Computing and Informatics, Vol. 33, 2014, 1–34

TOWARDS AN UNSUPERVISED METHOD FOR NETWORK ANOMALY DETECTION IN LARGE DATASETS

Monowar Hussain Bhuyan, Dhruba K. Bhattacharyya

Department of Computer Science and Engineering
Tezpur University
Napaam, Tezpur-784028
Assam, India
e-mail: {mhb, dkb}@tezu.ernet.in

Jugal K. Kalita

Department of Computer Science
University of Colorado at Colorado Springs
CO 80933-7150, USA
e-mail: jkalita@uccs.edu

Abstract. In this paper, we present an effective tree based subspace clustering technique (TreeCLUSS) for finding clusters in network intrusion data and for detecting known as well as unknown attacks without using any labelled traffic or signatures or training. To establish its effectiveness in finding the appropriate number of clusters, we perform a cluster stability analysis. We also introduce an effective cluster labelling technique (CLUSSLab) to label each cluster based on the stable cluster set obtained from TreeCLUSS. CLUSSLab is a multi-objective technique that employs an ensemble approach for labelling each stable cluster generated by TreeCLUSS to achieve high detection rate. We also introduce an effective unsupervised feature clustering technique to identify the dominating feature set from each cluster. We evaluate the performance of both TreeCLUSS and CLUSSLab using several real world intrusion datasets to identify known as well as unknown attacks and find that results are excellent.

Keywords: Cluster, unsupervised, cluster stability, ensemble, anomaly detection

1 INTRODUCTION

Advances in networking technology have enabled us to connect distant corners of the globe through the Internet for sharing vast amounts of information. However, along with this advancement, the threat from spammers, attackers and criminal enterprises is also growing at multiple speed [1]. As a result, security experts use intrusion detection technology to keep secure large enterprise infrastructures. Intrusion detection systems (IDSs) are divided into two broad categories: misuse detection [2] and anomaly detection [3] systems. Misuse detection can detect only known attacks based on available signatures. Thus, dynamic signature updating is important and therefore, new attack definitions are frequently released by IDS vendors. However, misuse based systems cannot incorporate most or even all of the rapidly growing number of vulnerabilities and exploits. On the other hand, anomaly based detection systems are designed to capture any deviation from profiles of normal behavior. They are more suitable than misuse detection systems for detecting unknown or novel attacks without any prior knowledge. However, they normally generate a large number of false alarms.

There are three commonly used approaches for detecting intrusions [4, 5]:

1. supervised (i.e., both normal and attack instances are used for training),
2. semi-supervised (i.e., only normal instances are used for training) and
3. unsupervised (i.e., without using any prior knowledge).

The first two cases require training on the instances for finding anomalies; but getting a large amount of labelled normal and attack training instances may not be feasible for a particular scenario. In addition, generating a set of true normal instances with all the variations is an extremely difficult task. Hence, unsupervised network anomaly detection, which does not require any prior knowledge of network traffic instances, is more suitable in this situation.

1.1 Motivation

To overcome obstacles faced by supervised and semi-supervised network anomaly detection methods, unsupervised network anomaly detection methods aim to detect known as well as unknown intrusions without using any prior knowledge of existing network traffic instances. Clustering is an established unsupervised network anomaly detection technique that can be used to identify unknown attacks. However, a common limitation of some clustering approaches is that they require the number of clusters a priori, which often can be difficult to provide. In such cases, stability analysis of the cluster results can be of great help. Validity of the cluster results in terms of real life and benchmark datasets is important to establish the effectiveness of the results. In high-dimensional data, many features are irrelevant to form a specific set of clusters when a full space clustering technique is applied. These are the reasons why we develop an unsupervised method for identification of known and unknown attacks with minimum false alarms.

1.2 Contributions

We aim to provide an unsupervised solution for identifying network attacks with high detection rate. The main contributions of this paper are stated below.

- We introduce a tree based clustering technique (TreeCLUSS) to identify network anomalies in high dimensional datasets. The following are some of the advantages of the proposed TreeCLUSS algorithm.
  - The number of clusters is not required as input parameters.
  - It is free from the use of a specific proximity measure.
  - It requires a minimum number of input parameters and the results are not heavily dependent on them.
  - It is able to identify both known as well as unknown attacks.
- We present a cluster stability analysis to obtain a stable set of results generated by TreeCLUSS. It uses majority voting based decision for cluster stability to get a stable set of clusters.
- We introduce a cluster labelling technique (CLUSSLab) for labelling the clusters generated by TreeCLUSS as normal or attack. It uses a majority voting based decision fusion technique of the results of various cluster indices, cluster sizes and dominating features sets.
- Finally, we develop an effective unsupervised feature clustering technique to identify a dominating feature subset for each stable cluster that is used for cluster labelling. It is important to identify a relevant feature set for a particular set of clusters to match with a previously identified feature set during cluster labelling.

