A Taxonomy and Comparative Evaluation of Algorithms for Parallel Anomaly Detection

Shashank Shanbhag*, Yu Gu†, Tilman Wolf*
*Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA 01002
Email: {sshanbha,wolf}@ecs.umass.edu
†NEC Laboratories America, Inc., Princeton, NJ, USA
Email: yugu@nec-labs.com

***Abstract*—Anomaly detection in network traffic is an important technique for identifying operation and security problems in networks. Numerous anomaly detection algorithms have been proposed and deployed in practice. The recent availability of high-performance embedded processors in network systems has made it possible to implement these algorithms to monitor traffic in real-time. Since it is unlikely that any single anomaly detection technique will ever be sufficient, we propose the use of multiple existing anomaly detection algorithms in parallel. In this paper, we develop a method of combining different classes of anomaly detection algorithms and address the question of which combination of existing anomaly detection algorithms achieves the best detection accuracy. We also present a taxonomy of anomaly detection algorithms and evaluate six specific algorithms on a common evaluation platform. Based on this evaluation, we identify the combination of anomaly detection algorithms that achieve the highest detection accuracy and derive a few rules that can be used when deciding on combining and aggregating multiple algorithms.**

***Index Terms*—network measurement, anomaly detection, data aggregation**

I. Introduction

Traffic anomaly detection is the first step towards defending the Internet from abusive use and malicious attacks. Timely and accurate identification of these traffic anomalies is essential for the effective response with counter-measures such as admission control, source throttling or rate limits. As traffic anomalies potentially affect the availability and effectiveness of critical Internet applications such as online services, transactions and real time remote cooperation, the role of traffic anomaly detection has become increasingly important.

Over the years, a broad range of techniques have been applied on the problem of traffic anomaly detection and many algorithms have been proposed and/or deployed by the networking community. However, these algorithms are often evaluated on individually collected traces that are not publicly available. And as there is no unified benchmark for evaluating these anomaly detectors, there is no consistent view of their performances. When it comes to deployment, it is then difficult to select the best one for a certain environment.

Furthermore, recent availability of high-performance embedded processors in network systems has made it possible to implement these algorithms to monitor traffic in parallel and in real-time. Since it is unlikely that any single anomaly detection technique will ever be sufficient, the use of multiple existing anomaly detection algorithms in parallel could be a potential optimal policy. However, it remains unknown that if a limited number of algorithms can be applied, how to pick algorithms that will constitute the best combination?

In this work, we are targeting at solving the above issues by proposing a single evaluation framework with a suite of solutions.

- We first develop a taxonomy of anomaly detection algorithms based on their underlying techniques. According to this taxonomy, we select and implement six of the various detection algorithms from different categories.
- We then propose a set of benchmark traffic models for evaluating these algorithms. Real traces conforming to these benchmark models are generated in Deterlab, which is a real network testbed, and are used as input to the selected algorithms.
- We derive a similarity metric that specifies performance correlations across algorithms. This information is then used as a guideline of picking the right set of algorithms when selecting algorithms for our parallel online detection engine.
- We have evaluated the six different anomaly detection algorithms, both individually and with combinations, to determine which combination of algorithms yields the most accurate anomaly detector.

Building a taxonomy of anomaly detection algorithms according to their underlying techniques helps understand the nature of different algorithms. The benchmark traffic models then provide a gauge where each algorithm's performance is evaluated. Our similarity metric then quantifies the correlations across different algorithms. This metric helps identify the algorithms whose combination are more likely to generate superior performance over other combinations.

The six algorithms we have selected are: Holt-Winter forecasting [1], Cumulative Sum [2], Wavelet analysis [3], K-Means [4], Support Vector Machine (SVM) [5], and One-Class Neighbor Machine (OCNM) [6]. The first three algorithms detect anomalies based on time-series information and the latter three are all machine learning based algorithms.

Our evaluation results indicate that the taxonomy, the benchmark traffic models and our similarity metric together constitute a consistent and effective framework of identifying anomaly detectors that have distinct properties and whose combination could generate superior performance. Algorithms belonging to different categories demonstrate small values for the similarity metric while those in the same category often have high values. Similar algorithms tend to reinforce each other's decisions while dissimilar algorithms detect anomalies which are missed by others in the combination. Thus, combining algorithms with relatively low similarity values and high detection accuracy essentially results in a higher detection accuracy for the combination as compared to any of the individual algorithms in the combination.

Furthermore, our work yield the following interesting observations:

- The best performing single algorithm is Holt-Winter forecasting. The best aggregate algorithm is a combination of Holt-Winter forecasting and Cumulative Sum.
- Even when using multiple algorithms that have low detection accuracy, the aggregation of these algorithms can achieve high accuracy – in particular when combining algorithms whose detection characteristics are dissimilar (according to our similarity metric).
- Combining algorithms belonging to different classes can yield a higher detection accuracy.
- Combining more number of algorithms does not always result in a better performance. Infact, the combination can possible do worse than every algorithm in the combination.

The rest of the paper is structured as follows. Section II discusses related work in the area of anomaly detection. Section III describes a taxonomy of various existing anomaly detection algorithms. Section IV discusses how multiple anomaly detection algorithms can be combined to improve detection accuracy. The evaluation of algorithms with our anomaly detection benchmark is described in Section V. Results are discussed in Section VI. Our contributions and results are summarized in Section VII.

