Hybrid Genetic-FSM Technique for Detection of High-Volume DoS Attack

Mohamed Samy Nafie[1], Hassan Abounaser[3]
Department of Computer Engineering
Arab Academy for Science, Technology and Maritime Transport (AASTMT), Cairo, Egypt

Khaled Adel[2]
Department of Computer Science
Ain Shams University
Cairo, Egypt

Amr Badr[4]
Department of Computer Science
Cairo University
Giza, Egypt

***Abstract*—Insecure networks are vulnerable to cyber-attacks, which may result in catastrophic damages on the local and global scope. Nevertheless, one of the tedious tasks in detecting any type of attack in a network, including DoS attacks, is to determine the thresholds required to discover whether an attack is occurring or not. In this paper, a hybrid system that incorporates different heuristic techniques along with a Finite State Machine is proposed to detect and classify DoS attacks. In the proposed system, a Genetic Programming technique combined with a Genetic Algorithm are designed and implemented to represent the system core that evolves an optimized tree—based detection model. A Hill-Climbing technique is also employed to enhance the system by providing a reference point value for evaluating the optimized model and gaining better performance. Several experiments with different configurations are conducted to test the system performance using a synthetic dataset that mimics real-world network traffic with different features and scenarios. The developed system is compared to many state-of-art techniques with respect to several performance metrics. Additionally, a Mann-Whitney Wilcoxon test is conducted to validate the accuracy of the proposed system. The results show that the developed system succeeds in achieving higher overall performance and prove to be statistically significant.**

Keywords*—Denial of Service (DoS); Evolutionary Algorithms (EA); Finite State Machine (FSM); Genetic Algorithm (GA); Genetic Programming (GP); Hill-Climbing Search

I. Introduction

In 1969, ARPANet invented the first link between two computers, which was the main predecessor of the Internet that appeared in 1983 [1]. Since then, computer networks have been in rapid development. Nowadays, they are continuously growing in size, complexity and efficiency, thus becoming one of the most important daily aspects of people's lives. For instance, people can use computer networks to send electronic e-mails, communicate via Voice over IP (VoIP) or transfer money via online bank portals instead of sending postal letters, making phone calls or going to the bank. Moreover, they can use it to access video streaming services, read news feeds, subscribe to online foreign exchange markets and many others things[2]. In addition to the basic services, new emerging technologies such as the Internet of Things (IoT) and Software Defined Networks (SDN) make use of computer networks to accomplish their goals [3],[4].

Consequently, such vital services have to be available for the end-users, allowing them to acquire and exchange information in an agile, easy and pervasive way on a daily basis. From another perspective, a major security concerns is for such services to become unavailable in a way that can vastly and adversely affect users. Denial of Service (DoS) is one type of network attacks, which threatens the availability of the victim's network by disrupting it, hence disabling any legitimate users from reaching the desired services. Another form of DoS attack is the Distributed Denial of Service (DDoS). DDoS attacks are usually launched with the aid of botnets [5]. A botnet is a set of compromised hosts controlled by a malicious attacker, which are instructed to perform illegal and malicious actions.

To support the availability and resiliency of computer networks and protect them from DoS attacks, a network monitoring approach such as NetFlow can help by providing flow analysis rather than standalone packet analysis, grasping a wider picture of the network's behavior. NetFlow is a protocol that captures and collect the flow of data entering and exiting network devices such as routers and switches. A flow data includes both source and destination IP addresses and ports, and transport protocol type. The Internet Engineering Task Force (IETF) currently standardizes it under the name IP Flow Information Export (IPFIX) [6].

Since insecure networks are vulnerable to cyber-attacks that may lead to catastrophic damages, improving Intrusion Detection System (IDS) is one of the hot topics that are widely researched. Different techniques are used to enhance the detection rate of attacks. Some of these techniques are Statistical Analysis, Clustering, Soft Computing such as Artificial Neural Networks (ANN), Support Vector Machines (SVM), Evolutionary Algorithms (EA) and others [7]. Additionally, it is proven that hybrid classifiers are capable of boosting the weakness of the single classifiers, hence producing better overall outcomes. Therefore, a hybrid detection approach is preferred [8],[9].

Genetic Algorithm [10] and Genetic Programming [11] are well-known meta-heuristic evolutionary algorithms that encompass the behavior of the natural selection of the fittest [12]. Both operate on a population of randomly initialized individuals, named chromosomes. Each chromosome resembles a possible solution for the required objective. The basic genetic operators are selection, crossover and mutation. By applying these operators, there is a chance that the individuals will be enhanced with each generation, producing finer solutions. To evaluate the finesse of each individual, a fitness function that generates a fitness-based value is computed. A number of generations is sustained until the termination criteria is met.

Hill-Climbing is a variant generate-and-test heuristic algorithm which follows the approach of trial and error by starting with an arbitrary solution and keeps iterating by making incremental changes[13]. It continuously tests newly generated solutions until an optimal one is reached. Hill-Climbing is sometimes called greedy local search because it aims towards a better solution in hope of finding the optimal one. The Hill-Climbing technique has three different types known as Simple, Stochastic and Steepest-Ascent, and these are applicable to both discrete and continuous search spaces[14]. Each type differs in how it approaches the neighboring nodes during the testing phase.

Many expert systems used for DoS detection rely on a predetermined threshold of certain parameters to determine whether there is an attack occurring inside the network or not. These approaches can yield a higher false-positive rate as well as a low detection accuracy. Often, these values are interleaved and depend on the type of network; thus, the manual calibration of such values are exhaustive due to an extremely large search space[14],[15]. Hence, it is very important to find an effective technique capable of detecting the occurrences of high-volume DoS attacks[16]. In this research, a system combining different heuristic techniques along with a Finite State Machine is proposed to detect the occurrences of high-volume DoS attacks.

The rest of this paper is organized as follows. Section II is a review of related work in detecting denial of service and presents different anomaly classification techniques. Section III demonstrates the algorithms and techniques proposed by this research. In Section IV, the preparation details of the utilized dataset are explained, and the obtained results of the conducted experiments are discussed. Finally, Section V concludes this work.

II. Related Work

Network reliability is important for both consumers and administrators. DoS attacks are common due to the fact that they can be easily launched in addition to their potential to cause catastrophic impacts on large scale networks[17]. Therefore, sophisticated and effective detection methods are considered as a priority to combat such attacks.

In [18], the authors have used supervised Artificial Neural Networks (ANNs) to detect the occurrences of DoS attacks. Each ANN is used to analyze the pattern of packet headers sent to a specific IP address if the number of packets is larger than a certain threshold. The system then deduces the packet threat level based on the majority of votes among multiple trained ANNs. The authors created their own dataset and environment to simulate the attacks and they compared their results with Chi-squared, Probabilistic Neural Networks and SVM, and proved to provide better detection results.