1.3 Organization of the Paper

The rest of the paper is organized as follows. Section 2 provides a review of existing unsupervised network anomaly detection methods. The problem formulation is introduced in Section 3. Section 4 describes our unsupervised network anomaly detection framework in two parts: TreeCLUSS and CLUSSLab. Section 5 describes experimental results and comparison with competing algorithms. Finally, concluding remarks are presented in Section 6.

2 RELATED WORK

The problem of unsupervised detection of network attacks and intrusions has been studied for many years with the goal of identifying unknown attacks in high-speed network traffic data. Most network-based intrusion detection systems (NIDSs) are misuse- or signature-based. For example, SNORT [6] and BRO [7] are two well-known open source misuse-based NIDS. To overcome the inability of such systems to detect unknown attacks, novel anomaly-based NIDSs have been introduced in the past decade. A detailed study can be found in [8, 9]. Here, we briefly discuss some recent unsupervised network anomaly detection methods.

2.1 Clustering-Based Network Anomaly Detection

Clustering is an important technique used in unsupervised network intrusion detection. A majority of unsupervised network anomaly detection techniques are based on clustering and outlier detection [10, 11, 12]. Leung and Leckie report a grid-based clustering algorithm to achieve reduced computational complexity [11]. An unsupervised intrusion detection method by computing cluster radius threshold (CBUID) is proposed by [13]. The authors claim that CBUID works in linear time with respect to the size of datasets and the number of features. Song et al. report an unsupervised auto-tuned clustering approach that optimizes parameters and detects changes based on unsupervised anomaly detection for identifying unknown attacks [14]. Noto et al. present a new semi-supervised anomaly detection method (FRaC) [15] that builds an ensemble of feature models based on normal instances, and then identifies instances that disagree with these models as anomalous. Casas et al. present a novel unsupervised outlier detection approach based on combining subspace clustering and multiple evidence accumulation to detect several kinds of intrusions [16]. They evaluate the method using KDDcup99 and two other real-time datasets.

2.2 Cluster Stability Analysis

Several cluster stability analysis techniques have been proposed in the literature [17, 18, 19, 20]. We analyze cluster stability for identifying the actual number of clusters generated by our clustering algorithm using stability calculation. Lange et al. introduce a cluster stability measure to validate clustering results [17]. It determines the number of clusters by minimizing the classification risk of their measure. An experimental analysis of cluster stability measures for the identification of the number of clusters is discussed by [18]. Ben-David et al. provide a formal definition of cluster stability with specific properties [19]. They conclude that stability can be determined based on the behavior of the objective function. If the objective function is a unique global optimizer, the algorithm is stable. Das and Sil also present a cluster validation method for stable cluster generation using stability analysis [20].

2.3 Cluster Labelling

Cluster labelling is a challenging issue in unsupervised network anomaly detection. Most common cluster validity measures are summarized in [21, 22, 23]. Validity measures are usually based on internal and external properties of clustering results. Normally, internal validity measures obtain the compactness, connectedness and separation of the cluster partitions. External validity measures assess agreement between a new clustering solution and the reference clusters: the measure of interest to us is the approach by [21]. Jun [23] presents an ensemble method for cluster analysis. It uses a simple voting mechanism for making decision from the results obtained by using several cluster validity measures. Labelling of a cluster is a must in case of cluster-based unsupervised network anomaly detection. Our proposed cluster labelling technique works based on the cluster size, compactness and the dominating feature set.

2.4 Discussion

We provide a generic comparison of some published papers on network anomaly detection [10, 11, 13, 12, 14, 16, 15] in Table 1. Based on a review of existing techniques for clustering-based anomaly detection, cluster stability analysis and cluster labelling, we observe the following.

