Using Genetic Programming for Combining an Ensemble of Local and Global Outlier Algorithms to Detect New Attacks

***Abstract*—Modern intrusion detection systems must be able to discover in real-time new types of attacks. To this aim, automatic or semi-automatic techniques can be used; outlier detection algorithms are particularly apt to this task, as they can work in an unsupervised way. However, due to the different nature and behavior of the attacks, the performance of different outlier detection algorithms varies largely. In this ongoing work, we describe an approach aimed to understand whether effectively an ensemble of outlier algorithms can be used to detect new types of attacks in intrusion detection systems. In particular, Genetic Programming (GP) is adopted to build the combining function of an ensemble of local and global outlier detection algorithms, which are used to detect different types of attack. Preliminary experiments, conducted on the well-known NSL-KDD dataset are encouraging and confirm that 1) depending on the type of attacks, it would be better to use local or global detection algorithms 2) the GP-based ensemble improves the performance in comparison with commonly used combining functions.**

***Index Terms*—outlier detection, ensemble of classifiers, genetic programming, intrusion detection**

I. Introduction

As the number of network connections and also the speed of these networks is increasing, the problem of analyzing large streams of data in real time for detecting possible attacks gains relevance in the scientific community of cybersecurity. Typically, *Intrusion Detection Systems* (IDS) [1] are used to detect unauthorized accesses to computer systems and networks (in this case, they are named *Network Intrusion Detection Systems*, NIDS).

Data mining techniques, often combined with the interesting paradigm of the ensemble, are largely used to support the detection phase of the NIDS, as witnessed by a large number of papers published in recent years. However, most of the data mining algorithms are used to classify known attacks, while the potential of outlier detection techniques for discovering new types of anomaly and of attack is not sufficiently exploited. Ensemble [2] [3] [4] is a learning paradigm where multiple component learners are trained for the same task by a learning algorithm, and the predictions of the component learners are combined for dealing with new unseen instances. Among the advantages in using ensemble of classifiers, we would like to remind that they help to reduce the variance of the error, the bias, and the dependence from a single dataset; furthermore, they can be built in an incremental way and they are apt to distributed implementations. Finally, for the particular task of intrusion detection, they are able to combine different classifiers that together provide complementary information and also are particularly apt to handle unbalanced classes.

Indeed, in the field of cybersecurity, the ensemble paradigm has been used mainly for the classification of attacks [5] [6], while a few works as the paper [7] adopt outlier algorithms to discover anomalies or new types of attack; in the latter paper, the experimental results, conducted on the DARPA 1998 dataset, demonstrated that some outlier algorithms are more apt to detects some types of anomalies.

An ensemble of outliers was employed by Nguyen et al. [8] for the detection of anomalies in high dimensional datasets. The ensemble improved the outlier detection accuracy compared with other well-established methods. Finally, they showed that the approach can also be used for abnormal data detection in a real-world context. Zimek et al. [9] adopted a subsampling method for efficient and effective unsupervised outlier detection ensembles, as subsampling introduces diversity among the individual outlier detectors. They compared the sample-based ensemble against feature bagging and verified that it is more effective in comparison with the bagging algorithm and the single outlier detectors. Lazarevic and Kumar [10] proposed a novel feature bagging approach for detecting outliers in high dimensional and noisy dataset, based on the division of the data into many randomly chosen subsets; experimental results confirmed the goodness of the performance of the ensemble.

On the basis of the analysis of the works present in literature, in this paper, we we are going to investigate the feasibility of using an ensemble of outlier algorithms to detect new types of attacks in intrusion detection systems. In particular, a genetic programming tool is adopted to build the combining function of an ensemble of local and global outlier detection algorithms. In addition, an analysis is conducted in order to assess the performance of local and global outlier algorithms with respect to different types of attack. Preliminary experiments, conducted on the NSL-KDD dataset, confirmed that the combining function developed by the ensemble improves the performance in comparison with commonly used combining functions. However, this is an ongoing work and we are aware that outlier detection techniques alone cannot be used to discovery attacks, however, their usage is useful in combination with other anomaly detection and classification techniques or in semi-automatic approaches in combination with domain experts, also in order to reduce the number of false positives.

The rest of the paper is structured as follows. Section II introduces some background information about global and local outliers and about the genetic programming method. Section III illustrates the methodology used and the scenario in which it can be used. In Section V, the GP-based technique adopted to combine the scores of the different outlier detected methods is described in detail. Section VI show some experiments conducted to verify the effectiveness of approach. Finally, Section VII concludes the work and addresses some future research directions.

II. Backgrounds

In this section, some background information is supplied, i.e., the main differences between local and global outlier algorithms and the genetic programming method.

A. Local vs global outlier detection algorithms

Different categorizations of outlier detection algorithms, mainly for their adoption in the ensemble paradigm are present in literature [11]. Here, we consider the categorization that divides the techniques for the detection of the outliers into two categories, on the basis of the outlier the algorithm looks for: global outlier and local outlier algorihtms. A global outlier is a data object that significantly deviates from the other samples of the dataset. On the other hand, a local outlier is a data sample, which significantly deviates from its neighbors.

