5, 0.75\}$, x_i behaves chaotically.

$r \in (0,1)$ acts as a control or a growth parameter. Varying the growth parameter, gives different behavior patterns. The population goes extinct and eventually dies when r is chosen between the values 0 and 1. The population approaches the value (r-1)/r quickly when r is chosen between 1 and 2. When r is between 2 and 3, the population first oscillates around the value and ultimately approaches the value (r-1)/r. When r is beyond 3, the population approaches permanent oscillations between two values. The population oscillates among 4 values when r is chosen between 3.45 and 3.55 and among 8 values when r increases beyond 3.55. The behavior behaves chaotic when $r \approx 3.57$ or above. When r is equal to 4 the system will be in complete chaos and when r increases beyond 4, the initial values diverges.

The bifurcation diagram in Figure 1, shows the summarized pattern of logistic maps behavior. The possible values of growth parameter r is shown in x-axis and the population size x is shown in y-axis. The diagram shows all chaotic and non-chaotic behavior for different values of r. For growth rates between 3.5 and 4.0, it shows different values for each generation. For growth rate around 3.2, the system oscillates between two discrete paths. At around 3.4, it bifurcates into four paths.

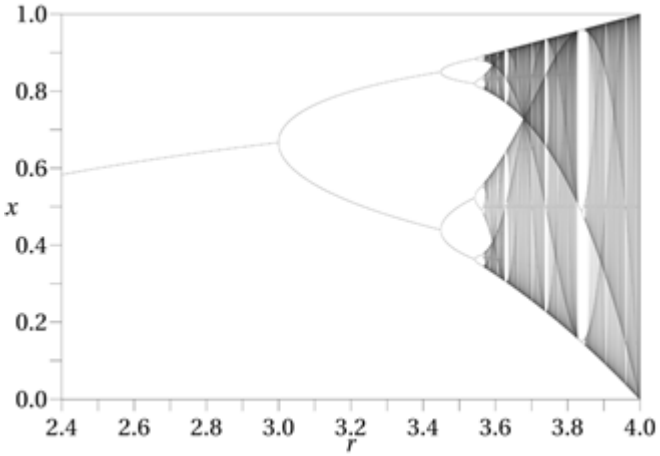


Fig. 1. Logistic map-Bifurcation diagram

III. PROPOSED MODEL

In this section, the different steps followed in this research is explained. The initial step involves reducing the features of the dataset and normalizing the values. The later steps involve, using a GP based method to generate and evolve the individuals using crossover, mutation and reproduction operators and using a logistic map function to control the population size.

A. Selection of Features and Normalization

Most of the features in the datasets may not be favorable for classification purpose and may not result in getting high accuracy. The features will have to be reduced in order to get high classification accuracy. Reduced set of features not only results in developing better models but also in avoiding over fitting of the problem.

Many researchers have developed different algorithms to reduce the features of NSL-KDD dataset. Algorithms like "Flexible Mutual Information Feature Selection" [12] and another method which was based on information gain [13] were developed successfully to create a reduced set of features. Both the methods achieved low false positive rate and good detection rate with 22 features. Another method proposed by [14] used the union of two algorithms called Relief and OneR to construct a reduced feature set.

There method obtained good accuracy for 12 features. In our work, the experiments were performed using 18 features which were selected as the best features by different researchers.