- Although many clustering-based network intrusion detection techniques have been reported in the literature [10, 11, 13, 12], only a few have full features of an unsupervised intrusion detection system [13]. Many methods use only clustering techniques for network anomaly detection without having cluster labelling strategies. Hence, there is still room to develop a full-featured unsupervised network anomaly detection technique.
- Existing stability analysis techniques have been mostly applied to analyze non-intrusion data; but network traffic data is high-dimensional and voluminous. Thus, there is scope for further enhancement in the network anomaly detection domain.
- Only a very few labelling techniques are available in the literature [21, 22, 23]. An appropriate use of indices can help in developing an effective labelling technique, which can support unsupervised anomaly detection to a great extent.

Due to these reasons, we see an opportunity to develop an integrated unsupervised network anomaly detection method.

3 PROBLEM FORMULATION

Our work analyzes large amounts of network traffic data over an optimal and relevant feature space without any prior knowledge to identify anomalous or non-conforming test instance(s) with minimum false alarm. The problem is defined as follows. Let D be a collection of network traffic data with n data objects, where each object has f features. The problem is to analyze D over an optimal and relevant feature subspace n_f, where $1 \leq n_f \leq f$ to identify groups of similar instances, C_i, where each C_i is labeled either as normal or anomalous.

The proposed method works in two phases:

1. TreeCLUSS creates k clusters, i.e., $C_1, C_2, \ldots, C_k$ from dataset D using a subset of relevant features, n_f, where each C_i is evaluated in terms of stability by using the function StableCLUSS, and

Author(s)	Method	Offline/ Online	Packet/ Flow level	Data Type	Unknown attack handled	Detection criteria	Full/Reduced space
Portnoy et al. [10], 2001	Clustering based	offline	packet	numeric	yes	cluster size, distance	Full
Leung and Leckie [11], 2005	Clustering based	offline	packet	numeric	no	distance, boundary value	Full
Jiang et al. [13], 2006	Clustering based	offline	packet	categorical	yes	distance	Full
Bhuyan et al. [12], 2011	Outlier based	offline	packet	numeric	yes	distance	Full
Song et al. [14], 2011	Clustering based	offline	packet	numeric	yes	distance	Full
Casas et al. [16], 2012	Clustering based, UNIDS	offline	flow	numeric	yes	distance	Reduced
Noto et al. [15], 2012	Model based	offline	other	numeric	no	distance	Full

Table 1. Unsupervised network anomaly detection methods: a comparison

2. CLUSSLab labels each cluster, C_i based on the two assumptions:
   (a) The majority of network connections are normal, and
   (b) Intra-similarity among the attack traffic instances is high.

CLUSSLab exploits cluster size, compactness, dominating feature subset and outlier scores to label each cluster.

4 UNSUPERVISED NETWORK ANOMALY DETECTION: THE FRAMEWORK

The main aim of this work is to detect network anomalies using an unsupervised approach with a minimum amount of false alarms. It can detect network anomalies without relying on existing signatures, training or labeled data. The proposed approach runs in two consecutive phases for analyzing network traffic in contiguous time slots of fixed length. Figure 1 provides a conceptual framework of the proposed anomaly detection method.

In the first phase, we introduce a tree based subspace clustering technique (TreeCLUSS) for generating clusters in high-dimensional large datasets. It is well known that network intrusion dataset is high-dimensional and large. We apply our technique over a subset of features. TreeCLUSS uses the MMIFS technique [24] for finding a highly relevant feature set. It uses a subset of features during cluster formation while not using any class labels. We analyze the stability of the cluster results obtained. Cluster stability analysis for real life data is not a trivial task. It is performed using an ensemble of several index measures, viz., Dunn index [25], C-index (C) [26], Davies Bouldin index (DB) [27], Silhouette index (S) [28] and Xie-Beni index (XB) [29]. We choose a stable set of clusters when a certain num-