II. Related Work

We first give a brief list over some recently proposed traffic anomaly detectors. Brutlag [1] studied the technique of Holt-Winter forecasting. Wang et al. [2] applied Cumulative Sum algorithm to detect TCP SYN flooding attack. Barford et al. [3] exploited wavelet analysis to study high and medium variance in the traffic. Soule et al. [7] proposed Kalman-filter based algorithms. These all treat traffic as time series. There are also a number of approaches exploiting techniques from machine learning areas. For example, Munz et al. [4] studied K-Means algorithms, Lazarevic et al. [5] compares several machine learning techniques including Support Vector Machine (KVM). And Ahmed et al. [6] studied One-Class Neighbor Machine (OCNM) techniques. Besides these, Lakhina et al. [8] proposed the application of Principle Component Analysis (PCA). There are also anomaly detectors that based on traffic distribution or entropy, e.g. [9]–[11].

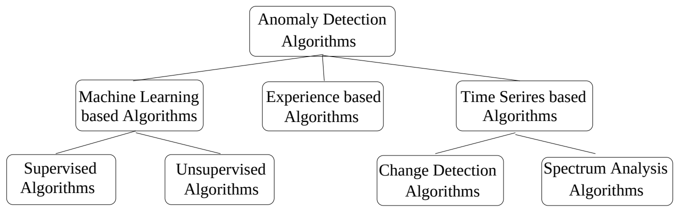

Anomaly Detection Algorithms
Machine Learning based Algorithms
Experience based Algorithms
Time Serires based Algorithms
Supervised Algorithms
Unsupervised Algorithms
Change Detection Algorithms
Spectrum Analysis Algorithms

Fig. 1. Taxonomy of anomaly detection algorithms

There have been some other proposed taxonomies for anomaly detection algorithms. Estevez-Tapiador et al. have classified anomaly detection based on the network feature that is analyzed, the underlying behavior model, and the scale at which traffic is analyzed [12]. However, their work does not provide a comparative evaluation of different algorithms. Qayyum et al. propose a taxonomy of anomaly detection algorithms in the context of intrusion detection systems [13]. Their work provides a qualitative evaluation of different approaches. The key aspect of our work is that we provide a quantitative evaluation of different approaches in a single evaluation framework.

Our previous work [14], [15] demonstrate that combining multiple detection algorithms does offer an increase in performance over individual detectors but did not consider the criteria on which the algorithm choices should depend. This paper builds on our previous work and deals with those criteria. Besides, Gao et al. [16] also propose using an ensemble of algorithms to build a more accurate model for continuously arriving data and proved theoretical improvement over each single algorithm. However, their work did not consider how to pick the best combination of algorithms.

In terms of traffic anomaly benchmarks, Mirkovic et al. [17] proposes a set of rules that can be used to specify characters of a Denial-of-Service attacks. These rules can be used as guidelines to simulate a Denial-of-Service attack. Compared with their work, we are more specific in defining the traffic anomaly models.

III. Taxonomy of Anomaly Detection Algorithms

In this section, we classify anomaly detection algorithms based on the underlying techniques exploited by these algorithms. By proper classification of these algorithms, we can have a better understanding of the nature of the results provided, which also helps determine different combinations of algorithms when they are combined in a parallel environment. **Machine learning based algorithms** treat traffic data as a set of (multi-dimensional) data points. Clustering algorithms such as K-Means, SVM are then applied on the data in an effort to sort anomaly traffic data out from benign underlying traffic. Based on whether pre-labeled data are used or not, machine learning based algorithms can further be classified into supervised algorithms and unsupervised algorithms. Supervised algorithms take advantage of human knowledge by obtaining information from labeled data. Unsupervised algorithms classify traffic data as anomalous or benign based on built-in metrics from the algorithm itself. On the other

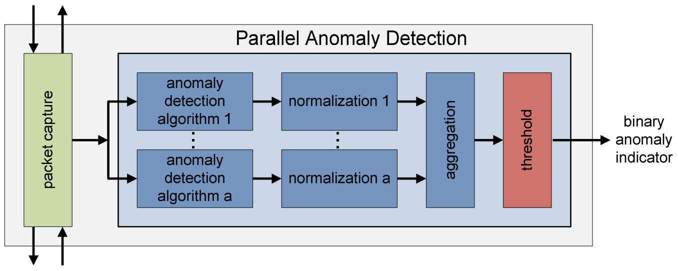


Fig. 2. System Architecture.

hand, noises in the traffic could lead to false positives as each data point is treated independently. **Time series based algorithms** treat traffic as a discrete or continuous function of time. This function is then examined, often, to locate sudden changes, which are acute deviations from past values of the function. These sudden changes are usually deemed to be signs of anomalies. Various techniques have been applied to detect these sudden changes including Holt-Winter, CUSUM. More complicated methods such as wavelet analysis has also been examined. The advantage of these approaches is that often correlations in time local data can be considered, which helps remove unnecessary false positives. However, it is often the case that time series based algorithms tend to ignore those ”slow-cooking” anomalies, i.e. where anomaly traffic volume increases very slowly over time. **Experience based algorithms** usually present themselves in the way of rule-based systems. In these systems, a set of rules is predefined. Often, these rules are based on expert knowledge and are hand crafted into the system. At each time point, the state of the system is used as input to these rules and each rule is checked to see whether it is satisfied. Each satisfied rule incurs certain actions among which an anomaly is identified. The advantage of experience based algorithms is that the rules can be fully customized to fit the requirement of the protected systems. However, defining the rules usually requires immense human labeling and going through all the rules is usually very computationally intensive.