In [19], the authors used Hidden Markov Models (HMM) to represent different states of the network according to various features and behaviors. The system is composed of multiple HMM for each feature vector, including source and destination ports, source and destination IP addresses, and length of packets. Based on the HMMs output, the system calculates the suspicion level for the packets. The authors generated their test suite using OpenFlow to evaluate the accuracy of detection. The system showed good results in flagging malicious packets. One advantage of this system is that HMM can readily adapt to changes that happen to the networks without re-training.

In [20], the authors proposed a system that uses a modification of Holt-Winters named Holt-Winters for Digital Signature (HWDS), which generates a Digital Signature of Network Segment using Flow Analysis (DSNSF) for seven analyzed dimensions of the IP flow. Their system compares the collected flows, which differ from the generated DSNSF, with signatures of known attacks such as DoS and DDoS. Finally, a game-theoretic approach is used to mitigate the impact of the attacks.

In [21], the authors also used the concept of DSNSF to encapsulate six analyzed dimensions for each IP flow. The system monitors the network and extracts traffic characterization to the standard IPFIX form. The authors have used unsupervised GA to generate the DSNSF from the network information. Afterwards, the DSNSF is passed to a Fuzzy Logic classifier that assigns an anomalous score for each dimension in the real collected DSNSF using a Gaussian membership function. The scores are aggregated and a flow is labeled anomalous if that total score surpasses a certain cutoff value. The authors compared their results with Outlier Detector, SVM, CkNN, and the system provided better results in terms of detection accuracy.

In [22], the authors proposed a hybrid classification technique based on Artificial Bee Colony (ABC) and Acritical Fish Swarm (AFS) to enhance the detection accuracy of IDS. Additional techniques such as Fuzzy C-Means Clustering (FCM) and Correlation-based Feature Selection (CFS) are employed to split the training dataset and remove insignificant features. CART technique is implemented to generate If-Then rules that distinguish the normal and malicious records. Finally, the proposed hybrid system is trained via the generated rules.

III. Proposed System

This section aims to find an effective supervised-learning technique capable of automatically detecting the occurrences of high-volume DoS attack. To address this issue, a hybrid system of different heuristic techniques, along with a Finite State Machine (FSM) is proposed. The core of this system is a nested technique composed of a Genetic Algorithm (GA) representing the outer layer and a Genetic Programming (GP) representing the inner layer. The GP technique is designed and implemented to evolve the detection model for the attacks whereas the GA technique is designed and implemented to optimize the coefficients of this model. In details, the role of the GP technique is to evolve a tree-based mathematical expression capable of detecting the occurrences of DoS attacks whereas the role of the GA technique is to fine-tune the coefficients of this expression. The system produces two final models; one is for classifying the DoS attack entries while the other is for classifying the legitimate ones.

Fig. 1 illustrates the structure of one GA chromosome and a set of assigned GP chromosomes. All GP nodes that have a letter *k* denote the used coefficients from the GA chromosomes, while those that have a letter *v* denote different features used from the provided dataset.

The system starts by generating and initializing a random population for the outer-layer GA and the inner-layer GP according to the specified sizes n_1 and n_2 respectively. The random population of the GA acts as a feed for the GP expressions, where each value inside the GA genes can be sourced to the GP chromosomes. This is governed as a multi-to-multi relationship where each gene value of each GA chromosome can be utilized multiple times inside the same GP. This happens across different GP chromosomes within the population and can be completely neglected.

Each GA chromosome is assigned with a number of GP chromosomes. Afterwards, the GP starts its own evolution process based on the previously assigned values from the GA and runs for a number of g_2 generations. The final fitness value for each iteration is calculated based on the best fitness value resulting from all the inner-layer GP chromosomes. The whole process is repeated for an overall of g_1 generations.

Fig. 2 illustrates the core operation which is described above for the proposed system and shows the interfacing between the outer-layer GA and inner-layer GP techniques. It also displays how the overall fitness value is calculated for each iteration.

A. *Hill-Climbing*

As previously mentioned, a reference point is required for the fitness function to be able to evaluate the fitness of GP chromosomes. Since there is no predetermined value to use, the Hill-Climbing technique is employed to calculate a pivot-point that acts as a reference value to overcome this issue. The algorithm utilizes an initial value, an acceleration factor, acceleration directions and step size parameters. The initial value is the starting point from where the search begins and the acceleration direction determines whether the search will accelerate, decelerate or stop. It also provides the direction of the search and whether the pivot-point should increase or decrease according to the selected acceleration factor. A step size decides how far the search should hop into the search space. A large step size helps in discovering a larger space, but it can delay the algorithm from reaching a stable point [23].

For each iteration, all the acceleration directions are tested by generating random chromosomes and evaluating them using the testing dataset, then finally returning the best fitness. If the best fitness is accompanied with a stopping direction, then the step size would decrease which narrows the search plane, implying that the search is yielding positive results towards a satisfactory pivot-point. In contrast, if the stopping direction is not accompanied with the best fitness, then the step size increases according to both the acceleration direction and factor. This technique will run for a certain number of iterations until the final value is obtained.

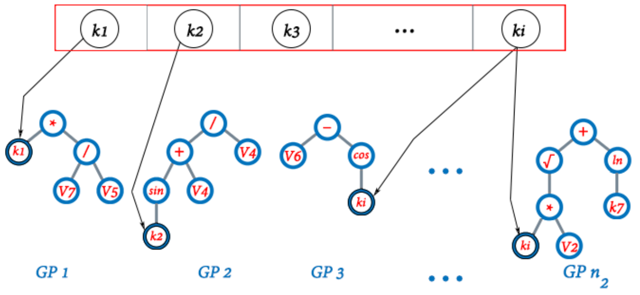


Fig. 1. Structure of GA and GP Chromosomes.