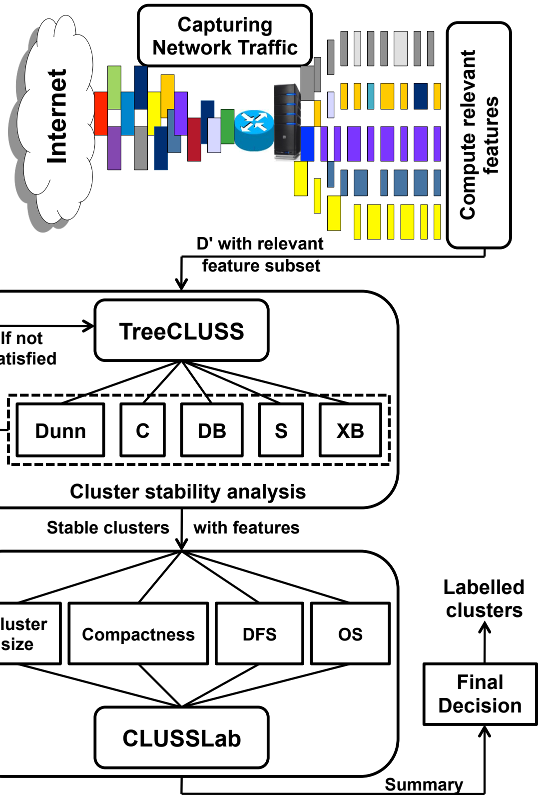

Figure 1. High-level description of the unsupervised network anomaly detection method

ber of clusters produces better result after multiple execution of this module. In the second phase, we apply a cluster labelling technique (CLUSSLab) to label the stable clusters using a multi-objective approach. CLUSSLab takes into account the following features: cluster size, compactness obtained from the ensemble of five index measures, dominating feature subset (DFS) obtained for each cluster based on unsupervised feature clustering technique discussed in Section 4.3, and outlier score (OS) obtained based on the RODD technique [42]. Finally, we label each cluster as normal or anomalous based on the described measures. The symbols used to describe the unsupervised network anomaly detection method are given in Table 2.

4.1 TreeCLUSS: The Clustering Technique

TreeCLUSS is a tree based subspace clustering technique for high-dimensional data. It is especially tuned for unsupervised network anomaly detection. It uses the MMIFS technique [24] to identify a subset of relevant features. TreeCLUSS de-

Symbol	Meaning
D	dataset
n	total number of data objects
C	set of clusters
f	feature set
sim	proximity measure between two objects O_i and O_j
α	threshold for L_1 cluster
β	threshold for L_2 cluster
γ	threshold for class-specific feature selection
ε	a factor for step down ratio
k	number of clusters
l	level
x_i	i^{th} data object
θ	height of the tree
n_f	total number relevant features
$minRank_f$	minimum rank value found w.r.t. MMIFS algorithm [24]
N_i	i^{th} node in tree
CL	class label
P	matching probability of dominant feature set

Table 2. Symbols used

pends on two parameters, viz., initial node formation threshold (α) and a step down ratio (ε) to extend the initial node, depth-wise. Both parameters are computed using a heuristic approach. We now present notations, definitions and a lemma which help in the description of the TreeCLUSS algorithm.

Definition 1 (Data Stream)**.** A data stream D is denoted as $\{O_1, O_2, O_3 \ldots, O_n\}$ with n objects, where O_i is the i^{th} object described with a d-dimensional feature subset, i.e., $O_i = \{x_{i1}, x_{i2}, x_{i3}, \ldots, x_{id}\}$.

Definition 2 (Neighbor of an object)**.** An object O_i is a neighbor of O_j over a subset of relevant features f, w.r.t. a threshold α, iff $sim_f(O_i, O_j) \leq \alpha$, where sim is a distance measure.

Definition 3 (Connected objects)**.** If object O_i is a neighbor of object O_j and O_j is a neighbor of O_k w.r.t. α, then O_i, O_j, O_k are connected.

Definition 4 (Node)**.** A node N_i in the l^{th} level of a tree is a non-empty subset of objects x', where for any object $O_i \in N_i$ there must be another object $O_j \in x'$, which is a neighbor of O_i, and O_i is either a) itself an initiator object or b) is within the neighborhood of another initiator object $O_j \in N_i$.

Definition 5 (Degree of a node)**.** The degree of a node N_j w.r.t. α is defined as the number of objects in N_j that are within α-neighborhood of any object $O_j \in N_j$.

Definition 6 ($L_{1,i}^{f,\alpha}$ cluster)**.** It is a set of connected objects C_i at level 1 w.r.t. α, where for any two objects $O_i, O_j \in C_i$, the neighbor condition (Definition 2) is true with reference to f_i.

Definition 7 ($L_{2,i}^{f,\beta}$ cluster)**.** It is a set of connected objects C_j at level 2 w.r.t. β, where for any two objects $O_i, O_j \in C_j$ the neighbor condition (Definition 2) is true with reference to f_i and $\beta \leq (\frac{\alpha}{2} + \varepsilon)$. Also, $L_{2,i}^{f,\beta} \subseteq L_{1,i}^{f,\alpha}$.