In summary, we see that there is much diversity in anomaly detection algorithms. In the following we will focus on machine learning based algorithms and time series based algorithms. Since experience based algorithms usually involve a large amount of human labor, it can be a beneficial complement to an existing system.

IV. Anomaly Detection with Multiple Algorithms

A. System Outline

The architecture of our system is shown in Figure 2. The system keeps track of packet rate and bit rate observed on the monitored link. All anomaly detection algorithms process this rate and output an internal continuous anomaly metric. The anomaly metric output by the algorithms are further normalized by the normalization module. All the normalized anomaly metrics are then aggregated and a system wide threshold is applied to make a binary decision.

B. Aggregation of Anomaly Detection Algorithms

The aggregation process for multiple anomaly detection algorithms has been described in our prior work [14], [15]. We repeat some of this discussion in this subsection to ensure the reader is familiar with the general idea. For more details, see [14], [15].

When using multiple different anomaly detection algorithms, it is important to consider how outputs from different algorithms can be aggregated into a single metric. Existing algorithms typically yield a binary decision (i.e., a ‘1’ if an anomaly is detected and ‘0’ if not). One could aggregate such binary outputs by using majority decision or similar binary functions. However, we believe that such an approach is too coarse. Instead, we observe that practically all anomaly detection algorithms use continuous metrics internally, before applying a threshold and generating the binary output. Our system uses these continuous metrics for aggregation and applies the threshold as late as possible. Since different algorithms use different internal metrics, it is crucial to normalize these metrics before aggregation.

1) Notation: Our system processes traffic volume data (e.g., byte-rate, packet-rate) to determine anomalies. Let c^t be the packet count during the observation interval $[t, t+\tau)$. Parameter τ is assumed to be a preset fixed time interval that decides the granularity of the data. The interval can range from 1 second to a few hours. Let s be the total number of different traffic subsets S_i, $1 \leq i \leq s$ and a be the number of anomaly detection algorithms A_j, $1 \leq j \leq a$. Then, during the observation interval $[t, t+\tau)$, the algorithms use a metric $m_{i,j}^t$=$c_i^t/p_{i,j}^t$, where $p_{i,j}^t$ denotes the prediction produced by each algorithm A_j based on the history of packet counts $c_i^{t_k}$ ($t_k < t$) and c_i^t denotes the current packet count during the interval $[t, t+\tau)$ for subset S_i.

2) Normalization: The prediction $p_{i,j}^t$ depends on the characteristics of the algorithm j. Thus, normalization is necessary to ensure equal influence of each algorithm on the aggregate. We define a normalization function, N, that produces the normalized metric $n_{i,j}^t$, for algorithm j and subset i at time t. This function normalizes $m_{i,j}^t$ to the continuous interval $[0, 1]$, where 0 represents the condition of ”no anomaly” and 1 represents ”anomaly”. The normalization function is as follows:

$$n_{i,j}^t = N\left(m_{i,j}^t, \alpha_j\right) = \min(1, max(0, 0.5 \cdot \alpha_j \cdot m)). \quad (1)$$

Parameter α_j determines the slope of the normalization function, and is adjusted such that the boundary between the anomalies and no anomalies falls exactly in the middle of the range $[0, 1]$ at θ^*=0.5 for $\alpha_j^* = \frac{n^*}{\theta_j}$ (refer Figure 3). This also ensures we do not have to use a system specific threshold, and can use θ^*=0.5 to compare the final aggregated value with (Other normalization functions we have evaluated are $N(m, \alpha)$=max$(1 - e^{-\alpha \cdot \ln m}, 0)$, $N(m, s)$=$1 - e^{-|\alpha \cdot \ln m|}$, but they do not provide better performance (see [18])).

3) Aggregation and Anomaly Decision: Once the normalized anomaly metric $n_{i,j}^t$ is output by the normalization

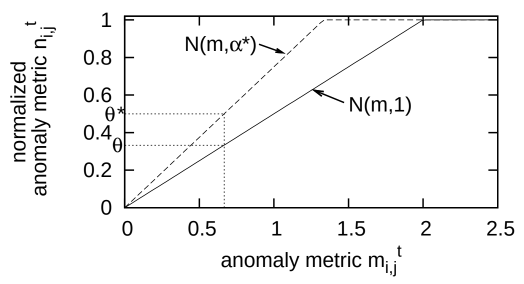


Fig. 3. Normalization Function $N(m, \alpha)$. α^* ensures that boundary between anomalies and no anomalies occurs at θ^*=0.5.

module for all algorithms, j, for a particular subset S_i, the aggregation module uses the aggregation function G to determine the final aggregated anomaly metric g_i^t across all a algorithms for subset i at time t. Some of the functions explored were arithmetic mean, geometric mean, median, minimum, and maximum. Evangelista et al. [19] proposed the average of mean and minimum as an aggregation function. However, for our purpose, the average of mean and maximum has proven to be most effective:

$$g_i^t = G\left(n_{i,1}^t, \ldots, n_{i,a}^t\right) = \frac{1}{2}\left(\frac{1}{a}\sum_{j=1}^{a} n_{i,j}^t + \max_j n_{i,j}^t\right). \quad (2)$$

The mean assigns equal weights to various algorithms in the combination independent of whether the algorithm has high accuracy or not. The max function selects the algorithm that is most sensitive and produces the highest normalized value at any particular instance. This combination of the mean and max functions results in an improved overall detection performance. Finally, the aggregated anomaly metric g_i^t is compared to a threshold, θ^*, to make a binary decision as follows:

$$r_i^t = \begin{cases} 0, & \text{if } g_i^t < \theta^* \\ 1, & \text{if } g_i^t \geq \theta^*. \end{cases} \quad (3)$$

The result is then reported to the user.