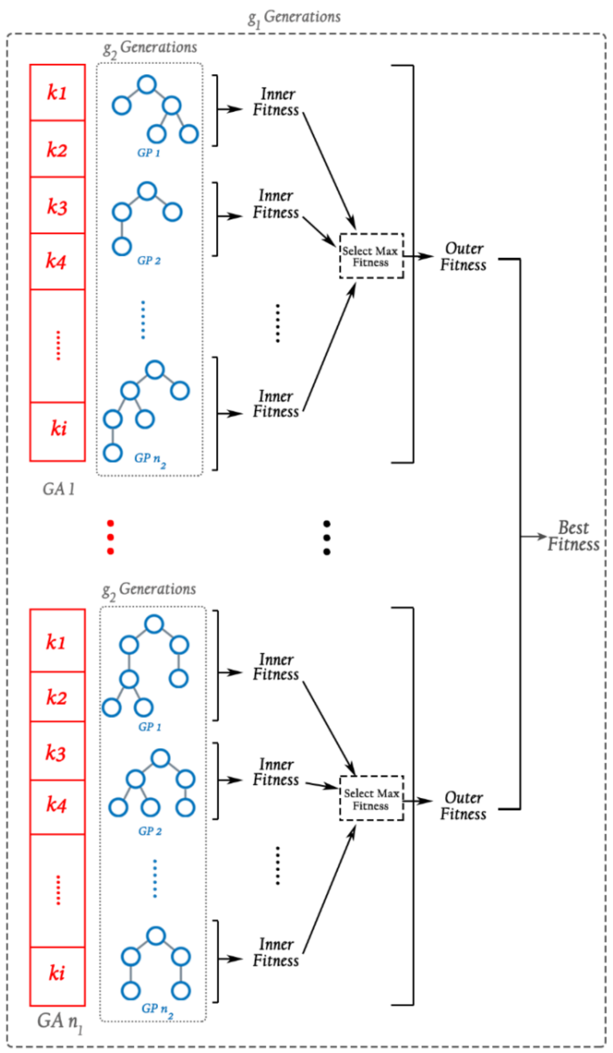


Fig. 2. Structure of the Proposed System. GA is Highlighted in Red and the GP is Highlighted in Blue.

B. Outer-Layer Genetic Algorithm (GA)

The chromosome consists of *k* genes; each one is a floating-point value. The crossover operator uses the standard single-point method, where two separating points are randomly assigned to the selected parents, and the genes between them are swapped. The selection operator uses the traditional roulette-wheel operator to select the fittest parents according to their fitness, in such a way that a parent chromosome with a higher fitness value will have a higher probability of being selected for crossover.

The mutation is carried in a non-uniform fashion as it depends on the value of best and average fitness of the current generation. The lower the average fitness of the population compared to the best fitness, the higher the mutation probability by a maximum of 50% probability. Non uniform mutation has proved to provide higher fitness values compared to the uniform fashion[24]. Elitism is also implemented and used to make sure that the fittest individual from a previous generation is transferred to the newer generation, so that the fitness value for the newest generation is either sustained or increased.

The objective of the of outer-layer GA is to fine-tune the performance of the inner-layer GP, in such a way that per each generation, a number of iterations for the inner-layer GPs are tested. The fitness function consists of a formula that selects the fittest individual from within the assigned GPs to each chromosome respectively.

C. Inner-Layer Genetic Programming (GP)

As mentioned earlier, the GP technique represents the core of the developed system along with the GA technique. It is designed and implemented to evolve a mathematical expression that can be used to classify the network traffic into DoS attacks, legitimate traffic or anomalous traffic. The tree structure of the inner-layer GP technique has a pre-determined depth *d*. Each GP chromosome is designed to encode a single tree representing a mathematical expression. Each gene represents a tree node that may hold a dataset feature, a mathematical operator or a coefficient value passed from the assigned GA chromosome of the currently evolving outer-layer GA generation.

A roulette-wheel operation is used to select two candidate parents that for the crossover operation. Then, according to a given probability value, the crossover operation will combine the genetic information in the parents to produce two new offspring. This is accomplished by generating a random cut-off point that is bounded by the tree dimension of the selected parents, around which, the two sub-trees are interchanged. Fig. 3 illustrates the crossover operation in inner-layer GP between the individuals GP_1 and GP_2.

The mutation probability value is calculated in the same non-uniform fashion utilized by the GA; however, the mutation operator itself is completely different. It can be a simple mutation or a subtree mutation. Simple mutation involves the modification of an individual gene, where the contents can be altered or expanded. Subtree mutation creates a new subtree initialized and grown from the mutating gene, taking into account the maximum allowable depth. Both mutation operators take into consideration the semantic and syntax limitations of the expression.

As for the fitness function, the fitness is computed based on the count of the correctly predicted entries. Each candidate model is compared to the pivot-point that has been previously calculated using Hill-Climbing technique. The system begins with classifying the DoS attacks aiming to produce a model for detecting them. If the evaluation of the mathematical model is found to be higher in value than the pivot-point, the system considers the entry to be an attack; otherwise, it is not. This prediction is compared to the expected label from the dataset. If it is correct, it is counted towards the fitness value, alternatively it is not counted. After the specified number of generations ends, the whole developed technique is repeated for the legitimate entries using the evaluation process. However, it differs in such a way that if the result of evaluating the mathematical model is found to be lower in value than the pivot-point, it is considered legitimate; otherwise, it is not.

D. Finite State Machine (FSM)

The finite state machine is responsible for interfacing all the outputs of the GP expressions into a single system developed to decide whether the output class is classified as attack, normal or anomalous. The FSM consists of three states corresponding to each of the output classes, along with an initial starting state. Each entry in the testing dataset is fed to the FSM. The FSM examines the entry by taking each attribute and assigning it to the corresponding parameter found in the GP expression. According to the evaluation of the expression against the previously found pivot-point, the FSM decides which state it will switch to. Fig. 4 illustrates the interfacing of the GP expressions into the FSM.

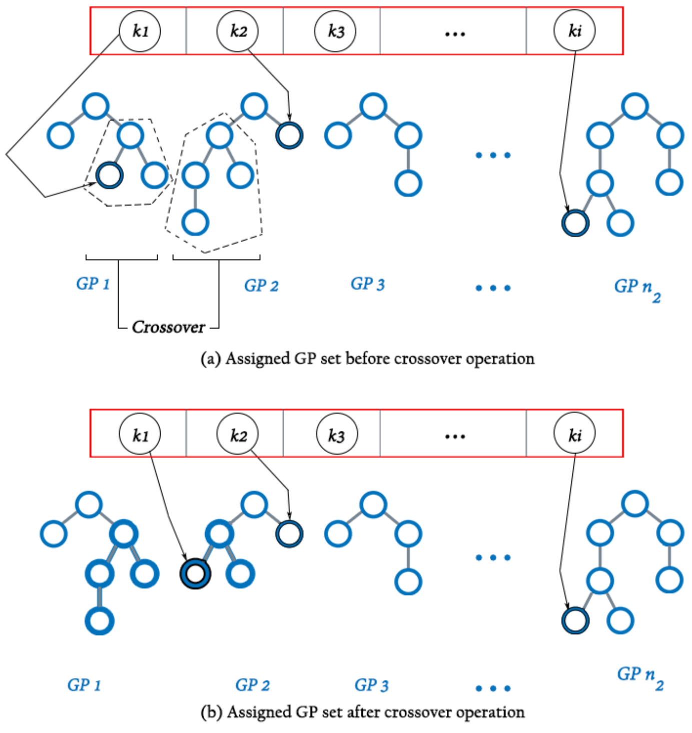

k1
k2
k3
...
ki
GP 1
GP 2
GP 3
GP n2
Crossover
(a) Assigned GP set before crossover operation
k1
k2
k3
...
ki
GP 1
GP 2
GP 3
GP n2
(b) Assigned GP set after crossover operation

Fig. 3. Crossover Operation between two GP Chromosomes.