From another point of view, some outliers can be defined as collective. Indeed, a subset of data samples can be recognized as a collective outlier if this subset significantly deviates from the other data objects. Note that, in this case each individual data object may not be an outlier. Detecting collective outlier is very important in some application such as intrusion detection. For example, in a DOS (*Denial of Service*) attack, a packet alone sent from a computer to another one can be considered as a normal pattern and not an outlier. However, if we consider many computers sending DOS packets, the whole pattern is an anomaly and they should be recognize as collective outliers. It is important to note that, in a dataset they may exist different types of outliers. Thus, for our task, we need different types of outlier detection methods.

Another categorization distinguishes supervised outlier detection methods and unsupervised outlier detection methods. In this paper, we focus on unsupervised outlier detection methods. Indeed, some outlier detection algorithms are based on the assumption that the normal samples form some clusters. Some of these methods recognize outliers as data objects not belonging to any cluster, or data objects belonging to small clusters.

Outlier detection methods based on clustering have some advantages. The first one is that they are unsupervised and thus they can detect outliers without the need of labeled data. The second one is that, after the process of clustering, it is only needed to compare each data object with each cluster, or recognize small clusters as collective outliers. Thus, this type of outlier detection methods is typically quite fast. However, these approaches are appropriate for recognizing global and collective outliers, but most of them are not very successfully to detect local outliers.

Another important category of unsupervised outlier detection methods is based on computing the *Local Outlier Factor* (LOF). The LOF is the ratio between the local density of a data object and the mean of the local density of its neighbors. These methods calculate the local outlier factor for each data object and then rank them based on this factor; the objects having highest LOF are recognized as local outliers.

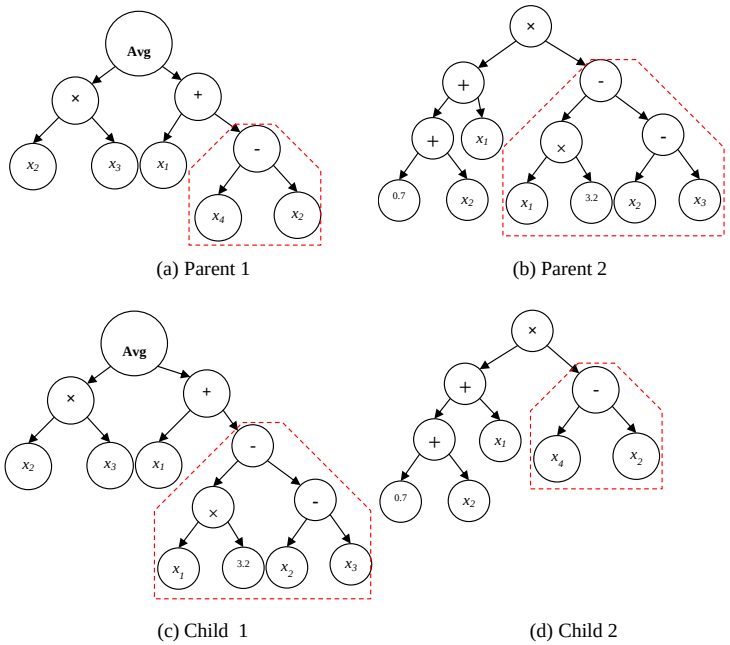
B. Genetic Programming

Genetic Programming is one of the most used evolutionary algorithms and, inspired by the the evolutionary theories of Darwin, it evolves a population of solutions (individuals) of a problem for a number of generations. It can be used to learn both the structure and parameters of the model and the individuals (chromosomes) of a standard GP are trees. The internal nodes of the tree are functions and the leaves are typically the problem variables, constants or random numbers. The initial population of GP is a set of trees generated randomly. During the evolutionary process, the individuals are evolved until they reach the optimal solution (or a good approximation of it) of the problem, or until a maximum number of generations is reached. The evolution is driven by a function of fitness, which is chosen for the particular problem to be solved and represents the goodness of a solution of the problem. Similar to other evolutionary algorithms, for each generation, two genetic operators (crossover and mutation) are performed on some individuals, chosen randomly on the basis of their fitness: individuals with better fitness have more chance to be chosen.

The crossover operator swaps two random subtrees of two individuals (parents) and generates two new individuals (children). Moreover, the mutation operator (typically used with a lower probability in comparison with the crossover) is performed on a single individual and mutates a random subtree and generate a new individual. Figures 1 and 2, show examples of crossover and mutation, respectively. By these two operators GP can search the problem landscape in order to find the optimal solution. Then, the new generated individuals are added to the populations and compete with other individuals based on their fitness, i.e., the better individuals have more chance to survive. This process leads to find better solutions during the evolution of the process.