Definition 8 (Outlier)**.** An object $O_i \in D$ is an outlier if O_i is not connected with any other object $O_j \in D$, where $O_j \in L_{1,i}^{f,\alpha}$. In other words, O_i is an outlier if there is no $O_j \in D$, so that O_i and O_j are neighbors (as per Definition 2).

Lemma 1. Two objects O_i and O_j belonging to two different nodes are not similar.

Proof. Let $O_i \in N_i$, $O_j \in N_j$ and O_i be a neighbor of O_j. According to Definition 2 and Definition 4, O_i and O_j should belong to same node. Therefore, we come to a contradiction and hence the proof. □

We present our TreeCLUSS algorithm for network anomaly detection in Algorithms 1 and 2. TreeCLUSS starts by creating a tree structure in a depth-first manner with an empty root node. The root is at level 0 and is connected to all the nodes in level 1. The nodes in level 1 are created based on a maximal subset of relevant features by computing proximity within a neighborhood w.r.t. an initial cluster formation threshold α. The tree is extended depth-first by forming lower level nodes w.r.t. $(\frac{\alpha}{2} + \varepsilon)$, where ε is a controlling parameter of the step down factor, i.e. $\frac{\alpha}{2}$. α and ε are computed using a heuristic approach. A proximity measure *sim* is used in TreeCLUSS during cluster formation. Although *sim* is free from the restriction of using a specific proximity measure, we used Euclidean distance to construct the tree from D.

The algorithm is illustrated using an example. Let D be a dataset of d dimensions with details given in Table 3. Let $D = \{O_1, O_2, \ldots, O_{16}\}$ and $f = \{f_1, f_2, \ldots, f_{10}\}$. The extracted relevant feature set is given in Table 4. The class specific relevant features are identified from D w.r.t. a threshold γ. We achieved best results when $\gamma \geq 1$ for class C_1, $\gamma \geq 0.918$ for class C_2 and $\gamma \geq 0.917$ for class C_3, as shown in Table 4. An example tree obtained from D is shown in Figure 2 with reference to the reduced feature space as given in Table 4.

4.2 Cluster Stability Analysis

We analyze the stability of clusters obtained from TreeCLUSS and several other clustering algorithms, viz., k-means, fuzzy c-means, and hierarchical clustering. A general stability comparison among these clustering algorithms w.r.t. detection rate using the TUIDS datasets is given in Figure 3. The TUIDS datasets were built by us using our own testbed with a variety of attacks (more details are given in 5.1.2). We propose an ensemble based cluster stability analysis technique based

Algorithm 1 : Part 1 TreeCLUSS (D, α, β)

Input: D, the dataset; α, threshold for L_1 cluster formation; β, threshold for L_2 cluster formation;
Output: set of clusters, $C_1, C_2, C_3, \ldots, C_k$

initialization: node_id $\leftarrow$ 0
function BUILDTREE(D, *node_id*)
  for $i \leftarrow 1$ to D **do**
    if (D_i.classified ! = 1 and check_ini_feat(MMIFS(D_i)) == true) and $sim(O_i, O_j) \leq \alpha$ **then**
      CreateNode(D_i.no, p_id, temp, $node_{count}$, node_id, l)
      while (n_f - (l - 1)) $\geq \theta$ **do**
        l++
        for $i \leftarrow 1$ to D **do**
          if D_i.classified ! = 1 **then**
            p_id = check_parent(D_i.no, l)
            if (p_id > -1 and check_ini_feat(MMIFS(D_i)) == true) **then**
              CreateNode(D_i.no, p_id, temp, $node_{count}$, node_id, l)
            end if
          end if
        end for
      end while
      l = 1
    end if
  end for
end function
function CREATENODE(no, p_id, temp, $node_{count}$, id, l)
  node_id = new node()
  node_id.temp = temp
  node_id.$nodel_{count}$ = $node_{count}$
  node_id.p_node = p_id
  node_id.id = id;
  node_id.level = l
  ExpandNode(no, id, node_id.temp, $node_{count}$, l)
  temp = NULL;
  $node_{count}$ = 0;
  node_id++
end function
function EXPANDNODE(no, id, temp, $node_{count}$, l)
  if D_{no}.classified == 1 **then**
    return
  else
    D_{no}.classified = 1;
    D_{no}.node_id = id
    for $i \leftarrow 1$ to D **do**