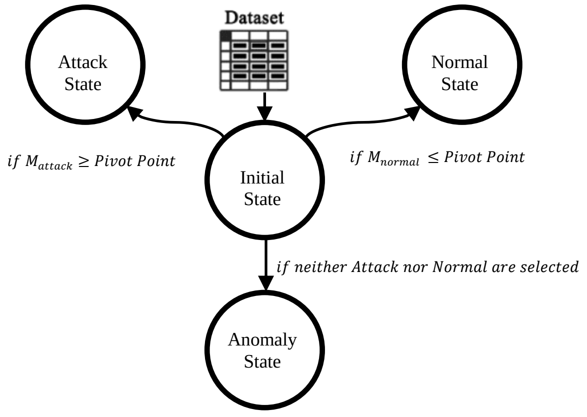

Dataset
Attack State
Normal State
Initial State
Anomaly State
$if\ M_{attack} \geq Pivot\ Point$
$if\ M_{normal} \leq Pivot\ Point$
$if\ neither\ Attack\ nor\ Normal\ are\ selected$

Fig. 4. Design of FSM.

IV. Results and Discussion

The following results were obtained from a series of experiments conducted using the developed system described above to acquire an optimized detection model. The experiments were performed on an Intel Core i5-3750K overclocked at 4.2 GHz CPU, with a 64-bit edition of Windows 10 and 16GB of RAM. A Nvidia 1080 GTX GPU conjointly with AleaGPU[25] software was used to handle all the computation of fitness function evaluation utilizing the power of parallel computing using CUDA. The full code has been implemented using C# on Microsoft Visual Studio 2017 with the Accord.NET [26] framework installed.

A. *Dataset Description and Prepration*

The experiments are conducted on the following dataset named CICIDS2017, provided by the Canadian Institute for Cybersecurity (CIC), University of New Brunswick (UNB). The original dataset contains 12 different types of attacks and 80 extracted network features [27]. However, since this research mainly focuses on high-volume DoS attacks, which are labeled “Attack”, all other non-DoS attacks are out of our scope of research, and thus, they were removed from the dataset. The remaining DoS attack types are labeled “Anomaly”. They are Heartbleed, Slowloris, GoldenEye and slowhttptest. Finally, the legitimate traffic is labeled “Normal”.

For the system to work as intended, features fed to the system are required to be represented quantitatively. Volumetric features such as duration, number of packets and bytes are numerical; however, IP addresses and ports information are nominal. To extract the properties of these features, Shannon Entropy is employed. Shannon Entropy measures the uncertainty associated with the randomness of a variable. When the variable distribution is concentrated, the entropy value tends to be minimal, while it is completely zero when all the variables are same. In contrast, the value tends to be high in case of discrepancy and fluctuations. The Shannon entropy value is usually between zero and one[28]. Given a set of features $X = \{ x_1, \dots, x_i, \dots, x_n \}$, the Shannon Entropy is defined by:

$$H(X) = -\sum_{i=1}^{n} P_i * \log_2(P_i)$$

where P_i is the probability of distribution of each feature from the set *X*. To calculate the probability of distribution, the frequency of occurrences of feature *i* is calculated by:

$$P_i = \frac{x_i}{\sum_{i=1}^{n} x_i}$$

Entries are aggregated based on the timestamp, source IP address, destination IP address, and source port in a similar way to how NetFlow aggregates data flows. Employing such an aggregation scheme creates an aggregated flow for each destination IP address in correspondence to every source IP juxtaposed with its source port for each time interval. This can help the system to effectively detect attacks[29]. Quantitative features are aggregated, while qualitative features are calculated separately. Table I shows the features used in the training process for each entry.

B. *Performance Analysis*

The results are obtained by running the dataset 5 times using 16 different configurations, and the output is then averaged to obtain a more accurate representation. The size of the GA chromosome utilized is 10 genes and the depth of the GP chromosomes utilized ranges between 3 and 5. The number of the generations for the outer-layer GA and inner-layer GP techniques are 25 and 50 respectively. Table II illustrates the number of training and testing entries in the dataset. Table III illustrates the settings used to train and test the system. Table IV lists the configurations for the common settings used to train and test the system. These configurations are sorted according to the number of GA and GP population sizes n_1 and n_2 respectively.

TABLE I. Features used in the Training after Pre-Processing

Feature	**Explanation**
Duration	The duration of the flow activity in minutes.
# of source packets	Total number of packets sent from source IP address to a destination IP address.
# of destination packets	Total number of packets sent from destination IP address to a source IP address.
# of source bytes	Total number of bytes sent from source IP address to a destination IP address.
# of destination bytes	Total number of bytes sent from destination IP address to a source IP address.
IP Entropy	Shannon Entropy calculated for source IP address(es).
Port Entropy	Shannon Entropy calculated for destination port(s)
Source to Destination Packet Ratio	Ratio between total number of source packets to destination packets
Label	Attack, Normal, Anomaly

TABLE II. Number of Training and Testing Entries in the Dataset

Labels	**Training Entries**	**Test Entries**
Attack	110992	47568
Normal	257084	110180
Anomaly	13356	5724

TABLE III. LIST OF SETTINGS USED FOR TRAINING AND TESTING

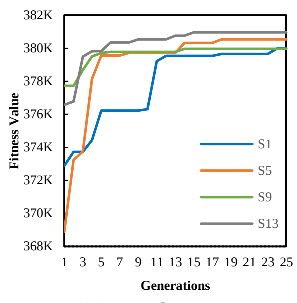
Setting	GA / GP Crossover %	GA / GP Population Size
S1	30% / 30%	24 / 240
S2	30% / 30%	24 / 480
S3	30% / 30%	48 / 240
S4	30% / 30%	48 / 480
S5	30 % / 60%	24 / 240
S6	30 % / 60%	24 / 480
S7	30 % / 60%	48 / 240
S8	30 % / 60%	48 / 480
S9	60 % / 30 %	24 / 240
S10	60 % / 30 %	24 / 480
S11	60 % / 30 %	48 / 240
S12	60 % / 30 %	48 / 480
S13	60 % / 60%	24 / 240
S14	60 % / 60%	24 / 480
S15	60 % / 60%	48 / 240
S16	60 % / 60%	48 / 480

TABLE IV. LIST OF CONFIGURATIONS SHOWING THE COMMON SETTINGS

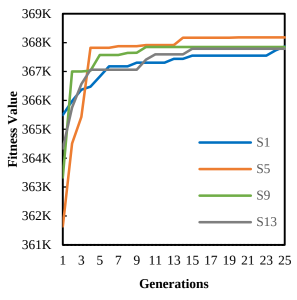
Configuration	Settings
C1	S1, S5, S9, S13
C2	S2, S6, S10, S14
C3	S3, S7, S11, S15
C4	S4, S8, S12, S16

Fig. 5 shows the convergence response for the training phase using the preselected configurations. Each one of these graphs illustrates the results for both normal and attack scenarios, showing the calculated fitness value across different generations. By inspecting the results, it is evident that those with a lower initial fitness value tend to converge faster than those with a higher initial fitness. This phenomenon is attributed to the fact that the system utilizes a non-uniform mutation function as already mentioned. Also, it is clear that the fitness values achieved after the run has finished is almost the same for all settings, which means that all settings are capable of properly training the system, but each with different performance. For instance, the settings with increased number of populations for both the GA and the GP yielded a fitness response with a higher initial value, which in turn, delivered better convergence output.