III. Methodology and scenario

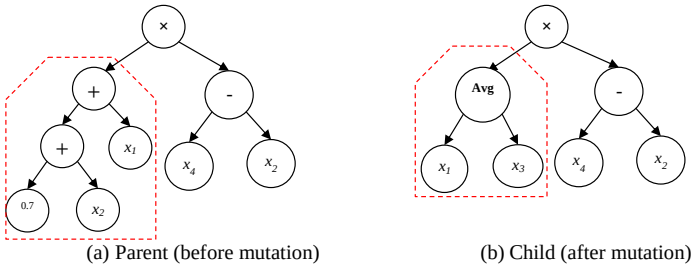
As remarked in the introduction, the problem of detecting new types of attack in intrusion detection systems could hardly be resolved with our approach alone. Indeed, the technique here described should be integrated into a framework including at least a module for classifying known attacks and a module for anomaly detection. Furthermore, it would be better whether the output of our outlier detection ensemble method is analyzed by an expert of the domain. In addition, in order for our approach to be useful for an intrusion detection systems, some conditions should be satisfied, as listed in the following:



(a) Parent 1 (b) Parent 2

(c) Child 1 (d) Child 2

Fig. 1. An example of GP crossover: In GP crossover two random subtrees of the parents are selected and swapped with each others and generates two new individuals. Here, the function set contains Avg, $+$, and $\times$, and the terminal set contains problem variables and some random numbers.



(a) Parent (before mutation) (b) Child (after mutation)

Fig. 2. An example of GP mutation: In GP mutation, a random subtree of the parent is selected and substituted with a new random subtree. Here, the function set contains Avg, $+$, and $\times$, and the terminal set contains problem variables and some random numbers.

- Typically, a new type of attack is not isolated, so it is reasonable to be able to collect a significant sample of data containing information about the attack.
- Attacks should deviate from normal activities.
- A domain expert should be able to recognize that a tuple is not related to a normal connection but it is probably a new type of attack.

Anyway, the constraint are quite reasonable and realistic, therefore, we are confident that our method has a good applicability.

In this work, we want to experiment the usage of an ensemble of outlier detection algorithms. There are different methods to combine an ensemble of classifiers; in this work, we adopt the method based on computing a function directly of the scores produced by the outlier algorithms, as it permits to compute the combining function of the ensemble without re-running the single algorithms. Consider the case of an ensemble of n_1 (n_2) local (global) outlier detection algorithms, as illustrated in Figure 1.Each outlier detection algorithm outputs different outlier score vectors SV_i with i indicating the i_{th} algorithm, that reflects the probability of each tuple of the dataset being an outlier (this probability is first normalized in order to compare different outlier detection algorithms). Then, the outlier score vectors are combined by using some aggregative function; for example, the *Sum* function sums the corresponding entry of the vectors and finally output as outliers the K points having the largest sum. Given the j^{th} tuple x_j, $Sum(x_j) = \sum_{i=1}^{n_1} SV_i(x_j)$.

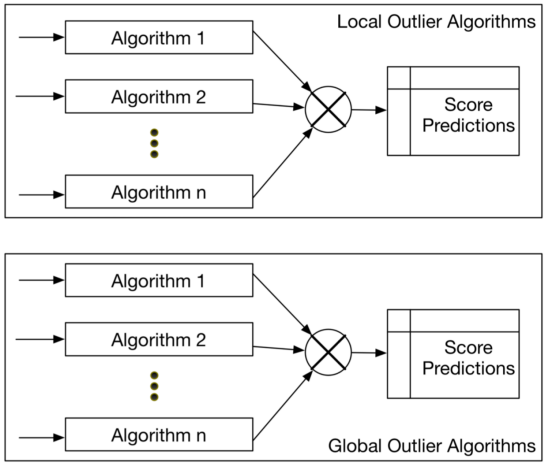


Fig. 3. Combining the scores of the outlier detection algorithms in an ensemble.

Other functions commonly used and included in this work are: *Avg*, which considers the average of all the scores, *Max*, which takes the maximum of all the scores, *Product*, which is the product of all the scores and *Square*, which uses the square root of the sum of the scores.

IV. Proposal of an Architecture for Detecting Attacks

As different outlier detection algorithms score outliers in different ways, combining the scores of different outlier detection methods is very important and challenging. However, using simple aggregation functions may not have obtain good results, ass different outlier detection algorithms score outliers in different ways, combining the scores of different outlier detection methods is very important and challenging. However, using simple aggregation functions may not have obtain good results, ass different outlier detection algorithms score outliers in different ways, combining the scores of different outlier detection methods is very important and challenging. However, using simple aggregation functions may not have obtain good results, ass different outlier detection algorithms score outliers in different ways, combining the scores of different outlier detection methods is very important and challenging. However,

using simple aggregation functions may not have obtain good results, as

V. Combining the scores of different outlier detection methods using GP

As different outlier detection algorithms score outliers in different ways, combining the scores of different outlier detection methods is very important and challenging. However, using simple aggregation functions may not have obtain good results, as also confirmed by the experimental results of subsection VI-D. Therefore, we propose to use evolutionary algorithms to find a better aggregation function. Indeed, a function in form of a GP tree was used to combine the local and global outlier detection algorithms of the ensemble.