TABLE I: Features of NSL-KDD, ISCX-IDS-2012 and CICIDS2017 datasets

Dataset	Features Extracted From Database	No Of Features
NSL_Kdd	Srv.SerrorRate, Dst.HostSrvSerrorRate, Dst.HostSerrorRate, Dst.HostDiffSrvRate, Dst.HostSameSrvRate, Dst.HostSrvCount, Diff.SrvRate, Same.SrvRate, Serror.Rate, Count, Root.Shell, Logged.In, Hot, Dst.Bytes, Src.Bytes, Flag, Service, Duration	18
ISCX-IDS-2012	Destination.port, Source.port, Protocol.name, Destination.tcpflagsdescription, Source.tcpflagsdescription, Direction, Total.sourcepackets, Total.destinationpackets, Total.destinationbytes, Total.sourcebytes, App.name	11
CICIDS2017	Total.LengthOfBwdPackets, Fwd.PacketLengthMin, Bwd.PacketLengthMin, Bwd.PacketLengthStd, Flow.Iat.Mean, Flow.IatMin, Fwd.IatMin, Bwd.IatTotal, Bwd.IatMean, Bwd.IatStd, Bwd.IatMin, Fwd.PacketsS, Bwd.PacketsS, MinPacketLength, Packet.LengthVariance, Psh.FlagCount, AckFlagCount, DownUp. Ratio, Average.PacketSize, Avg.FwdSegmentSize, Subow.FwdBytes, Init.WinBytesForward, Init.WinBytesBackward, Active.Mean, Idle.Min	25

For ISCX-IDS-2012 dataset, the experiments were performed using 11 features out of 19 features. The payload and the date features were eliminated and only the relevant features were extracted from the dataset.

For the CICIDS2017 dataset, the experiments were performed using 25 features out of 84 features. These features were selected based on the features used in [15] and [16]. The methods obtained good results for the reduced set of features.

Table I show the list of features selected for the datasets. All the datasets were preprocessed and normalized before it is used. Preprocessing involved, removing redundant data, conversion of symbolic data and combining all the abnormal classes into single class. The missing values were replaced with zeros and the infinite values by the mean value of that attribute. Since the values of the attributes varied between 0 and infinity, a normalization equation was used to normalize and scale the values between the range 0 and 1:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (2)$$

x is the value of a particular attribute in the dataset.

B. Framework of Genetic Programming

The algorithm of the proposed Chaotic-GP framework consists of the following steps:

Step 1: Create an initial population of arithmetic expressions called individuals. These individuals are trees generated randomly for a specified population size. Each individual is evaluated using a fitness function and the best individuals are selected for reproduction.

Step 2: A chaotic map operation is applied to determine the population size for the further generations. Crossover and mutation operations are applied on the remaining individuals to generate new offspring's.

Step 3: The process is repeated for a predefined number of generations or until the optimum solution is converged. The best individuals are transferred into a solution set.

The detailed procedure of the proposed Chaotic-GP model is shown in the Algorithm.

Algorithm: Modified GP model

Input: A training dataset, testing dataset
Output: Individuals with the best fitness

1. Set the population size N_s, growth parameter r, initial variable x_0, crossover rate and mutation rate.
2. Generate an initial population P of individuals randomly with population size N_s.
3. Evaluate the individuals of each individual using a fitness measure.
4. Copy the individual with the highest fitness to the new population.
5. Perform Chaotic Map operation:
   a. Apply the logistic mapping using the formula:
      $x_{i+1} = r * x_i * (1 - x_i)$
   b. Determine the population size for the next generation:
      Nextpopsize = round ($x_{i+1} * N_s$)
6. Apply crossover operation on the individuals:
   a. Randomly choose four individuals from the population. Run a tournament and select the best two individuals as parents.
   b. Apply a crossover point on the parents and swap the individuals.
   c. Evaluate the fitness of the children.
   d. Compare the fitness of the parents and the children.
   e. Select the best two individuals from the family based on the fitness.
7. Apply mutation operation on the individuals:
   a. Select a mutation point.
   b. Randomly create a node or a function.
   c. Replace the subtree with the newly generated node from the mutation point.
8. Evaluate the fitness of the individuals and check whether the termination criteria is satisfied or the required number of generations are explored.
   If yes: Go to step 9.
   Else: Repeat steps 4 to 8 and continue the process.
9. Return the individuals comprising the best fitness as the solution.

C. Algorithm details

1) Initializing the population

In genetic programming, the initial population can be generated using any of the commonly used methods like full, grow or ramped-half-and-half methods. In grow method, the terminal nodes are expected to have the same depth, whereas in full method, the depth might vary. In ramped-half-and-half method, the initial population is generated using both full and grow methods. In our work, a ramped-half-and-half method is used to construct the initial population. The trees generated using this method will have different size and shapes.