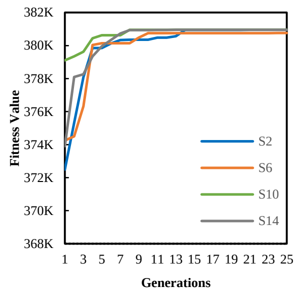
This trend can be associated with the fact that an elitism-based selection operator is performed by the system; therefore, those initially high values are always either improving or getting transferred to the newer generations. This way, starting with a relevant higher fitness value will positively influence the convergence performance. Table V shows the results of testing the proposed system. The testing phase involves testing the entries in the dataset against the output of the GA/GP and interfacing the output to the FSM, which is responsible for producing the final decision about whether the given entry signifies attacking, normal or anomalous behavior.



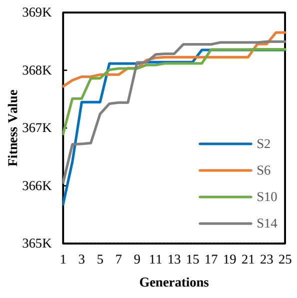
(a1) Attack convergence for settings S1, S5, S9 and S13



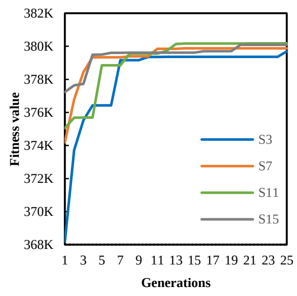
(n1) Normal convergence for settings S1, S5, S9 and S13



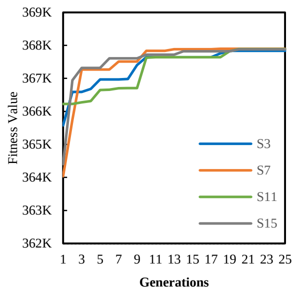
(a2) Attack convergence for settings S2, S6, S10 and S14



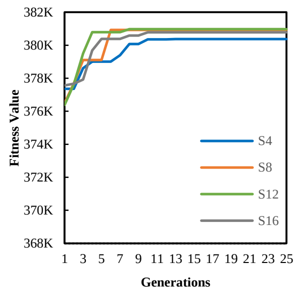
(n2) Normal convergence for settings S2, S6, S10 and S14



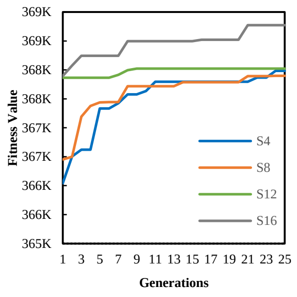
(a3) Attack convergence for settings S3, S7, S11 and S15



(n3) Normal convergence for settings S3, S7, S11 and S15



(a4) Attack convergence for settings S4, S8, S12 and S16



(n4) Normal convergence for settings S4, S8, S12 and S16

Fig. 5. The Convergence Performance for the Training phase using different Configurations.

By inspecting the relative standard deviation (RSD) and the mean fitness values, it is clear that almost all configurations have yielded very close final outputs. This means that varying the probability of crossover, mutation and the initial number of populations does not significantly affect the fitness value, but it mainly shapes the performance of the convergence process. Also, it is noticeable that setting S2 yielded the best testing accuracy while S11 yielded the worst results. This has occurred because the number of assigned GP chromosomes per GA chromosome were 20 GPs and 5 GPs for the settings S2 and S11, respectively. It can be observed that the diversity of the GP population per GA chromosome had a role into the final accuracy of the training phase. Therefore, these two settings will be the subject of further testing below.

Interpreting the output of the inner-layer GP, which consists of mathematical formulas and a reference point, is another major issue as it is mainly used to validate the decision with the supervision labels. As already mentioned, the key was to determine a pivot-point value that can be used as the reference point. To solve this issue, a Hill-Climbing technique is employed and tested with the aforementioned two settings, S2 and S11 based on their accuracy results. The results are illustrated in Table VI.

We can infer two important pieces of information from the table above. The first one is the significance of the Hill-Climbing technique in calculating the pivot-point value. A continuous sweeping range of values was tested as starting points for the Hill-Climbing technique. The calculated points were generated based on the difference between the highest and lowest values initially computed for the 100-interval pivot-point. Therefore, three different pivot-points based on this value were included. It can be seen that the values around the 100-interval gave the most accurate results. Fig. 6 shows the convergence performance of the system using the three pivot-points for the already declared best and worst settings.

TABLE V. THE 5-RUN AVERAGE FITNESS AND THE RELATIVE STANDARD DEVIATION OF THE TESTING PHASE FOR EACH CONFIGURATION

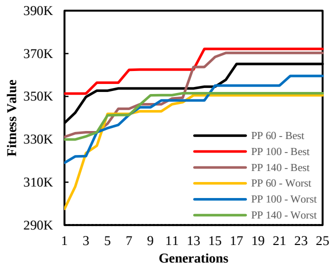
Config	Setting	Average Fitness	RSD (%)	Accuracy (%)
C1	S1	160.67K	0.621 %	98.318 %
	S5	161.09K	0.521 %	98.611 %
	S9	160.48K	0.383 %	98.175 %
	S13	160.88K	0.671 %	98.418 %
C2	S2	161.44K	0.289 %	98.761 %
	S6	161.28K	0.287 %	98.662 %
	S10	160.66K	1.090 %	98.283 %
	S14	161.31K	0.244 %	98.678 %
C3	S3	160.02K	0.840 %	97.889 %
	S7	160.39K	0.630 %	98.116 %
	S11	159.79K	0.798 %	97.752 %
	S15	160.61K	0.513 %	98.252 %
C4	S4	160.78K	0.550 %	98.357 %
	S8	161.24K	0.242 %	98.639 %
	S12	161.02K	0.379 %	98.505 %
	S16	161.34K	0.651 %	98.701 %

TABLE VI. OUTPUT ACCURACY USING DIFFERENT PIVOT-POINTS AND THE NUMBER OF GP GENERATIONS FOR THE BEST (S2) AND WORST SETTINGS (S11)

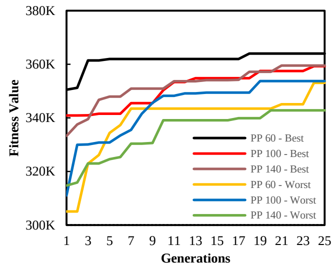
Pivot\GP Generations	1		25		50	
	Best	Worst	Best	Worst	Best	Worst
60	93.9%	87.5%	98.3%	98.1%	98.5%	98.3%
100	95.2%	90.6%	98.6%	98.1%	98.8%	98.3%
140	92.6%	87.0%	98.5%	98.0%	98.5%	98.2%

It is also worth mentioning that from all the runs that have been conducted, the *Port Entropy* were the most prominent feature appearing across all the produced models for the attack. This indicates that this feature has a special significance in the classification process. Entropy values tend to fluctuate higher during attacks, rendering the system more sensitive to such fluctuations[30].