As mentioned before, combining the results of different outlier detection algorithms is a challenging problem. There is no useful prior knowledge that suggests the best strategy to combine the outlier scores of different methods. We need to find a combination function f that, for each data instance x, takes as input the outlier score $SV_i(x)$ given by each outlier detection algorithm i and returns a final outlier score $f(SV_1(x), SV_2(x), \ldots, SV_n(x))$ for that data instance. Using GP can be very helpful in this case, because it does not need any information about the structure of the combination function and it can find both its structure and its parameters with a few kknowledge about the domain.

As GP engine, the CellulAr GEnetic programming (CAGE) [12] [13] is adopted. It can run both on distributed-memory parallel computers and on distributed environments and it is based on the fine-grained cellular model. The generation of the population, the different operators (i.e., crossover, mutation, etc.) are the same defined in the classical GP introduced by Koza [14].

The functions used to build the GP trees are simply the aggregation function defined in the previous subsection: *Avg*, *Max*, *Product*, and *Square*, all of them having as input the outlier scores of some outlier algorithms (both local and global ones). The functions were replicated with different arity of input, i.e., 3, 4 and 5. The terminal set is constituted by all the outlier detection algorithms (described in detail in the subsection VI-B), some local outlier detection methods, i.e., LOF (Local Outlier Factor), LoOP (Local Outlier Probabilities), with or without PCA (Principal Component Analysis) to obtain a 2 dimensionality reduction, some global outlier detection methods, i.e., SVM (Support Vector Machines) and X-means with and without PCA. Figure 4 show an example of a GP individual, which is a potential combination function of outlier scores.

The fitness function is computed on the basis of a validation set. We suppose that, for a small sample of data, the information about the real class of the attacks is present. The quite realistic hypothesis is that a domain expert is able to distinguish attacks and normal connections, at least for a small sample of data. On the basis of this hypothesis, in the experimental section, some experiments are conducted using a validation set of 5%, 10% or 20% of the entire dataset. The fitness

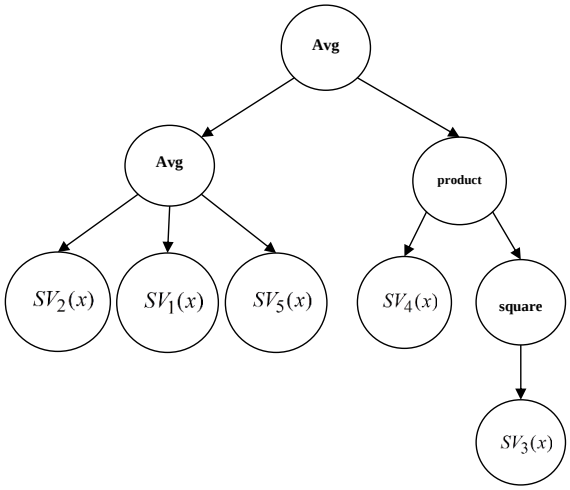


Fig. 4. A sample GP individual which is a potential combination function. $SV_i(x)$ represents the outlier score of data instance x given by outlier detection method i.

function is computed as the average classification accuracy of the minority and majority classes over the validation set. This function works better than the overall classification accuracy in the case of unbalanced classes [15]. More in detail, Algorithm 1 presents the pseudo-code of the GP method used to find the appropriate combination function for aggregating the scores of different local and global outlier detection methods. For the sake of simplicity, the details about the cellular GP algorithm are not reported; we refer to the original paper cited above, for a detailed description.

VI. Experimental results

As this is an ongoing work, the experimental section aims to try to answer to some research topics, i.e., we want to investigate whether:

(i) Local and/or global outlier algorithms can be used to detect new types of attacks.
(ii) A methodology or analysis can be used to decide if it is better to use local or global techniques to detect a certain type of attack.
(iii) An ensemble of outliers can improve the accuracy of local and global outlier detection techniques.
(iv) GP can be used to develop a combining function improving the performance of simple combining functions of the ensemble.

A. Dataset and parameters

The experiments were conducted on the NSL-KDD dataset [16], which overcomes some of the issues regarding the KDD dataset, as avoiding redundant tuples, eliminating duplicate records and increasing the difficult level of some attacks.

The NSL–KDD dataset is sampled in subsets containing all the normal connections and all the connections belonging to a defined type of attack. The percentage (attacks) of the sample is chosen in a way to create a dataset with an outlier percentage

Input: Validation Data
$D = \left\{ SV_1(x_j), \ldots, SV_{n_1+n_2}(x_j), \mathsf{Class}(x_j) \right\}_{j=1}^{m}$.
▷ x_j: The j^{th} data instance of the validation data.
▷ n_1: Number of local outlier detection methods
▷ n_2: Number of global outlier detection methods
▷ $SV_i(x_j)$: Score that the outlier detection method i gives to the data instance j.
output : The best GP individual, i.e., a function for combining the scores
Function Set: *Avg, Max, Product,* and *Square* with different arity (3, 4 and 5)
Terminal Set: variables $SV_1, SV_2, \ldots, SV_{n1+n2}$
Fitness Function: $Avg\_class\_accuracy = \frac{TP}{TP+FN} + \frac{TN}{TN+FP}$ over data D