Algorithm 2 : Part 2 TreeCLUSS (D, α, β)

```
            if (D_i.classified ! = 1) then
                minRank_f = find_minRank(MMIFS(D_i))
                if (n_f - minRank_f) ≥ θ then
                    minRank_f++ until get a specific cluster; otherwise stop.
                    ExpandNode(D_i.no, id, temp, temp_count, l)
                end if
            end if
        end for
    end if
end function
function StableCLUSS(C_k)
    for i ← 1 to k do
        for j ← 1 to 5 do
            VI_c[j] = compute(I_{c_i})
            if (VI_c[j] ≥ σ or VI_c[j] ≤ τ) then
                VI_c[j] = 1
            else
                VI_c[j] =0
            end if
        end for
        if (C_i = Max(VI_c[i])) then
            stable cluster, C_i
            Return Max(VI_c[i])
        else
            go to step 2
        end if
    end for
end function
```

on Dunn index [25], C-index (C) [26], Davies Bouldin index (DB) [27], Silhouette index (S) [28] and Xie-Beni index (XB) [29] (shown in Figure 1). Thus, we choose several well known cluster validity measures for stability analysis. We analyze each cluster based on distance to reduce computational overhead. All our measures are distance-based. We briefly discuss each measure along with the values expected for good clusters in Table 5.

We pass each cluster C_i to a function StableCLUSS to measure stability. It computes all the indices for each of the clusters $C_1, C_2, \ldots, C_k$. If it judges that the result is good for an index, it stores a 1, otherwise assigns 0. It computes 1 or 0 for each of the indices as given below. σ and τ are threshold parameters.

$$V_i = \begin{cases} 1, & I_i \geq \sigma \text{ or } I_i \leq \tau \\ 0, & \text{otherwise.} \end{cases}$$

Object ID	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	CL
O_1	9.23	0.71	2.43	0.60	104	2.80	3.06	0.28	2.29	5.64	1
O_2	22.53	6.51	6.64	4.96	72.79	2.60	11.63	0.80	9.00	36.97	8
O_3	16.37	5.76	1.16	11.88	95	5.50	1.10	0.49	6.87	20.45	5
O_4	10.37	1.95	9.50	0.80	110	1.85	2.49	0.64	3.18	9.80	2
O_5	14.67	4.85	1.92	11.94	96	4.10	1.79	0.12	4.73	10.80	4
O_6	9.20	0.78	2.14	0.20	103	2.65	2.96	0.26	2.28	4.38	1
O_7	12.37	0.84	1.36	11.60	95	3.98	1.57	0.98	1.42	10.95	3
O_8	9.16	1.36	9.67	0.60	110	1.80	2.24	0.60	3.81	9.68	2
O_9	16.17	5.86	1.53	11.87	93	5.89	1.75	0.45	6.73	20.95	5
O_{10}	18.81	6.31	4.40	4	70	2.15	8.09	0.57	7.83	27.70	6
O_{11}	14.64	4.82	1.02	11.80	94	4.02	1.41	0.13	4.62	10.75	4
O_{12}	20.51	6.24	5.25	4.50	70.23	2	9.58	0.60	8.25	32.45	7
O_{13}	12.33	0.71	1.28	11.89	96	3.05	1.09	0.93	1.41	10.27	3
O_{14}	20.60	6.46	5.20	4.50	71	2.42	9.66	0.63	8.94	32.10	7
O_{15}	18.70	6.55	5.36	4.50	73.24	2.70	8.20	0.57	7.84	27.10	6
O_{16}	22.25	6.72	6.54	4.89	69.38	2.47	10.53	0.80	9.85	36.89	8