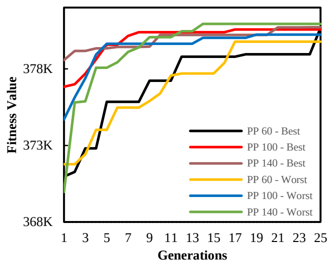
It is evident that the implementation of the Hill-Climbing method has helped in increasing the convergence performance. It can be seen from the graphs that the settings that were tested with the 100 pivot-point yielded a highest initial fitness value and continued to converge competitively quicker than the other two pivot-points, which means that the selection of the pivot- point is critical and can contribute to better results. To validate the results, a confidence interval metric based on a 95% confidence level is implemented. The confidence value is calculated based on the average accuracy of five different runs for each setting. It means that if the experiments are to be repeated, the output of the accuracy value will still fall between the calculated confidence interval ranges. Fig. 7 shows the results of implementing the confidence metric of detection accuracy for the attack, anomaly and normal classes.



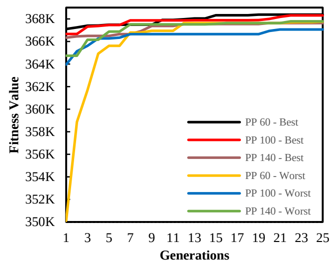
(a1) Attack convergence with 1 inner-layer GP generations



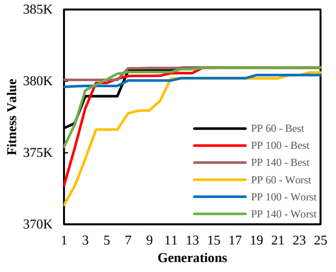
(n1) Normal convergence with 1 inner-layer GP generations



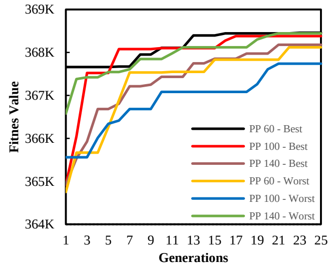
(a2) Attack convergence with 25 inner-layer GP generations



(n2) Normal convergence with 25 inner-layer GP generations



(a3) Attack convergence with 50 inner-layer GP generations



(n3) Normal convergence with 50 inner-layer GP generations

Fig. 6. Convergence Performance for the GP using the Three Pivot-Points with the Best (S2) and Worst (S11) Settings.

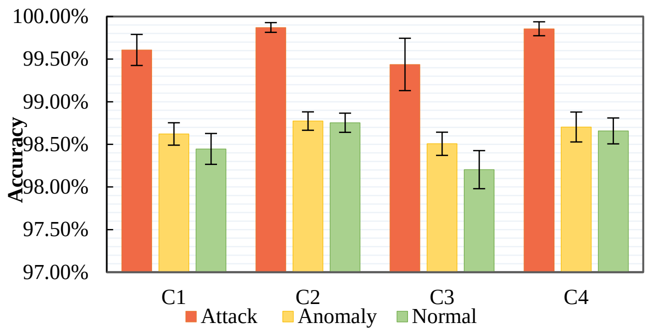


Fig. 7. Confidence Interval of 95% for the, Attack, Anomaly and Normal Labels.

It can be seen from the results in the figure above that the confidence interval for testing the developed system using the common settings C2 and C4 yielded the least error rates with 0.06% and 0.08% respecitvely for the attack detection. These two settings compared to the remaining settings, have the highest number of GP population, which means that the GP positively contributed to the consistency and coherence of the results, due to the higher diversity of the possible solutions, regardless of the final accuracy value.

C. Comparitive Study

Supplementary tests were carried out to evaluate the performance of the developed system. The system is compared to four state-of-art supervised learning techniques. These techniques are Fuzzy Hybrid Genetic Based Machine Learning (FH-GBML)[31], Genetic Programming-based learning of Compact and Accurate Fuzzy rule based classification for High Dimensional Problems (GP-COACH) [32] and Genetic Programming with AdaBoost (GPBoost) [33]. All of the previously mentioned techniques are implemented in well-known data mining tools such as KEEL [34] and WEKA [35].

FH-GBML is a Fuzzy Rule-Based Classification System (FBRCS) that establishes a combination of Michigan and Pittsburgh style methods. Additionally, it implements Genetic Cooperative Competitive Learning (GCCL) [36]. The GCCL approach mainly encodes a single fuzzy rule per individual. The main idea of FH-GBML is to generate a random population of rule sets using the Pittsburgh approach followed by one iteration applying the genetic operators. Then a pre-defined probability is calculated to decide whether to perform a single Michigan iteration for the whole rule set or not. Those new rules generated by the Michigan iteration replace the original rule set. Finally, all the rule sets generated during the Pittsburgh process replace the whole population by applying Elitism. GFS-Adaboost is a similar system to FH-GBML but makes use of Adaboost to boost the fuzzy rules along with the iterative genetic algorithm tuning [37].

While FBRCS provides high accuracy, the generated rule sets are usually complex and substantial, which may render them impractical for systems where execution speed is a crucial factor. GP-COACH simplifies the generated rule sets with minimal loss in accuracy incorporating Genetic Programming.

GP-COACH aims to compress the rule set after the learning phase. It uses the GCCL approach while employing token competition to promote diversity. Rules are codified in a context-free grammar fashion that enables new children to be corrected, maintaining the validity of the population. The population is then evaluated according to a global fitness function that promotes both accuracy and interpretability.

GPBoost uses Genetic Programming in conjunction with a boosting algorithm known as AdaBoost. Boosting algorithms are suitable for binary classification problems because they are capable of finding highly accurate classifications by combining many weaker classifiers, each of which is accurate. In each iteration, the algorithm updates the weights of the training examples based on those which have been already classified correctly using the current classifiers. GP can provide reasonably accurate models, which can act as weak classifiers for AdaBoost. Combining boosting with GP always provides better results compared to the results obtained with standalone GP. In this version of GPBoost, AdaBoost is extended to support multiclass classifications [38]. Each algorithm has been run 5 times and the average accuracy value along with the 95% confidence interval are calculated. The parameters used for each algorithm are the ones recommended by their original authors.