Generating a random population
$\text{POP}_{\text{initialization}} = \left\{ f_k(SV_1, \ldots, SV_{n_1+n_2}) \right\}_{k=1}^{N}$.
▷ N: The population size.
▷ Each GP individual is a combination function that aggregates the scores of different methods.
while *terminationcondition* **do**
  Evaluate fitness of each new individual k (Fitness(f_k) over D).
  Operator ← SelectGeneticOperator(Crossover, Mutation, Reproduction)
  if *Operator* = *Crossover* **then**
    P_1 and P_2 ← Parent Selection(Population)
    c_1 and c_2 ← Crossover (P_1, P_2)
  end
  if *Operator* = *Mutation* **then**
    P_1 ← Parent Selection(Population)
    c_1 ← Mutation (P_1)
  end
  Survival selection
end
Return The best individual ($f_{best}(SV_1, SV_2, \ldots, SV_{n1+n2})$)

Algorithm 1: The pseudo-code of the GP-based approach.

in the range 1–5%, i.e., if the percentage of attacks is greater than 5%, it is sampled so it is reduced to this percentage, on the contrary, its percentage remains unaltered. DoS attacks were removed from the dataset, as they are really similar to normal connections, are very numerous and then they are hardly recognized by outlier detection techniques. Then, for the following types of attack: satan, nmap, portsweep and ipsweep, we built specific datasets for each attack. For attacks with a very low number of connections, we built datasets using the class of the attack; i.e., based on a predefined mapping, the attacks are collected in the following classes: u2r, and r2l (see Table I).

All the experiments were performed on a Linux cluster with 16 Itanium2 1.4 GHz nodes, each having 2 GBytes of main memory and connected by a Myrinet high performance network. All the results of the next subsections were obtained by averaging 30 runs. No tuning phase has been conducted for the GP tool, but the same parameters used in the original paper were used, listed in the following: a probability of crossover equal to 0.7 and of mutation equal to 0.1, a maximum depth equal to 7, a population of 120 individuals and 500 as number of generations.

Among the many metrics for evaluating classifier systems, in this paper we adopt the area under the ROC curve (AUC). It is based on the metrics of recall and precision, which are commonly used for problems presenting unbalanced classes. The latter two metrics give an idea of the capacity of the system in individuating the attacks and in reducing the number of false alarms; indeed, recall represents the proportion of correctly predicted attack cases to the actual size of the attack class (a value of 100% indicate we detect all the attacks, however, we can individuate also a large number of false attacks); precision represents the proportion of attack cases that were correctly predicted relative to the predicted size of the attack class (a value of 100% indicates that no false alarms were signaled, however a large number of alarms could be not detected). In practice, the AUC metric is the value of the area under the ROC curve, where the ROC curve is computed comparing the false positive rate and the true positive rate. The first term measures the capacity to correctly detect attacks (i.e. recall). The second term measures the rate between the false alarm signaled above all normal connections processed. Computing the area, we have a number to describe the goodness of classifier. An AUC close to 1 means an optimal recognition rate.

The outlier detection algorithms used for the experiments were taken from the data mining framework ELKI, [17]; they are used with standard parameters and are listed in the following subsection.

B. Using global and local outlier methods

In this section, the experiments conducted using different local and global outlier algorithms are reported. We basically consider two types of algorithms: *local*, and *global* one. The class of local algorithms considered contains LOF, LoOP, with a variant using PCA to obtain a two-dimensionality reduction. Global algorithms comprise SVM outlier and k-means outlier, the last one employing X-means as clustering algorithm, also considering a variant of the algorithms using PCA.

Table II summarizes the AUC scores obtained (on the left, the local and on the right, the global techniques). As we are interested in understanding the effectiveness of local and/or global techniques, we highlighted local (global) scores, when they are significantly better than all the other global (local) scores for a given type of attack. While for the attacks named ipsweep, nmap, portsweep and satan, the global techniques work considerably better, for r2l and u2r, the local techniques outperform the global one. In addition, with the exception of the satan and the r2l attack, the type of technique (local or global) having accuracy smaller than the other one, obtains really poor performance. Therefore, it is unfeasible to adopt one or the other type of technique for all the attacks.

TABLE I
TYPE OF ATTACKS IN THE NSL–KDD DATASET

Attack	Class	Attack	Class	Attack	Class
back	dos	ipsweep	probe	phf	r2l
land	dos	nmap	probe	spy	r2l
neptune	dos	portsweep	probe	warezclient	r2l
pod	dos	satan	probe	warezmaster	r2l
smurf	dos	ftp_write	r2l	buffer_overflow	u2r
teardrop	dos	guess_passwd	r2l	loadmodule	u2r
normal	normal	imap	r2l	perl	u2r
		multihop	r2l	rootkit	u2r

TABLE II
SUMMARY OF EXPERIMENTS CONDUCTED ON THE NSL-KDD DATASET BY USING SINGLE OUTLIER ALGORITHM (AUC FOR LOCAL TECHNIQUES ON THE LEFT, GLOBAL ON THE RIGHT.