V. Evaluation of Anomaly Detection Algorithms

The experimental evaluation was carried out on the DETER testbed [20] which provides an infrastructure for conducting medium-scale security experiments. It provides researchers with traffic generation tools which can be used to generate both background and attack traffic with the desired distributions. The experimental setup is similar to the one used by Mirkovic et al. [21] consists of seven networks, four of which are client/server networks and three are attack networks. The networks are interconnected through four core routers. Each client/server network consists of three client nodes and four server nodes. All links have a bandwidth of 100Mbps and no delay. The server nodes run five services: Web, FTP, IRC, Telnet and DNS. Every client node generates a mix of request traffic targeted towards various servers in it's own as well as other networks. The background traffic generated has an average bitrate of 20Mbps. The SEER traffic generation tool [22] is used to generate two types of traffic: background and attack. To make sure the background traffic accurately models traffic in a typical network, the traffic parameters and distributionsas suggested by Mirkovic et al. [21] were used. SEER also includes a flooder tool that allows us to control the characteristics of the attacks like custom protocols, packet rates, sizes, varying rate characteristics and shapes. In all our experiments, all the attack nodes generate floods that target port 80 in one of the Web/DNS server nodes. The description of the attack scenarios follows.

TABLE I
Description of Scenarios

Scenario	Type/Intensity	On/Off Time	Attack/Total Duration (s)
1	Flat Pulses/High	60/60	200/300
2	Flat Pulses/Low	60/60	200/300
3	Flat Pulses/High	30/30	200/300
4	Flat Pulses/Low	30/30	200/300
5	Ramped Pulses/High	60/60	200/300
6	Ramped Pulses/Low	60/60	200/300
7	Ramped Pulses/High	30/30	200/300
8	Ramped Pulses/Low	30/30	200/300

To evaluate the system and the anomaly detection algorithms used, it is important that we target the system with different types of attacks. The anomaly detection benchmark consists of eight scenarios of different types. Table I summarizes the different scenarios with their characteristics. The objective of varying the pulse widths (on and off-times) has to do with the sensitivity of algorithms. Certain algorithms adapt to the network traffic rate and having a long pulse results in the detection of the start of the attack, but may result in a lot of false positives in the later part of the pulse as the algorithm adapts to the new traffic rate. Likewise, ramped attacks are harder to detect since the traffic increase is gradual and allows the attack to 'sneak' past any thresholds set. This is especially true in case of algorithms that make use of an adaptive threshold to detect attacks.

A. Evaluated Anomaly Detection Algorithms

The anomaly detection algorithms implemented in our prototype system are:

- **Holt-Winter Forecasting** builds a time series model that uses multiple levels of exponential smoothing to decompose the observed time series into three components: a baseline, a seasonal trend and a linear trend. (e.g., [1]).
- **CUSUM** belongs to one of the change point detection algorithms widely studied in the field of statistics. By tracing the distance between the current point with the point of the smallest value in history, it detects anomalies by comparing this distance with a predefined threshold. (e.g., [2]).
- **Wavelet** method decomposes the time series into three components, corresponding to low frequency signals, medium frequency signals and high frequency signals that can be used to reconstruct the original time series. The variability of the low frequency signals and that of the medium frequency signals are then combined using a weighted sum and are regarded as the variation part of

the original time series. A threshold is then applied to the variation part and anomalies are reported if the variation part deviates far beyond the threshold.

- **K-Means** is a clustering algorithm that classifies given data into K clusters where K is a predefined constant integer. The technique is usually very simple: Given a set of data points, K random points are selected. Then each data point is assigned to one of the selected points where minimum distance is obtained. This forms an initial classification of the data. Then for each cluster, the centroid of the cluster is calculated and each data point is again assigned to the nearest centroid. This process continues until the set of selected centroids does not change. In our case, K is set to 2 corresponding to anomaly traffic and benign traffic respectively. One of the clusters is labeled as anomaly and the other is labeled as benign. Then new data points are compared to the centroids of these two clusters. If it is closer to the anomaly cluster or is too far away from the benign cluster, it is labeled as an anomaly. This approach is studied in [4].
- **SVM** is also a supervised algorithm. It tries to separate two classes of multi-dimensional data by locating a hyperplane that maximizes its distances to the neighboring data points. There are various algorithms that targeting at solving the hyperplane for SVM. In our evaluation, we use that implemented by SVM-Light [23]. Once the hyperplane is obtained, new data points are then classified by checking which side of the hyperplane it belongs to.
- **OCNM** is an unsupervised algorithm. For a given set of data points, it calculates for each point its average distance to its K nearest neighbors. Then, these distances are sorted and the ones with large distances are deemed to be anomalies.

VI. Results

We start the presentation of results by discussing the accuracy of individual algorithms followed by a definition of the "similarity metric" that we use to compare algorithms. The algorithms are tuned to use the best possible internal parameters to achieve optimal performance. During the normalization and aggregation process, the best possible normalization function parameter α_j^* for each algorithm A_j is used. To better quantify and understand the accuracy of both individual algorithms and the aggregation of combinations of algorithms, we use their receiver operating characteristic (ROC) curves [24] which illustrates the sensitivity of a binary classifier for varying thresholds (i.e., θ). An area under the curve of 1.0 represents a perfect algorithm whereas an area of 0.5 represents an algorithm that bases its predictions on chance. Table IV lists the top three and bottom three AUC values for individual algorithms as well as pairs, 2-tuples, 3-tuples, 4-tuples and 5-tuples.