Table 3. Sample dataset, D and CL in the last column is the class label

Class	Object ID	Relevant feature set	Feature rank value
C_1	O_1, O_4, O_6, O_8	f_5, f_6, f_2, f_3, f_9, f_{10}, f_7, f_8	1, 1, 1, 1, 1, 1, 1, 1
C_{11}	O_1, O_6	f_5, f_6, f_2, f_3, f_9, f_{10}, f_7	1, 1, 1, 1, 1, 1, 1
C_{12}	O_4, O_8	f_5, f_6, f_2, f_3, f_9, f_{10}	1, 1, 1, 1, 1, 1
C_2	O_3, O_5, O_7, O_9, O_{11}, O_{13}	f_1, f_2, f_6, f_9, f_8, f_{10}	1.585, 1.585, 1.585, 1.585, 1.585, 0.918
C_{21}	O_3, O_9	f_1, f_2, f_6, f_9, f_8	1.585, 1.585, 1.585, 1.585, 1.585
C_{22}	O_5, O_{11}	f_1, f_2, f_6, f_9	1.585, 1.585, 1.585, 1.585
C_{23}	O_7, O_{13}	f_1, f_2, f_6, f_8	1.585, 1.585, 1.585, 1.585
C_3	O_2, O_{10}, O_{12}, O_{14}, O_{15}, O_{16}	f_7, f_1, f_{10}, f_8, f_9, f_4, f_3	1.584, 1.584, 1.584, 1.584, 1.584, 0.917, 0.917
C_{31}	O_2, O_{16}	f_7, f_1, f_{10}, f_8, f_9, f_4	1.584, 1.584, 1.584, 1.584, 1.584, 0.917
C_{32}	O_{10}, O_{15}	f_7, f_1, f_{10}, f_8, f_9	1.584, 1.584, 1.584, 1.584, 1.584
C_{33}	O_{12}, O_{14}	f_7, f_1, f_{10}, f_8, f_4	1.584, 1.584, 1.584, 1.584, 0.917

Table 4. Relevant feature set (n_f) and attribute rank values

Finally, we take the maximum number of occurrences of 1 to decide if a cluster is stable or not. If a cluster C_i is not stable, it sends control back to TreeCLUSS to regenerate another set with a different number of clusters. We choose the best set of stable clusters after we execute the module multiple times.

4.3 CLUSSLab: The Cluster Labelling Technique

CLUSSLab is a multi-objective cluster labelling technique for labelling the clusters generated by TreeCLUSS. It decides the label of the instances of a cluster based on a combination of the following measures:

1. cluster size,
2. compactness,
3. dominating feature subset and
4. outlier score of each instance.

Each measure is described next.

1. *Cluster size:* It is the number of instances in a cluster.

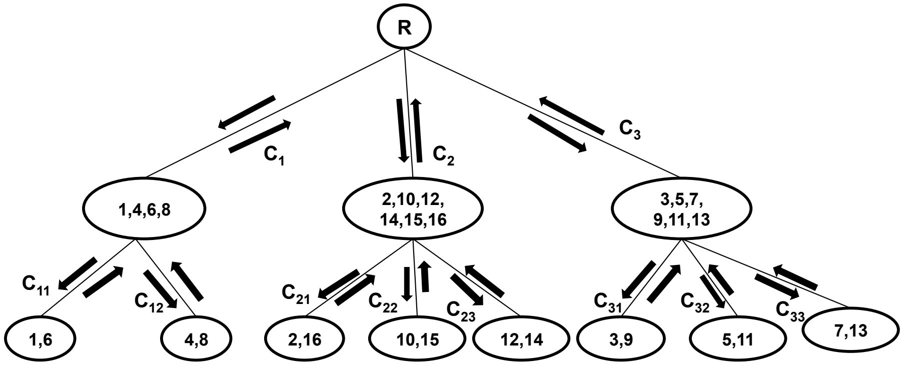

Figure 2. Tree obtained from D, given in Table 3

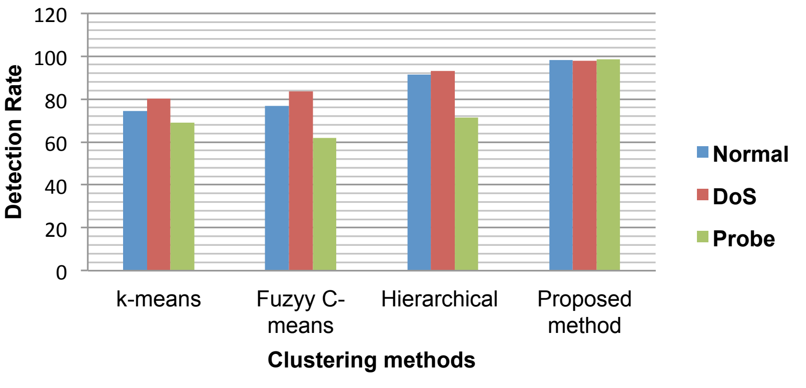

Figure 3. Comparison of stability analysis with various algorithms using TUIDS packet level intrusion dataset

2. *Compactness:* To find the compactness of a cluster C_i, obtained from TreeCLUSS, we use the five very well known indices as given in Table 5 and discussed earlier in Section 2.2.