Individuals in the tree are built using a set of functions F (f1to fn) and a set of terminals T (t1 to tn). The functions consists of a fixed number of arguments or operands and are used as internal nodes of a tree. The various operations or functions that can be applied to the trees can be logical, mathematical, arithmetic, Boolean, iterative, conditional or any other functions suitable for the problem or specific to the domain. The terminal set consists of constants like Ephemeral Random constants (ERC's) or named variables or 0-arity functions and forms the leaf nodes of the tree.

In this research, the population consists of individuals which are expressions using arithmetic operators and trained to create the classifiers needed to distinguish between the classes. The classifier generates real numbers as outputs. The output which gives a positive value is indicated as a positive class and that which gives a negative value is indicated as a negative class.

2) Fitness functions

Each individual's performance is evaluated using a fitness function assigned to each individual. Different performance metrics used in our work for evaluation are false positive rates, Detection rate and Accuracy which are given in equations (3) to (5).

Accuracy or fitness of an individual specifies the number of instances correctly classified as positive or negative. Accuracy is evaluated using the equation:

$$Accuracy = \frac{Total\ no\ of\ instances\ correctly\ classified}{Total\ no\ of\ instances} * 100$$

(3)

Detection rate (DR) specifies the number of instances classified correctly by the positive instances. DR is evaluated using the equation:

$$DR = \frac{Total\ no\ of\ instances\ correctly\ classified}{Total\ no\ of\ correct\ instances} * 100$$

(4)

False positive rate (FPR) specifies the no of incorrect instances misclassified as correct instances. FPR is evaluated using the equation:

$$FPR = \frac{Total\ no\ of\ misclassified\ instances}{Total\ no\ of\ negative\ instances} * 100$$

(5)

3) Genetic operators

The individuals in the population undergoes reproduction, crossover and mutation operations during the process of evolution. The operators are used to create better individuals in the future generations.

Crossover operation creates a change of population by combining the best features of the parents and generates offspring's. Four individuals are picked randomly from the population. A tournament is run and the best two individuals are selected as the parents. A crossover point is applied on both the parents and the part of the individuals after the crossover point is interchanged. A fitness function is used to evaluate the new individuals and compared with the fitness of the parents. The best two individuals from the whole family is selected as the individuals for the future generation.

Mutation operation is used to create a change in the individual. Initially, a function or a node is selected as a mutation point from any individual which is randomly selected. The part of the tree from the mutation point is replaced by the node or function created separately.

Reproduction operation is used to copy the best individual in each generation to the next generation without making any changes in the individual. For the individuals having the same fitness values, the individual with the least depth is selected for reproduction.

IV. EXPERIMENTAL SETTINGS

The algorithm of the Chaotic-GP method was implemented in Python 2.7. The experiments were conducted in an i3 Intel core CPU and windows 7 operating system.

A. Dataset used

The experimental data were taken from NSL-KDD, ISCX-2012 and CICIDS2017 datasets.

NSL-KDD dataset presented by Tavallaee, M, et al [17] was generated by MIT's Lincoln laboratory to resolve some of the intense problems of the existing KDD cup 1999 dataset like redundant and duplicate data in the dataset. The dataset does not give the perfect representation of actual network, but is used by many researchers for comparison of different intrusion detection methods. The dataset contains different datasets for training and testing purposes. The dataset contains 41 features and five classes. Attack class contains data of four different attack scenarios. In our work, we have combined all the attack classes into single class and used only two classes Normal and Attack.

The ISCX-IDS-2012 (Information Security Center of excellence) presented by Shiravi, et al. [18] is an evaluated intrusion detection dataset which consists of seven days of real network instances (normal and attack). It consists of pcap format of network traces labeled including full packet payload. It contains 19 features and five classes. Attack class contains four different attack scenarios. The dataset contains normal and malicious data collected for seven days. In our work, we have used the dataflow of Monday's data and used only two classes, Normal and Attack.

Even though the ISCX-2012 dataset consists of realistic network traffic, it does not have traces of HTTPS protocol. The CICIDS2017 dataset presented by Sharafaldin, et al.