TABLE II
Two algorithms are said to agree with each other if they have the same binary output for a particular event. We then use the number of instances where both algorithms are right or wrong to calculate the similarity metric.

A1	A2	agree/disagree
0	0	agree
0	1	disagree
1	0	disagree
1	1	agree

A. Similarity Metric

Given a set of algorithms, how can we maximize the accuracy and what is the best combination of algorithms that achieves this? To better answer these questions, we need to understand how similar (or dissimilar) the algorithms are in terms of detection characteristics. This is important because combining algorithms without understanding how similar they are can affect the accuracy of the system in an adverse manner. Thus, we define our "Similarity Metric" as a "measure of how much two algorithms agree with each other". For example, consider two algorithms A1 and A2. A1 and A2 are said to agree with each other if the two algorithms have the same binary output for a particular event. The Truth Table II summarizes this. It is not necessary for the algorithms to agree with the event label. We then define the similarity metric for two algorithms A1 and A2 as the ratio of total number of instances where they agree to the total number of events. Formally, let X and Y be two binary sequences defined as: $X = (x_1, x_2, ...x_N)$ and $Y = (y_1, y_2, ...y_N)$ where, $x_i \in \{0,1\}$, $y_i \in \{0,1\}$ and $i = (1,2,3,...N)$. If $A_{i,j}$ where $(i,j \in \{0,1\})$, be the number of times X and Y are matching, i.e., $A_{0,0}$ is the number of occurrences of $X = 0$, $Y = 0$, $A_{0,1}$ of $X = 0$, $Y = 1$, and so on. Then, we define the "similarity metric" as:

$$Sim(X,Y) = \frac{A_{0,0} + A_{1,1}}{A_{0,0} + A_{0,1} + A_{1,0} + A_{1,1}} \quad (4)$$

Table III summarizes the pairwise similarity metrics for all algorithms. Note that Equation 4 can be extended to derive the similarity of any number of sequences. We will now look at how the similarity affects the accuracy of the aggregated anomaly metric for different combinations of algorithms.

B. Aggregate Algorithm Accuracy and Similarity Relationship

We evaluate the various pairs, 3-tuple, 4-tuple and 5-tuple combinations of the six algorithms implemented. Table IV summarizes the accuracy of these combinations in terms of the AUC metric for all the eight scenarios. The entries in **bold** are the best performers for the scenario whereas the underlined entries are the worst performers. Figures 4 are the ROC curves for individual algorithms, combinations of two, three and five algorithms respectively. The ROCs represent the accuracy of the algorithm combinations for Scenario 6 which is hard to detect as compared to other scenarios.

The best performing aggregate over all scenarios is the combination of Holt-Winters forecasting and Cumulative Sum

TABLE III
PAIRWISE SIMILARITY METRIC

Scenario	HC	HW	HK	HO	HS	CW	CK	CO	CS	WK	WO	WS	KS	KO	OS
1	0.987	0.977	0.589	0.589	0.987	0.977	0.595	0.589	1.0	0.592	0.585	0.977	0.595	0.431	0.589
2	0.977	0.933	0.799	0.645	0.973	0.957	0.823	0.642	0.983	0.813	0.612	0.946	0.813	0.532	0.632
3	0.953	0.742	0.555	0.709	0.592	0.769	0.535	0.729	0.605	0.452	0.605	0.589	0.582	0.472	0.508
4	0.99	0.746	0.786	0.642	0.589	0.749	0.796	0.639	0.592	0.659	0.602	0.609	0.582	0.515	0.559
5	0.967	0.91	0.676	0.747	0.923	0.896	0.696	0.776	0.936	0.619	0.686	0.833	0.659	0.498	0.766
6	0.987	0.933	0.9	0.786	0.873	0.946	0.913	0.773	0.866	0.906	0.746	0.853	0.826	0.699	0.793
7	0.92	0.769	0.625	0.756	0.605	0.769	0.692	0.803	0.585	0.555	0.672	0.595	0.559	0.508	0.548
8	0.957	0.622	0.849	0.766	0.619	0.625	0.873	0.783	0.609	0.599	0.528	0.535	0.575	0.662	0.538

(denoted as HC), with an average of 0.97727 across all scenarios. The HC combination also demonstrates a high similarity (Table III) for all scenarios. HC performs the best essentially due to the high accuracy and similarity of the algorithms involved. A high similarity value translates to a high accuracy as calculated by the Equation 2 as the algorithms involved reinforce each other. Observe that the combination outperforms both Holt-Winter and Cumulative Sum in all eight scenarios. This is true in most combinations.

The combination HCK outperforms all combinations in Scenarios 1 and 7. The HCK combination has a low similarity value and outperforms the other combinations because of this dissimilarity, the reason being, an anomaly missed by one algorithm may be detected by another in the combination. This also reiterates the importance of the class to which the anomaly detection algorithm belongs. Holt-Winters and Cumulative Sum belong to change-point detection class whereas K-Means is a supervised machine learning algorithm. The two classes behave differently to different traffic scenarios and are thus expected to have low similarity values.