Attack	LOF	LOF - PCA	loOP	loOP - PCA	SVM	SVM - PCA	Xmeans	Xmeans - PCA
ipsweep	0.40 ± 0.0180	0.50 ± 0.0437	0.35 ± 0.0086	0.60 ± 0.0177	0.97 ± 0.0014	0.93 ± 0.0263	0.82 ± 0.0132	0.79 ± 0.0498
nmap	0.33 ± 0.0243	0.41 ± 0.0598	0.43 ± 0.0198	0.38 ± 0.0840	0.96 ± 0.0027	0.93 ± 0.0134	0.80 ± 0.0156	0.79 ± 0.0378
portsweep	0.70 ± 0.0161	0.68 ± 0.0540	0.64 ± 0.0247	0.66 ± 0.0581	0.68 ± 0.0316	0.94 ± 0.0215	0.91 ± 0.0212	0.80 ± 0.1236
satan	0.59 ± 0.0102	0.63 ± 0.0223	0.57 ± 0.0129	0.64 ± 0.0229	0.91 ± 0.0041	0.96 ± 0.0255	0.85 ± 0.0165	0.80 ± 0.0404
r2l	0.77 ± 0.0257	0.79 ± 0.0644	0.76 ± 0.0262	0.83 ± 0.0424	0.67 ± 0.0269	0.64 ± 0.0343	0.60 ± 0.0376	0.58 ± 0.0327
u2r	0.86 ± 0.0474	0.71 ± 0.0773	0.89 ± 0.0405	0.71 ± 0.0784	0.34 ± 0.0852	0.31 ± 0.0882	0.40 ± 0.1119	0.43 ± 0.1101

C. Distribution of attacks and normal data vs performance global/local detection algorithms.

Here, we try to investigate the following question: why, for some types of attacks, global outlier detection methods perform better than local outlier techniques and vice versa for some other attacks better performance is obtained by the global techniques?

Remember that collective outliers are defined as follows: a subset of data objects collectively deviate significantly from the whole data set, even if the individual data objects may not be outliers. Global outlier detection algorithms such SVM and k-means detect outliers by individuating small clusters. For this reason, this type of methods performs better in the case of collective and global outliers. On the contrary, local detection algorithms, such as LOF or LoOP, performs well when the outliers are local and they are scattered within the normal data. It would be really helpful to understand in which case, on the basis of the distribution of some properties of the data, it would be better to use one or the other type of algorithm.

To this aim, for each class of attack of the NSL-KDD dataset, we draw a graph of the distribution of the normal data and of the anomalies (attacks). In practice, the center (mean) of the normal data is computed and, then, the distance of each normal and anomaly tuple from the center is calculated. The mean and the standard deviation of the distance of the normal data from the normal center, and the distance of the anomaly data from the normal center are reported in Figure 5. This statistic shows the distribution of the anomalies with respect to the normal data. When the variance of the distance of the anomaly samples from the normal center is very low, it means that the anomalies are collective. More over if this variance is high, it could mean that the anomalies are spread across the normal data. However, from the figures, it is not possible to correlate the distribution of the attacks and of the normal data with the particular outlier detection technique. In addition, in presence of new types of attacks, we can hardly find the distribution of the attacks and of the normal data. To the best of our knowledge, no other studies in literature have found a strong correlation. Therefore, even if some (local or global) method obtains a good accuracy for a given type of attack, it is almost impossible to decide in advance which method is better to use. Therefore, in the next subsection, an analysis is conducted to understand whether an ensemble could perform well for all the attacks.

D. Using an ensemble of outlier methods

In many cases, an ensemble can improve the performance of the single methods. Therefore, here different combining functions of the ensemble (*Avg*, *Max*, *Product* and *Square*) were used to combine both local and global techniques.

Tables III and IV report the AUC scores obtained respectively for the ensemble of local and global algorithms. The scores that are significantly better than the corresponding function of the other global (local) score for a given type of

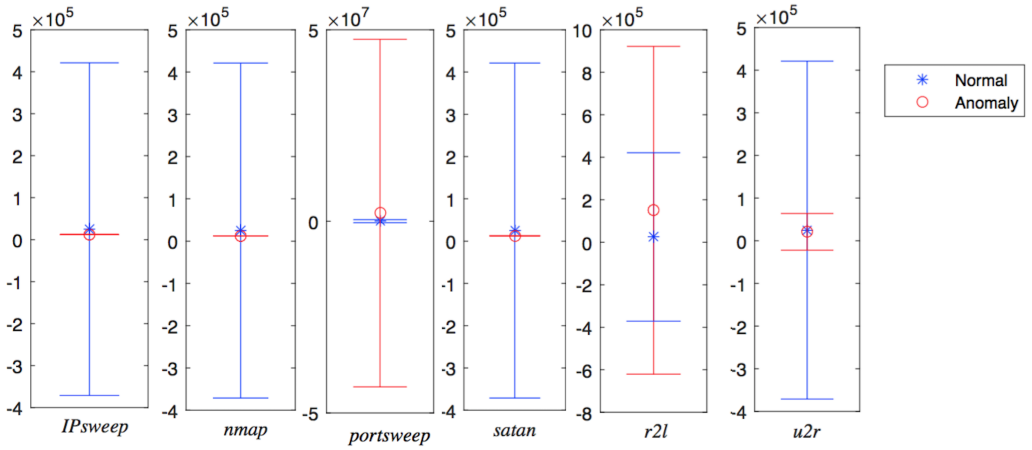


Fig. 5. Mean and standard deviation of the distance of each sample (anomaly and normal) from the normal center.