[19] consists of the most common up-to-date attacks and the pcap format of labeled network flows, including full packet payloads. It contains 84 features and 15 classes. The dataset contains data collected for 5 days from Monday to Friday. Network traffic of Monday contains benign data and the traffic of Tuesday to Friday contains applied attacks. In our work, we used the dataflow of Wednesday's data and used only two classes, Normal and Attack. Attack class contains 15 different types of attacks. All the datasets and the corresponding profiles are publicly available for download through their websites. The details of the reduced set of features is listed in Table I and the dataset distributions utilized in our work is shown in Table II.

Table- II: Distributions of NSL-KDD, ISCX-IDS-2012 and CICIDS2017 datasets

Datasets	Training instances	Testing Instances	Classes (Actual)	Features (Actual)
NSL-KDD	25192	11850	2 (5)	18 (41)
ISCX-IDS-2012	171380	51414	2 (5)	11 (19)
CICIDS2017	692704	207811	2 (15)	25 (84)

B. Parameters of GP

The performance of the algorithm depends on the parameters that are set for the system. The parameters used in the proposed system is represented in Table III. The system was run many times with different parameters and the one which gives the optimum solution is represented in the table.

Table- III: Parameters of GP

GP Parameters	Value
Initial Population Size	100 and 500
Maximum Generations	100% training accuracy or 50
Population method	Ramped-half-and-half method
Tree Depth	5
Function set	+, -, *, / (1 for division by zero)
Terminal set	Attributes, Ephemeral Constants
Selection for reproduction	Tournament
Tournament size	2
Crossover operator	0.9
Mutation operator	0.05

V. RESULT AND DISCUSSION

The main goal of our work was to investigate the effect of varying the population size to check the efficiency of GP in detecting intrusions. The algorithm was evaluated on NSL-KDD, ISCX-2012 and CICIDS2017 datasets. Rather than checking the performance of the genetic programming algorithm with different values of r, the analysis was restricted to only the three different dynamics: one with a constant population, one with the varying population and one with the chaotic population. To check the effect of population size, we ran the algorithm with low and high population sizes separately. The population sizes were taken as 100 and 500 respectively. The algorithm was run for 200 generations. The initial value of x was taken as 0.4. It was observed that, for higher population size, the growth was inconsistent initially, but improved faster. The reason for inconsistency was due to more number of less fit organisms that were generated in the initial generations. The later generations produced good results due to the evolution process.

Next, we tested the algorithm with different population dynamics for both the low and high population sizes. The results are shown in Table IV. The table shows the population sizes and the mean fitness value comparison between genetic programming algorithm and static, periodic and chaotic population dynamics for the three datasets. The algorithm was run ten times for each of the population dynamics and the mean value calculated.

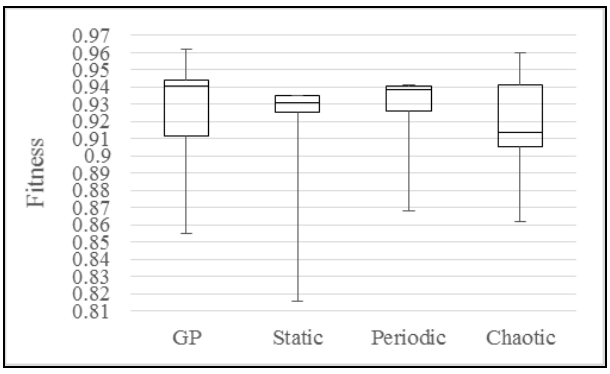
Table- IV: Mean fitness values for the different population

Dataset	Population size	GP	Static (r=2.8)	Periodic (r=3.5)	Chaotic (r=3.9)
NSL-KDD	100	0.9255	0.9243	**0.9304**	0.9151
	500	0.9383	0.9360	0.9397	**0.9414**
ISCX-IDS-2012	100	0.9262	0.9123	**0.9324**	0.9243
	500	0.9373	0.9360	0.9421	**0.9454**
CICIDS2017	100	0.9287	0.9255	0.9256	**0.9314**
	500	0.9402	0.9383	0.9437	**0.9461**