C. Conclusions

Our results yield the following interesting observations:

- The best performing single algorithm is Holt-Winter forecasting with a peak accuracy of 0.99515.
- The best aggregate algorithm is a combination of Holt-Winter forecasting and Cumulative Sum which has a peak accuracy of 1.0000. Thus, a combination of algorithms has a better accuracy as compared to a single algorithm. This is on account of a high similarity and high detection accuracy for both algorithms. Thus, they reinforce each other and hence the high accuracy of the combination.
- Aggregation of algorithms can achieve a high accuracy even when some of the algorithms that are being aggregated have a low detection accuracy.
- Similar algorithms reinforce each other. Aggregating similar algorithms which are also sufficiently accurate can result in a higher detection accuracy of the combination when compared to the individual algorithms.
- Algorithms belonging to different classes of anomaly detection algorithms may exhibit low similarity values because of different performance characteristics. Aggregating algorithms that are dissimilar but have sufficiently high accuracy can result in a much higher detection accuracy compared to any algorithm in the combination. This is primarily because an anomaly missed by one algorithm may be detected by another algorithm in the combination.
- Using more number of algorithms does not always result in better accuracy. For example, in Scenario 4 (Table IV), HCKOS has an accuracy of 0.8808 compared to HC with an accuracy of 0.99997.

VII. SUMMARY

We identified different types of anomaly detection algorithms based on the technique used for classifying network traffic and devised a method to aggregate information from anomaly detection algorithms belonging to different classes to create a single aggregate decision. Furthermore, we discussed a benchmark for evaluating anomaly detection algorithms on a real network testbed. We implemented and evaluated different combinations of six different anomaly detection algorithms to determine which combinations yield the most accurate anomaly detector. We also defined a similarity metric that can be used to compare algorithms. The results of the evaluation confirm that a combination of algorithms has a better detection accuracy than any single algorithm. Also, accurate anomaly detection can be achieved not only by using more accurate algorithms but also by increasing the diversity of algorithms.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0325868.

REFERENCES

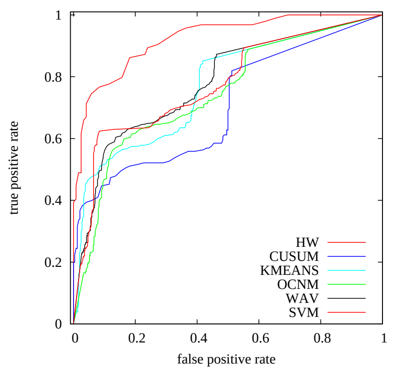
[1] J. D. Brutlag, "Aberrant behavior detection in time series for network monitoring," in *Proc. of the 14th Systems Administration Conference*, New Orleans, LA, Dec. 2000, pp. 139–146.

[2] H. Wang, D. Xhang, and K. G. Shin, "Change-point monitoring for the detection of dos attacks," *IEEE Transactions on Dependable Secure Computing*, vol. 1, no. 4, pp. 193–208, Oct. 2004.

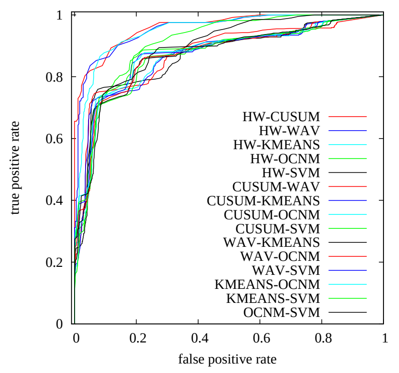
[3] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *Proc. of the 2nd ACM SIGCOMM Workshop on Internet Measurement (IMW)*, Marseille, France, Nov. 2002, pp. 71–82.

[4] G. Mϋnz, S. Li, and G. Carle, "Traffic anomaly detection using k-means clustering," in *Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und Verteilten Systemen, 4. GI/ITG-Workshop MMBnet*, September 2007.

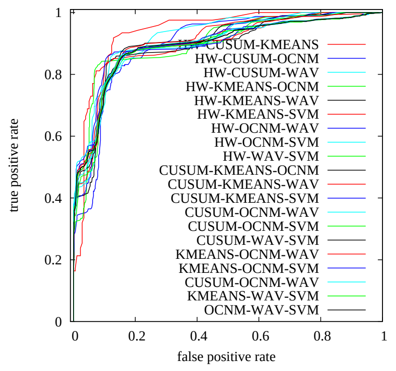
[5] A. Lazarevic, A. Ozgur, L. Ertoz, J. Srivastava, and V. Kumar, "A comparative study of anomaly detection schemes in network intrusion detection," in *In Proceedings of the Third SIAM International Conference on Data Mining*, 2003.



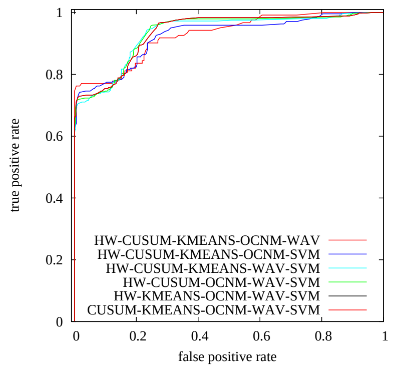
(a) Individual algorithms.



(b) Combinations of two algorithms



(c) Combinations of three algorithms



(d) Combinations of five algorithms

Fig. 4. Detection performance of various combinations for Scenario 6.

[6] T. Ahmed, B. Oreshkin, and M. Coates, “Machine learning approaches to network anomaly detection,” in *the Second Workshop on Tackling Computer Systems Problems with Machine Learning (SysML)*, April 2007.

[7] A. Soule, K. Salamatian, and N. Taft, “Combining filtering and statistical methods for anomaly detection,” in *IMC'05: Proceedings of the Internet Measurement Conference 2005 on Internet Measurement Conference*. Berkeley, CA, USA: USENIX Association, 2005, pp. 31–31.