TABLE III
COMPARISON OF DIFFERENT COMBINING FUNCTIONS OF THE ENSEMBLE, USING ONLY LOCAL TECHNIQUES.

Attack	MaxEnsemble	AvgEnsemble	SquareEnsemble	ProductEnsemble
ipsweep	0.44 ± 0.0230	0.43 ± 0.0301	0.43 ± 0.0212	0.48 ± 0.0044
nmap	0.38 ± 0.0462	0.37 ± 0.0526	0.38 ± 0.0480	0.45 ± 0.0589
portsweep	0.69 ± 0.0596	**0.72 ± 0.0647**	**0.70 ± 0.0617**	0.59 ± 0.0596
satan	**0.63 ± 0.0545**	**0.64 ± 0.0648**	**0.64 ± 0.0577**	0.54 ± 0.0411
r2l	**0.78 ± 0.0732**	**0.81 ± 0.0640**	**0.80 ± 0.0687**	**0.62 ± 0.0889**
u2r	**0.81 ± 0.0516**	**0.83 ± 0.0457**	**0.82 ± 0.0454**	**0.67 ± 0.1551**

TABLE IV
COMPARISON OF DIFFERENT COMBINING FUNCTIONS OF THE ENSEMBLE, USING ONLY GLOBAL TECHNIQUES.

Attack	MaxEnsemble	AvgEnsemble	SquareEnsemble	ProductEnsemble
ipsweep	**0.64 ± 0.1705**	**0.67 ± 0.1506**	**0.55 ± 0.1320**	0.32 ± 0.0897
nmap	**0.64 ± 0.1546**	**0.66 ± 0.1486**	**0.49 ± 0.0999**	0.34 ± 0.0804
portsweep	**0.76 ± 0.2024**	0.65 ± 0.1101	0.67 ± 0.1365	0.39 ± 0.0700
satan	0.60 ± 0.1720	0.58 ± 0.1313	0.49 ± 0.0942	0.33 ± 0.0548
r2l	0.57 ± 0.0580	0.59 ± 0.0664	0.51 ± 0.0554	0.42 ± 0.0489
u2r	0.45 ± 0.1217	0.40 ± 0.0902	0.57 ± 0.1100	0.61 ± 0.0977

attack are reported in bold, i.e., the local score is reported in bold only if it outperforms the global ensemble for the same combining function and vice versa.

For most of the cases, *Max* and *Mean* perform better than the other functions and usually *Product* performs poorly. While, for the attack named ipsweep, nmap, and satan, the ensemble of global techniques work considerably better, the ensemble of local techniques outperforms the ensemble of global algorithms for satan, r2l, u2r. With the exception of the attack named portsweep (in which both the techniques work quite well, even if the local ensemble using the *Max* function obtains the best result), the ensemble fails to mitigate the poor performance of a type of technique on some types of attack.

E. Using the GP combining function

The experiments conducted in the previous subsections fails in finding a single or an ensemble of techniques effectively useful to detect all the types of attack. Therefore, we conduct some experiments on using a GP-based approach to combine the scores of the different global and local outlier detection methods. A validation set of 5%, 10% or 20% of the entire dataset was used to compute the fitness function of the GP algorithm.

Table III and IV report the AUC scores obtained for the different types of attack by using three percentages of validation set (5%, 10% and 20%). For each attack, we report in bold the minimum percentage of validation set in which GP is significantly better than all the functions both for the global and local ensemble (Tables III and IV).

It is evident that the GP ensemble, also when only 5% of the dataset has information about the attack, obtains considerably good performance in terms of accuracy. For some typologies of attack (ipsweep, nmap, r2l, u2r) using a larger validation set, permits to improve the accuracy, while for the other attacks using 20% of validation set does not obtain an improvement, which balances out the effort necessary to validate 20% of the data. However, for most of the attack, a 5% validation set is sufficient to outperform both the local and global ensemble.

TABLE V
AUC VALUE FOR THE ENSEMBLE OF LOCAL AND GLOBAL OUTLIER USING THE COMBINING FUNCTION DEVELOPED BY GP FOR THREE PERCENTAGES OF VALIDATION SET (5%, 10% AND 20%).