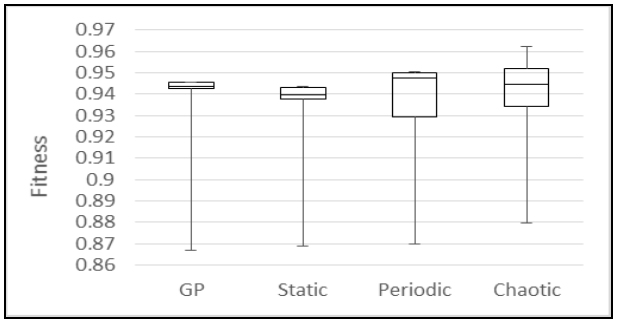
Figures 2 to 4 shows the boxplots of the fitness distributions for different values of r (population dynamics) and the normal GP method for the three datasets.

NSL-KDD

Figure 2 shows the boxplot for NSL-KDD dataset. It was observed that, there was not much difference in the fitness values between all the four methods for small population size. The figure shows that the periodic population dynamics performed better than the static and chaotic population dynamics. Due to very less population swings in static GP and the normal GP method, the population size almost remains constant and hence performs better than the chaotic population dynamics.



(a) Population: 100



(b) Population: 500

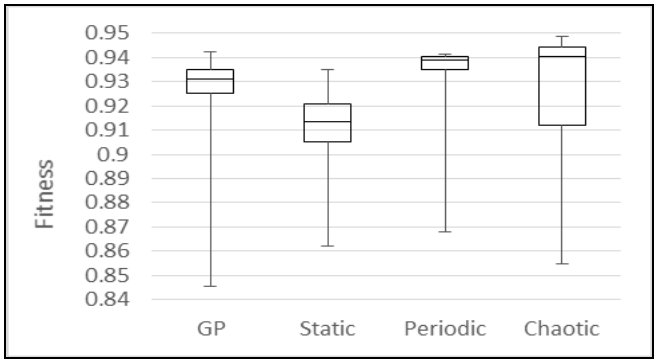
Fig. 2. Box plot showing the comparison of fitness values for NSL-KDD dataset

Chaotic GP performed best when the population size is high and worst when the population size is low. The main reason is that, when the population size is low, large population swings results in destroying good individuals from the gene pool. Whereas, at higher population size, the swings allow the chaotic GP to outperform the static, periodic and normal GP methods.

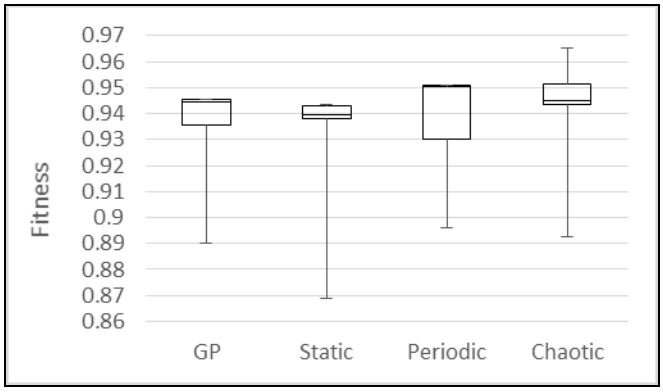
ISCX-IDS-2012

Figure 3 shows the boxplot for ISCX-IDS-2012 dataset. It can be seen that, chaotic method performed better compared to all other methods, but the mean fitness value of chaotic method was worse than the normal GP and periodic methods for low population size.

For high population size, the performance of chaotic method was better than all other methods. The performance of static method was worse for both low and high population sizes.