[8] A. Lakhina, M. Crovella, and C. Diot, “Diagnosing network-wide traffic anomalies,” in *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2004, pp. 219–230.

[9] ——, “Mining anomalies using traffic feature distributions,” *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 217–228, 2005.

[10] Y. Gu, A. McCallum, and D. Towsley, “Detecting anomalies in network traffic using maximum entropy estimation,” in *IMC '05: Proceedings of the 5th ACM SIGCOMM conference on Internet measurement*, 2005, pp. 1–6.

[11] K. Xu, Z.-L. Zhang, and S. Bhattacharyya, “Profiling internet backbone traffic: behavior models and applications,” in *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. New York, NY, USA: ACM, 2005, pp. 169–180.

[12] J. M. Estevez-Tapiador, P. Garcia-Teodoro, and J. E. Diaz-Verdejo, “Anomaly detection methods in wired networks: a survey and taxonomy,” *Computer Communications*, vol. 27, no. 16, pp. 1569–1584, Oct. 2004.

[13] A. Qayyum, M. Islam, and M. Jamil, “Taxonomy of statistical based anomaly detection techniques for intrusion detection,” in *Proc. of the IEEE Symposium on Emerging Technologies*, Islamabad, Pakistan, Sep. 2005, pp. 270–276.

[14] S. Shanbhag and T. Wolf, “Massively parallel anomaly detection in online network measurement,” in *Proc. of Seventeenth IEEE International Conference on Computer Communications and Networks (ICCCN)*, St. Thomas, USVI, Aug. 2008.

[15] ——, “Accurate anomaly detection through parallelism,” *Netwrk. Mag. of Global Internetwkg.*, vol. 23, no. 1, pp. 22–28, 2009.

[16] J. Gao, W. Fan, and J. Han, “On appropriate assumptions to mine data streams: Analysis and practice,” *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pp. 143–152, Oct. 2007.

[17] J. Mirkovic, E. Arikan, S. Wei, S. Fahmy, R. Thomas, and P. Reiher, “Benchmarks for ddos defense evaluation,” *Military Communications Conference, 2006. MILCOM 2006*, pp. 1–10, Oct. 2006.

[18] T. Wolf and S. Shanbhag, “Massively parallel anomaly detection system,” University of Massachusetts, Amherst, MA, Tech. Rep. TR-08-CSE-08, Mar. 2008.

[19] P. F. Evangelista, M. J. Embrechts, and B. K. Szymanski, “Data fusion for outlier detection through pseudo-ROC curves and rank distributions,” in *Proc. of International Joint Conference on Neural Networks (IJCNN)*, Vancouver, BC, Jul. 2006, pp. 2166–2173.

[20] T. Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab, “Design, deployment, and use of the DETER testbed,” in *DETER: Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007*, Boston, MA, 2007.

[21] J. Mirkovic, A. Hussain, B. Wilson, S. Fahmy, P. Reiher, R. Thomas, W.-M. Yao, and S. Schwab, “Towards user-centric metrics for denial-of-service measurement,” in *ExpCS '07: Proceedings of the 2007 Workshop on Experimental Computer Science*, San Diego, CA, Jun. 2007.

[22] S. Schwab, B. Wilson, C. Ko, and A. Hussain, “SEER: a security experimentation environment for DETER,” in *DETER: Proceedings of the DETER Community Workshop on Cyber Security Experimentation and Test on DETER Community Workshop on Cyber Security Experimentation and Test 2007*, Boston, MA, 2007.

[23] “Svm light.” [Online]. Available: http://svmlight.joachims.org/

[24] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, Jun. 2006.