Attack	GP Ensemble (5%)	GP Ensemble (10%)	GP Ensemble (20%)
ipsweep	**0.78 ± 0.0812**	0.85 ± 0.0650	0.87 ± 0.0422
nmap	**0.75 ± 0.1400**	0.83 ± 0.1133	0.85 ± 0.0855
portsweep	**0.85 ± 0.2280**	0.88 ± 0.1586	0.90 ± 0.0892
satan	**0.70 ± 0.1653**	0.75 ± 0.0842	0.76 ± 0.0747
r2l	0.69 ± 0.2211	0.74 ± 0.1358	**0.82 ± 0.0958**
u2r	0.67 ± 0.1482	0.75 ± 0.0998	**0.84 ± 0.0772**

Only for the attacks named r2l and u2r, probably due to the low number of tuples representing these attacks, a 20% validation set produces a significantly better accuracy, even if, by using 5% the validation set, a reasonable level of accuracy is reached. Anyway, as a general consideration, it is necessary to choose a tradeoff between having a small validation set and obtain a good accuracy.

VII. CONCLUSIONS AND FUTURE WORK

A methodology based on an ensemble of local and global outlier detection algorithms is presented. This methodology can be effectively integrated in a general framework using classification, and anomaly detection to detect new types of attack in intrusion detection systems. The novelty of the approach consists in using a module based on an ensemble of outlier detection methods in order to individuate new types of attacks. Genetic programming is used to generate the combining function of the ensemble, which obtains good improvements in terms of AUC in comparison with standard combination functions used for outlier detections. However, the good accuracy obtained by the GP approach is balanced out by the need and the effort required to generate a validation set, necessary to train the GP population. On the contrary, the ensemble of outliers does not require to label a sample of the data, but performs well only for some types of attack. Therefore, we know that this technique alone is not sufficient to detect new types of attack but we are confident that, in combination with other techniques, can help domain experts to reduce the number of false alarms to be analyzed. In addition, some limitations of this technique include that it cannot be applied to diffused attack (i.e., DoS attacks). In future works, we want to investigate the applicability of our technique to real-world intrusion detection systems.

REFERENCES

[1] M. Bhuyan, D. Bhattacharyya, and J. Kalita, "Network anomaly detection: Methods, systems and tools," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 1, pp. 303–336, First 2014.

[2] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[3] Y. Freund and R. Shapire, "Experiments with a new boosting algorithm," in *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96)*. Morgan Kaufmann, 1996, pp. 148–156.

[4] L. Kuncheva, "Combining pattern classifiers: methods and algorithms," 2004.

[5] G. Folino and F. S. Pisani, "Evolving meta-ensemble of classifiers for handling incomplete and unbalanced datasets in the cyber security domain," *Appl. Soft Comput.*, vol. 47, pp. 179–190, 2016.

[6] G. Folino, F. S. Pisani, and P. Sabatino, "A distributed intrusion detection framework based on evolved specialized ensembles of classifiers," in *Applications of Evolutionary Computation - 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 - April 1, 2016, Proceedings, Part I*, ser. Lecture Notes in Computer Science, vol. 9597. Springer, 2016, pp. 315–331.

[7] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava, *A Comparative Study of Anomaly Detection Schemes in Network Intrusion Detection*, ch. 3, pp. 25–36. [Online]. Available: http://epubs.siam.org/doi/abs/10.1137/1.9781611972733.3

[8] H. V. Nguyen, H. H. Ang, and V. Gopalkrishnan, "Mining outliers with ensemble of heterogeneous detectors on random subspaces," in *International Conference on Database Systems for Advanced Applications*. Springer, 2010, pp. 368–383.

[9] A. Zimek, M. Gaudet, R. J. Campello, and J. Sander, "Subsampling for efficient and effective unsupervised outlier detection ensembles," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 428–436.

[10] A. Lazarevic and V. Kumar, "Feature bagging for outlier detection," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, pp. 157–166.

[11] C. C. Aggarwal, "Outlier ensembles: Position paper," *SIGKDD Explor. Newsl.*, vol. 14, no. 2, pp. 49–58, Apr. 2013.

[12] G. Folino, C. Pizzuti, and G. Spezzano, "A scalable cellular implementation of parallel genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 1, pp. 37–53, February 2003.

[13] ——, "Improving induction decision trees with parallel genetic programming," in *10th Euromicro Workshop on Parallel, Distributed and Network-Based Processing (PDP 2002), 9-11 January 2002, Canary Islands, Spain*, 2002, pp. 181–18.

[14] J. R. Koza, *Genetic Programming: On the Programming of Computers by means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[15] U. Bhowan, M. Johnston, and M. Zhang, "Developing new fitness functions in genetic programming for classification with unbalanced data," *IEEE Trans. Systems, Man, and Cybernetics, Part B*, vol. 42, no. 2, pp. 406–421, 2012.

[16] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, July 2009, pp. 1–6.

[17] E. Schubert, A. Koos, T. Emrich, A. Züfle, K. A. Schmid, and A. Zimek, "A framework for clustering uncertain data," *PVLDB*, vol. 8, no. 12, pp. 1976–1987, 2015. [Online]. Available: http://www.vldb.org/pvldb/vol8/p1976-schubert.pdf