(a) Population: 100



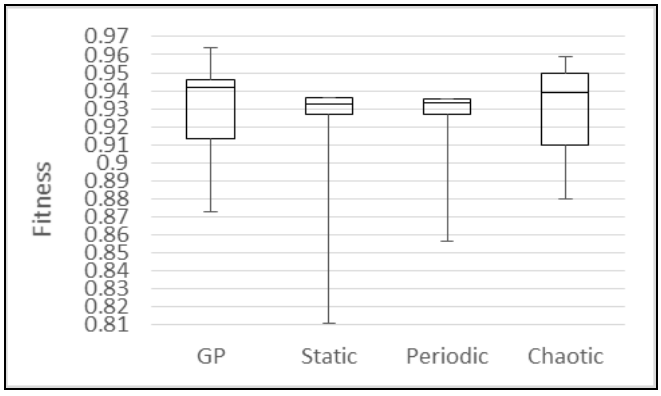
(b) Population: 500

Fig. 3. Box plot showing the comparison of fitness values for ISCX-2012 dataset

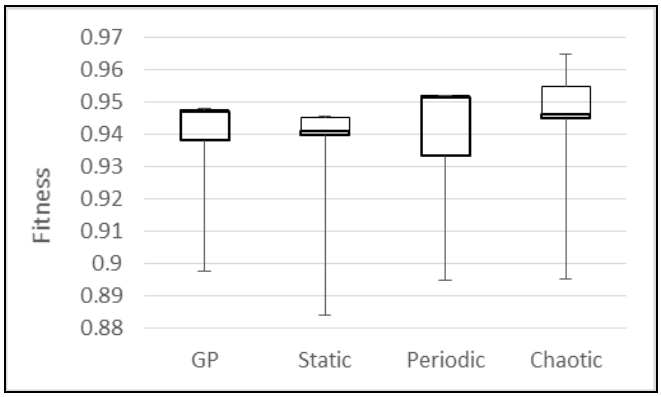
CICIDS2017

Figure 4 shows the boxplot for CICIDS2017 dataset. The figure shows that the normal GP method outperformed all other methods in getting the highest accuracy, but the mean fitness value of chaotic method was better than all other methods for low population size. The performance of static and periodic methods were worse in both the cases.

For high population size, the chaotic method outperformed all other methods. The performance of static method was worse than all other methods.



(a) Population: 100



(b) Population: 500

Fig. 4. Box plot showing the comparison of fitness values for CICIDS2017 dataset

The results show that the accuracy of the normal GP method can be made better by varying the population sizes in every generation. This gives room for creating new and better individuals rather than using the same individuals in each generation during the process of evolution.

Figure 5 depicts the fitness of the individuals obtained from the algorithm for different generations for all the three datasets for chaotic population dynamics. Even though there was lots of variations among the fitness obtained in the initial generations, the fitness almost remained constant for the later generations.

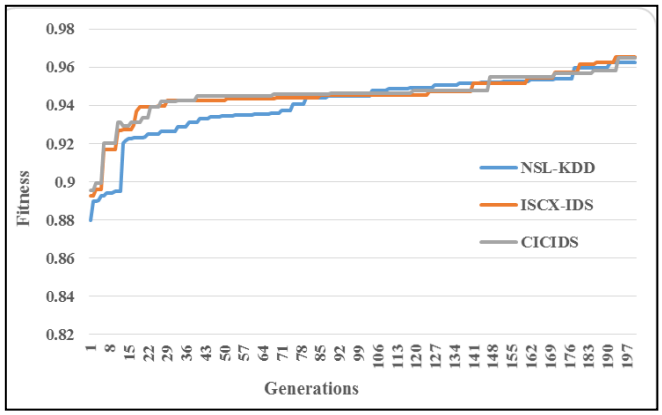


Fig. 5. Fitness obtained for chaotic dynamics for 200 generations

In order to better interpret the results obtained from our method, a comparison has been made with the results obtained from other methods. Table V shows the Detection rate and the false positive rate obtained by various researchers for the three datasets.

The proposed Chaotic-GP method gives good results with respect to detection rate. But the false positive rate obtained for few of the other methods are better than the proposed method. The comparison has been made only with few of the existing methods. The comprehensive comparison between most of the methods is a cumbersome task due to various facts like, sample size, number of features selected, etc. However, the results obtained shows that the performance of the Chaotic-GP method has greatly improved the accuracy of normal GP method.