Accuracy (Acc.), Sensitivity (Sen.) and Precision (Pre.) are used as performance metrics for attack (TK), normal (NR) and anomaly (AM) labels. These metrics are commonly used in the evaluation of binary classifications in soft computing, but can also be used for multi-class classifications[39]. Table VII presents the evaluation results for the proposed system, FH-GBML, GP-COACH, GFS-AdaBoost and GPBoost.

The developed system achieved the highest accuracy among all the labels compared to the other four techniques. It also achieved the highest recall value for the anomaly label, thus correctly detecting the highest number of anomaly instances. FH-GBML achieved a similar performance compared to our approach in terms of attack and normal instance classification; however, achieving a precision of 100% and sensitivity of 0% signifies that it failed to detect any anomaly instances. This also proves the 96.5 ± 0% confidence interval for the anomaly accuracy. Additionally, it is evident that all methods that implement fuzzy logic have the best confidence interval values for the attack label due to the ability of fuzzy systems to tolerate imprecisions and uncertainties [40]. GFS-AdaBoost has benefited from the boosting, improving its ability to detect a few anomaly instances in comparison to the FH-GBML and GP-COACH which detected none.

GPBoost is the closest competitor to the proposed system in terms of design structure. It achieved lower scores than the proposed system in all classes. This implies that the outer-layer GA layer was successfully able to fine-tune the inner-layer GP expressions yielding better results. The use of FSM also decreased the total number of GP expressions required by one, in contrast to GPBoost that requires an expression for each label. GP-COACH has the lowest overall performance among all the tested techniques. It is apparent that it is the most sensitive method for attacks. This means it has a very low false negative rate. On the other hand, it achieved the lowest precision value indicating a high false positive rate.

To further validate the results of the proposed system, a Two-Sample Mann–Whitney–Wilcoxon (MWW) test is conducted. MWW is a nonparametric statistical test that is best suited for evaluating evolutionary algorithms due to their non-deterministic nature [41]. Table VIII shows that the proposed system has the best overall accuracy with a high confidence value, displaying statistically significant results *($P < 0.05$)*.

TABLE VII. COMPARISON BETWEEN ACCURACY, SENSITIVITY AND PRECISION OF EACH LABEL FOR THE DIFFERENT ALGORITHMS

Metric	**Label**	**Proposed System**	**FH-GBML**	**GP-COACH**	**GFS-AdaBoost**	**GP Boost**
Acc. (%)	TK	99.77 ± 0.3 %	99.48 ± 0.18 %	95.9 ± 0.25 %	97.21 ± 0.16 %	98.72 ± 1.04 %
	NR	98.63± 0.48 %	96.6 ± 0.25 %	95.28 ± 0.34 %	98.16 ± 0.73 %	97.09 ± 0.38 %
	AM	98.8 ± 0.24 %	96.5 ± 0 %	96.5 ± 0 %	97.16 ± 0.13 %	97.37 ± 0.31 %
Sen. (%)	TK	99.74 ± 0.02 %	99.76 ± 0.01 %	99.79 ± 0.02 %	99.65 ± 0.14 %	98.55 ± 2.2 %
	NR	98.93 ± 0.69 %	99.79 ± 0.05 %	95.95 ± 0.11 %	98.65 ± 0.54 %	98.8 ± 0.82 %
	AM	81.63 ± 8.31 %	0 ± 0 %	0 ± 0 %	21.82 ± 2.09 %	58.1 ± 6.63 %
Pre. (%)	TK	99.48 ± 1.01 %	98.49 ± 0.61 %	87.71 ± 0.45 %	91.49 ± 0.48 %	97.55 ± 1.92 %
	NR	98.98 ± 0.44 %	95.37 ± 0.3 %	96.76 ± 0.42 %	98.62 ± 0.58 %	96.98 ± 1.15 %
	AM	84.03 ± 4.27 %	100 ± 0 %	100 ± 0 %	88.07 ± 7.91 %	77.4 ± 11.99 %

TABLE VIII. OVERALL ACCURACY AND SIGNIFICANCE OF THE PROPOSED SYSTEM AND OTHER ALGORITHMS

Metric	**Proposed System**	**FH-GBML**	**GP-COACH**	**GFS-AdaBoost**	**GP Boost**
Overall Accuracy (%)	98.6 % ± 0.48 %	96.29 % ± 0.04 %	93.92 % ± 0.37 %	96.26 % ± 0.46%	96.20% ± 0.99%
P-Value	-	< 0.05	< 0.05	< 0.05	< 0.05

V. CONCLUSION

In this research, a hybrid system consisting of Genetic Algorithms (GA), Genetic Programming (GP) and Finite State Machine (FSM) for the detection of high-volume DoS attacks along with anomalous DoS events was developed. Hill-Climbing was employed to search for a suitable pivot-point to be used for the evaluation of the GP expression. A synthetic dataset that mimics a real-word network traffic has been used to test the proposed system in comparison with several state-of-the-art techniques. The results show that the developed system achieved better overall performance in terms of accuracy, sensitivity and precision with 95% confidence intervals. MWW is applied for further validation, achieving a P-value of < 0.05, rendering the proposed system statistically significant.

For future work, it is suggested to test the system with a larger dataset containing a larger number of attack labels and validate it by comparing it to more state-of-the-art techniques other than the ones used. A mitigation system applying the findings of the detection system should still be implemented into different types of networks, to assess how the system will perform when dealing with various types of network features and topologies. Also, the chromosomes of the GP can be further investigated in terms of mathematical formulation, which in turn, can establish a rigorous mathematical framework for the proposed system.

REFERENCES

[1] L. Roberts, “The Arpanet and computer networks,” Proc. ACM Conf. Hist. Pers. Work., pp. 51–58, 1986.

[2] C. HAYTHORNTHWAITE, “Introduction: The Internet in Everyday Life,” Am. Behav. Sci., vol. 45, no. 3, pp. 363–382, 2001.

[3] T. Qiu, N. Chen, K. Li, M. Atiquzzaman, and W. Zhao, “How can heterogeneous internet of things build our future: A survey,” IEEE Commun. Surv. Tutorials, vol. 20, no. 3, pp. 2011–2027, 2018.

[4] E. Molina and E. Jacob, “Software-defined networking in cyber-physical systems: A survey,” Comput. Electr. Eng., vol. 66, pp. 407–419, 2018.

[5] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, “Botnet in DDoS Attacks: Trends and Challenges,” IEEE Commun. Surv. Tutorials, vol. 17, no. 4, pp. 2242–2270, 2015.