TABLE IV
PERFORMANCE OF VARIOUS COMBINATIONS OF ALGORITHMS FOR SCENARIOS 1 TO 8

Scenario 1	**HIGH** INTENSITY **LONG** ON **LONG** OFF FLAT PULSES				
Ranking	Combinations of algorithms				
	1	2	3	4	5
Top 3	H 0.9937	HW 0.99862	**HCK 0.99990**	HCKW 0.99838	HCKOW 0.99983
	C 0.90065	HC 0.99819	HCS 0.99790	HCKS 0.99828	HCKOS 0.99890
	K 0.87769	HK 0.99741	HCW 0.99790	HCOW 0.99821	HCOWS 0.99889
Bottom 3	W 0.87308	KO 0.95886	HWS 0.97133	CKOS 0.98734	HCKWS 0.99880
	S 0.86719	KS 0.95247	CKO 0.97020	HOWS 0.98697	CKOWS 0.99762
	O 0.81607	OS 0.94071	HOS 0.96584	HKOS 0.98612	HKOWS 0.99724
Scenario 2	**LOW** INTENSITY **LONG** ON **LONG** OFF FLAT PULSES				
Ranking	Combinations of algorithms				
	1	2	3	4	5
Top 3	H 0.99453	**HC 0.99811**	HCS 0.94417	HCKW 0.94675	HCKOW 0.98826
	K 0.89576	HW 0.98825	HCW 0.93806	KOWS 0.94602	CKOWS 0.96605
	W 0.88659	HK 0.97322	HCK 0.93432	COWS 0.94517	HKOWS 0.96287
Bottom 3	S 0.86650	KS 0.92390	HOS 0.91504	HCKS 0.92488	HCOWS 0.96035
	C 0.86015	CO 0.92070	HKO 0.91496	HCOS 0.91808	HCKWS 0.95452
	O 0.84883	OS 0.91362	HCO 0.87663	HCKO 0.84671	HCKOS 0.91866
Scenario 3	**HIGH** INTENSITY **SHORT** ON **SHORT** OFF FLAT PULSES				
Ranking	Combinations of algorithms				
	1	2	3	4	5
Top 3	C 0.99430	**HC 1**	HCK 0.99727	HCKO 0.99920	HCKOW 0.99766
	H 0.98705	HW 0.98130	HCO 0.99318	HCKW 0.99268	HCKOS 0.99765
	K 0.87415	HK 0.96537	HCS 0.99312	HCKS 0.99128	HCKWS 0.99310
Bottom 3	W 0.85722	KO 0.92659	KOS 0.95191	CKOW 0.96957	HCOWS 0.99273
	S 0.83912	KS 0.91381	KWS 0.94877	HOWS 0.96817	CKOWS 0.98458
	O 0.83696	OS 0.90533	OWS 0.94428	KOWS 0.96762	HKOWS 0.98379
Scenario 4	**LOW** INTENSITY **SHORT** ON **SHORT** OFF FLAT PULSES				
Ranking	Combinations of algorithms				
	1	2	3	4	5
Top 3	H 0.99515	**HC 0.99997**	HCK 0.96346	HCKW 0.89498	HCKOW 0.90819
	C 0.95172	HK 0.93411	HCS 0.91038	HCKS 0.89383	HCKWS 0.88805
	K 0.94673	HW 0.92880	HCW 0.89541	HCWS 0.88270	HCOWS 0.88740
Bottom 3	O 0.87991	KS 0.85662	KWS 0.86045	CKOS 0.86945	CKOWS 0.88542
	W 0.87356	KO 0.85538	KOS 0.85931	KOWS 0.86835	HKOWS 0.88462
	S 0.83717	OS 0.84467	OWS 0.85557	HCKO 0.86790	HCKOS 0.88080
Scenario 5	**HIGH** INTENSITY **LONG** ON **LONG** OFF RAMPED PULSES				
Ranking	Combinations of algorithms				
	1	2	3	4	5
Top 3	H 0.94368	**HC 0.97511**	HCK 0.96835	HCKO 0.95203	CKOWS 0.95142
	C 0.84907	HW 0.96033	HCW 0.93915	HCKW 0.94945	HKOWS 0.95081
	K 0.83953	HK 0.95975	HKW 0.93553	HCKS 0.94407	HCKOW 0.95075
Bottom 3	W 0.82437	CO 0.90549	COW 0.92457	CKOW 0.93931	HCOWS 0.95059
	S 0.81466	CW 0.90322	COS 0.92429	HKOS 0.93794	HCKWS 0.95037
	O 0.80931	OS 0.90196	HOS 0.92368	HCOS 0.93688	HCKOS 0.94933
Scenario 6	**LOW** INTENSITY **LONG** ON **LONG** OFF RAMPED PULSES				
Ranking	Combinations of algorithms				
	1	2	3	4	5
Top 3	H 0.92355	**HC 0.96143**	HCK 0.93877	HCKW 0.92799	HKOWS 0.93615
	W 0.77934	HW 0.95543	HCW 0.91836	HCKO 0.92365	CKOWS 0.93607
	S 0.77483	HK 0.94859	HKW 0.90519	KOWS 0.92081	HCOWS 0.93582
Bottom 3	K 0.76896	CK 0.86986	COS 0.89366	HCKS 0.91243	HCKWS 0.93371
	O 0.74587	CO 0.86729	COW 0.89366	HCOW 0.91142	HCKOW 0.92800
	C 0.70428	CS 0.86664	HOS 0.89306	HCOS 0.90728	HCKOS 0.92503
Scenario 7	**HIGH** INTENSITY **SHORT** ON **SHORT** OFF RAMPED PULSES				
Ranking	Combinations of algorithms				
	1	2	3	4	5
Top 3	H 0.92236	HC 0.94772	**HCK 0.96217**	HCKO 0.95013	HCKOW 0.94463
	C 0.89591	HK 0.92133	HCO 0.94027	HCKW 0.94747	HCKWS 0.94394
	K 0.85205	HW 0.91620	HCW 0.93449	HCKS 0.94165	HCKOS 0.94385
Bottom 3	W 0.84833	WS 0.89546	KWS 0.91734	KOWS 0.93125	HCOWS 0.94131
	O 0.82788	KS 0.89102	HWS 0.91580	CKOW 0.93117	CKOWS 0.94031
	S 0.82476	OS 0.88710	OWS 0.91509	HOWS 0.93008	HKOWS 0.93986
Scenario 8	**LOW** INTENSITY **SHORT** ON **SHORT** OFF RAMPED PULSES				
Ranking	Combinations of algorithms				
	1	2	3	4	5
Top 3	H 0.92386	**HC 0.93784**	HCK 0.91951	HCKO 0.92546	HCKOS 0.90979
	K 0.81126	HK 0.85279	HCO 0.89402	HCKW 0.88010	HCKOW 0.89914
	O 0.79590	HO 0.84497	HCW 0.86030	HCKS 0.87888	HCKWS 0.87727
Bottom 3	W 0.77869	KO 0.79935	KOW 0.83350	COWS 0.85360	CKOWS 0.87621
	S 0.75654	WS 0.79008	CWS 0.83283	CKOW 0.85323	HKOWS 0.87286
	C 0.75585	CW 0.78680	OWS 0.83149	HOWS 0.85007	HCOWS 0.87217