Table- V: Performance comparison with existing solutions

Researchers	Method	Dataset	DR	FPR
Kumar et al. [20]	AMGA2-NB	ISCX-2012	94.5	7.0
Tan et al. [21]	MCA + EMD	ISCX-2012	90.12	7.92
Sally et al. [22]	PLL + NGL	ISCX-2012	95.31	0.80
Chen et al. [23]	SEIDS	CICIDS2017	89.15	2.83
Chen et al. [23]	SEIDS	NSL-KDD	81.59	3.53
Li, Longjie, et al [24]	GINI-GBDT-PSO	NSL-KDD	78.48	3.83
Papamartzivanos et al.[25]	Dendron	NSL-KDD	95.97	1.08
Proposed Method	Chaotic-GP	NSL-KDD	**96.63**	4.17
		ISCX-2012	**96.37**	1.9
		CICIDS2017	**93.1**	3.76

VI. CONCLUSION

In this paper, a novel method of varying the population size during the process of evolution has been proposed. The goal of the work was to check how the variance in the population size effects the genetic programming algorithm. A logistic map was applied to the algorithm and tested for three different population dynamics. The results showed that the proposed method was able to evolve better individuals compared to the fixed population size method. The method was trained and tested on low and high population sizes. For low population size, the periodic population dynamics produced better results whereas for high population size, the chaotic dynamics produced better results.

Three benchmark datasets, namely, NSL-KDD, ISCX-IDS-2012 and CICIDS2017 datasets were used to train and validate the algorithm. The results obtained are evaluated on the basis of the accuracy of each individual. The experimental results shows that the accuracy of the normal GP method can be improved to a greater extent by varying the population sizes at each generation.

The main drawback of the algorithm is that it takes a very long period of time to compute the accuracy of the individuals. Proposed method was validated using small sets of benchmark datasets. In future, the experiments can be conducted on real network traffic for a larger set of data.

REFERENCES

1. Jabeen, Hajira, and Abdul Rauf Baig. "A framework for optimization of genetic programming evolved classifier expressions using particle swarm optimization." International Conference on Hybrid Artificial Intelligence Systems. Springer, Berlin, Heidelberg, 2010.
2. Dufourq, Emmanuel, and Nelishia Pillay. "Hybridizing evolutionary algorithms for creating classifier ensembles." Nature and Biologically Inspired Computing (NaBIC), 2014 Sixth World Congress on. IEEE, 2014.
3. Pozi, Muhammad Syafiq Mohd, et al. "Improving anomalous rare attack detection rate for intrusion detection system using support vector machine and genetic programming." Neural Processing Letters 44.2 (2016): 279-290.
4. Wa'el, M. M., Hamdy N. Agiza, and Elsayed Radwan. "Intrusion detection using rough set parallel genetic programming based hybrid model." International Journal of Computer Science and Network Security 9.10 (2009): 23-33.
5. Le, Thi Anh, et al. "Malware detection using genetic programming." the 2014 Seventh IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA). IEEE, 2014.
6. Arabas, Jaroslaw, Zbigniew Michalewicz, and Jan Mulawka. "GAVaPS-a genetic algorithm with varying population size." Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence. IEEE, 1994.
7. T. Back, A.E. Eiben, N.A.L. van der Vaart, An empirical study on GAs ''without parameters'', in Proceedings of the 6th International Conference on Parallel Problem Solving from Nature (PPSN VI), vol. 1917 of LNCS (Springer, 2000), pp. 315–324
8. C. Fernandes, A. Rosa, Self-regulated population size in evolutionary algorithms, in Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN IX), vol. 4193 of LNCS (Springer, 2006), pp. 920–929
9. R.E. Smith, Adaptively resizing populations: an algorithm and analysis, in Proceedings of the 5th International Conference on Genetic Algorithms (San Francisco, CA, USA Morgan Kaufmann Publishers Inc, 1993) , 653 pp.
10. G.R. Harik, F.G. Lobo, A parameter-less genetic algorithm, in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999) (Morgan Kaufmann, 1999), pp. 258–267
11. Nelson, Thomas Harrison. "Genetic Algorithms with Chaotic Population Dynamics." (2010).
12. Ambusaidi, Mohammed A., et al. "Building an intrusion detection system using a filter-based feature selection algorithm." IEEE transactions on computers 65.10 (2016): 2986-2998.
13. Tesfahun, Abebe, and D. Lalitha Bhaskari. "Intrusion detection using random forests classifier with SMOTE and feature reduction." Cloud & Ubiquitous Computing & Emerging Technologies (CUBE), 2013 International Conference on. IEEE, 2013.
14. Kumar, Krishan, Gulshan Kumar, and Yogesh Kumar. "Feature selection approach for intrusion detection system." International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE) 2.5 (2013): 47-53.
15. Yulianto, Arif, Parman Sukarno, and Novian Anggis Suwastika. "Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset." Journal of Physics: Conference Series. Vol. 1192. No. 1. IOP Publishing, 2019.
16. Boukhamla, Akram, and Javier Coronel Gaviro. "CICIDS2017 dataset: performance improvements and validation as a robust intrusion detection system testbed."
17. Tavallaee, Mahbod, et al. "A detailed analysis of the KDD CUP 99 data set." 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. IEEE, 2009.
18. Shiravi, Ali, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. "Toward developing a systematic approach to generate benchmark datasets for intrusion detection." Computers & Security 31, no. 3 (2012): 357-374.
19. Sharafaldin, Iman, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization." ICISSP. 2018.
20. Kumar, Gulshan, and Krishan Kumar. "Design of an evolutionary approach for intrusion detection." The Scientific World Journal 2013 (2013).
21. Tan, Zhiyuan, et al. "Detection of denial-of-service attacks based on computer vision techniques." IEEE transactions on computers 64.9 (2014): 2519-2533.