[6] A. Pras, R. Sadre, A. Sperotto, T. Fioreze, D. Hausheer, and J. Schönwälder, “Using NetFlow/IPFIX for network management,” J. Netw. Syst. Manag., vol. 17, no. 4, pp. 482–487, 2009.

[7] M. Ahmed, A. Naser Mahmood, and J. Hu, “A survey of network anomaly detection techniques,” J. Netw. Comput. Appl., vol. 60, pp. 19–31, 2016.

[8] A. A. Aburomman and M. B. I. Reaz, “A survey of intrusion detection systems based on ensemble and hybrid classifiers,” Comput. Secur., vol. 65, pp. 135–152, 2017.

[9] A. L. Buczak and E. Guven, “A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection,” IEEE Commun. Surv. Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016.

[10] D. E. Goldberg and J. H. Holland, “Genetic Algorithms and,” Machine Learning, vol. 3, no. 2–3, pp. 95–99, 1988.

[11] J. R. Koza, “Genetic programming as a means for programming computers by natural selection,” Stat. Comput., vol. 4, no. 2, pp. 87–112, 1994.

[12] C. Darwin, The origin of species by means of natural selection: or, the preservation of favoured races in the struggle for life and the descent of man and selection in relation to sex. Modern library, 1872.

[13] A. Rosete-Suárez, A. Ochoa-Rodríguez, and M. Sebag, “Automatic Graph Drawing and Stochastic Hill Climbing,” Proc. of the {G}enetic and {E}volutionary {C}omputation {C}onf. {GECCO}-99, pp. 1699–1706, 1999.

[14] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.

[15] E. M. Knorr and R. T. Ng, "A Unified Notion of Outliers: Properties and Computation," in Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, 1997, pp. 219–222.

[16] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanismsMirkovic, J., & Reiher, P. (2004). A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Computer Communication Review, 34(2), 39. https://doi.org/10.1145/997150.997156," ACM SIGCOMM Comput. Commun. Rev., vol. 34, no. 2, p. 39, 2004.

[17] G. Fernandes, J. J. P. C. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey on network anomaly detection," Telecommun. Syst., vol. 70, no. 3, pp. 447–489, Mar. 2019.

[18] A. Saied, R. E. Overill, and T. Radzik, "Detection of known and unknown DDoS attacks using Artificial Neural Networks," Neurocomputing, vol. 172, pp. 385–393, 2016.

[19] T. Hurley, J. E. Perdomo, and A. Perez-Pons, "HMM-based intrusion detection system for software defined networking," Proc. - 2016 15th IEEE Int. Conf. Mach. Learn. Appl. ICMLA 2016, pp. 617–621, 2017.

[20] M. V. De Assis, A. H. Hamamoto, T. Abrao, and M. L. Proenca, "A game theoretical based system using holt-winters and genetic algorithm with fuzzy logic for DoS/DDoS mitigation on SDN networks," IEEE Access, vol. 5, pp. 9485–9496, 2017.

[21] A. H. Hamamoto, L. F. Carvalho, L. D. H. Sampaio, T. Abrão, and M. L. Proença, "Network Anomaly Detection System using Genetic Algorithm and Fuzzy Logic," Expert Syst. Appl., vol. 92, pp. 390–402, 2018.

[22] V. Hajisalem and S. Babaie, "A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection," Comput. Networks, vol. 136, pp. 37–50, 2018.

[23] D. Yuret, "Massachusetts Institute of Technology From Genetic Algorithms To Efficient Optimization," Artif. Intell., no. 1569, 1994.

[24] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," IEEE Trans. Syst. Man. Cybern., vol. 24, no. 4, pp. 656–667, 1994.

[25] QuantAlea AG, "Alea GPU." 2017.

[26] C. R. Souza, "The Accord.NET Framework." São Carlos, Brazil, 2014.

[27] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," Proc. 4th Int. Conf. Inf. Syst. Secur. Priv., no. Cic, pp. 108–116, 2018.

[28] K. Li, W. Zhou, S. Yu, and B. Dai, "Effective DDoS attacks detection using generalized entropy metric," in International Conference on Algorithms and Architectures for Parallel Processing, 2009,pp. 266–280.

[29] T. Ding, A. Aleroud, and G. Karabatis, "Multi-granular aggregation of network flows for security analysis," 2015 IEEE Int. Conf. Intell. Secur. Informatics Secur. World through an Alignment Technol. Intell. Humans Organ. ISI 2015, pp. 173–175, 2015.

[30] J. David and C. Thomas, "DDoS attack detection using fast entropy approach on flow-based network traffic," Procedia Comput. Sci., vol. 50, pp. 30–36, 2015.

[31] H. Ishibuchi, T. Yamamoto, and T. Nakashima, "Hybridization of fuzzy GBML approaches for pattern classification problems," IEEE Trans. Syst. Man, Cybern. Part B Cybern., vol. 35, no. 2, pp. 359–365, 2005.

[32] F. J. Berlanga, A. J. Rivera, M. J. del Jesus, and F. Herrera, "GP-COACH: Genetic Programming-based learning of COmpact and ACcurate fuzzy rule-based classification systems for High-dimensional problems," Inf. Sci. (Ny)., vol. 180, no. 8, pp. 1183–1200, 2010.

[33] A. T. R. P. Luzia Vidal de Souza and Anselmo C. Neto and Joel M. C. da Rosa, "We are IntechOpen , the world ' s leading publisher of Open Access books Built by scientists , for scientists TOP 1 % Control of a Proportional Hydraulic System," Intech open, vol. 2, p. 64, 2018.

[34] A. F. Ernández, J. L. Uengo, and J. D. Errac, "KEEL: A software tool to assess evolutionary algorithms for Data Mining problems (regression, classification, clustering, pattern mining and so on)," J. Mult. Log. Soft Comput., vol. 17, pp. 255–287, 2011.

[35] E. Frank, M. A. Hall, and I. H. Witten, The WEKA workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques," 4th ed. 2016.

[36] H. Ishibuchi, T. Nakashima, and T. Murata, "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems," IEEE Trans. Syst. Man, Cybern. Part B Cybern., vol. 29, no. 5, pp. 601–618, 1999.

[37] M. J. Del Jesus, F. Hoffmann, L. J. Navascués, and L. Sánchez, "Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms," IEEE Trans. Fuzzy Syst., vol. 12, no. 3, pp. 296–308, 2004.

[38] Y. Freund and R. R. E. Schapire, "Experiments with a New Boosting Algorithm," Int. Conf. Mach. Learn., pp. 148–156, 1996.

[39] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation," in AI 2006: Advances in Artificial Intelligence, 2006, pp. 1015–1021.

[40] L. A. Zadeh, "Fuzzy logic, neural networks, and soft computing," Commun. ACM, vol. 37, no. 3, pp. 77–84, 2002.

[41] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," Swarm Evol. Comput., vol. 1, no. 1, pp. 3–18, 2011.