22. Sallay, Hassen, et al. "A real time adaptive intrusion detection alert classifier for high speed networks." 2013 IEEE 12th International Symposium on Network Computing and Applications. IEEE, 2013.
23. Chen, Ze-Hong, and Chun-Wei Tsai. "An Effective Metaheuristic Algorithm for Intrusion Detection System." 2018 IEEE International Conference on Smart Internet of Things (SmartIoT). IEEE, 2018.
24. Li, Longjie, et al. "Towards Effective Network Intrusion Detection: A Hybrid Model Integrating Gini Index and GBDT with PSO." Journal of Sensors 2018 (2018).
25. Papamartzivanos, Dimitrios, Félix Gómez Mármol, and Georgios Kambourakis. "Dendron: Genetic trees driven rule induction for network intrusion detection systems." Future Generation Computer Systems 79 (2018): 558-574.

AUTHORS PROFILE

Ms Sunitha Guruprasad is an Assistant Professor in the Department of Computer Science and Engineering, St Joseph Engineering College, Mangaluru, India. She has 20 years of work experience with specialization in Computer Science and Engineering and active researcher in the field of Network Security and evolutionary computing.

Dr Rio D'Souza G. L. earned his BE in Electrical and Electronics Engg from the Karnataka University, MTech in Computer Engineering from Sri Jayachamarajendra College of Engg, and PhD from National Institute of Technology Karnataka. He also holds an Advanced Diploma in Applied Bioinformatics from the Bioinformatics Institute of India. Over a period spanning ten years, he has worked in several academic positions at Kalpataru Institute of Technology at Tiptur, Aloysius Institute of Computer Sciences, Mangaluru and NITK. He was with Jal International Corporation, Saudi Arabia, as Project Manager, for five years. Since 2006, he is with the Dept of Computer Science and Engg, St Joseph Engineering College (SJEC), Mangaluru, and has served in various capacities, both academic, as well as administrative. From June 2018, he has taken over as the Principal of SJEC. His current research interests include evolutionary computing, cloud computing and engineering education. Dr D'Souza is a Life Member of Indian Society for Technical Education (ISTE) and Advanced Computing & Communications Society (ACCS). He is a Senior Member of IEEE and a Fellow of the Institute of Engineers (India). He has been included in the panel of International Experts on Engineering Education by the Indo-Universal Collaboration on Engineering Education (IUCEE) and has conducted Pre-Certification Workshops for the IUCEE International Engineering Educator Certification Program at several colleges in India.