3. *Dominating feature subset:* The subset of features which mostly influences the formation of the clusters is referred to as the dominating feature set. We identify the dominating features by using an adaptive unsupervised feature clustering technique (UReFT) based on Renyi's entropy [30]. Renyi's entropy performs non-parametric estimation by avoiding the problems of the traditional entropy metric. Renyi's entropy with probability density function (pdf) f_x for a stochastic variable x and Renyi's constant λ is given by

$$H_R(x) = \frac{1}{1-\lambda} \ln \int f_x^{\lambda} dx, \lambda > 0, \lambda \neq 1. \tag{1}$$

Stability measures	Definition	Features
Dunn index (Dunn)	$\frac{d_{\min}}{d_{\max}}$, where $d_{\min}$ denotes the smallest distance between two objects from different clusters and $d_{\max}$ is the largest distance between two elements within the same cluster.	(a) Computed for finding compact and well separated clusters. (b) Larger values of *Dunn* indicates better clustering, i.e., the range is $(0, \infty)$.
C-index (C)	$\frac{S-S_{\min}}{S_{\max}-S_{\min}}$, where S is the sum of distances over all pairs of objects form the same cluster, n is the number of such pairs, $S_{\min}$ and $S_{\max}$ are the sum of n smallest distances and n largest distances, respectively.	(a) Used to find cluster quality when the clusters are similar sizes. (b) Smaller values of C indicate better clusters, i.e., the range is $(0, 1)$.
Davies Bouldin index (DB)	$\frac{1}{n}\sum_{i=1, i\neq j}^{n} \max(\frac{\sigma_j+\sigma_j}{d(c_i,c_j)})$, where n is the number of clusters; σ_i is the average distance of all patterns in cluster i to their cluster center, c_i; σ_j is the average distance of all patterns in cluster j to their cluster center, c_j; and $d(c_i, c_j)$ represents the proximity between the cluster centers c_i and c_j.	(a) Lower value of DB indicates better clusters, i.e., the range is $(0, \infty)$. (b) It has low computational cost and can find better clusters of spherical shape.
Silhouette index (S)	$\frac{b_i-a_i}{\max\{a_i,b_i\}}$, where a_i is the average dissimilarity of i^{th} object to all other objects in the same cluster; b_i is the minimum of average dissimilarity of the i^{th} object to all objects in other clusters.	(a) Computed for a cluster to identify tightly separated groups. (b) Better if the index value is near 1, i.e., the range is $(-1, 1)$.
Xie Beni index (XB)	$\frac{\pi}{N.d_{\min}}$, where $\pi = \frac{\sigma_i}{n_i}$, is called compactness of cluster i. Since n_i is the number of points in cluster i, σ is the average variation in cluster i; $d_{\min} = \min \|\|k_i - k_j\|\|$.	Smaller values of XB are expected for compact and well-separated clusters, i.e., the range is $(0, 1)$.

Table 5. Cluster stability measure: definition, features and criteria for better clustering

Renyi's quadratic entropy is defined by [31] when $\lambda = 2$ as follows, assuming a Gaussian pdf:

$$\begin{aligned} H_R(x) &= -\ln \int f_x^2 dx \\ &= -\ln \left(\frac{1}{N} \sum_{i=1}^{N} G\left(x - x_i, \sigma^2\right) \right) \left(\frac{1}{N} \sum_{i=1}^{N} G\left(x - x_j, \sigma^2\right) \right) \\ &= -\ln \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} G\left(x_i - x_j, 2\sigma^2\right) \end{aligned} \tag{2}$$

where G is the Gaussian kernel, σ is the smoothing parameter (we found better results when $\sigma = 0.9$ to 0.12), x_i and x_j are the i^{th} and j^{th} features of N data objects. We also note that

$$G\left(x_i - x_j, 2\sigma^2\right) = \frac{1}{(2\pi)^{\frac{d}{2}}\sqrt{2\sigma^2}} exp\left(-\frac{(x_i - x_j)^2}{4\sigma^2}\right) \tag{3}$$

where d is the dimension of variable x. Assume that we obtain k feature clusters, i.e., $C = \{C_1, C_2, \ldots C_k\}$. A feature object x is assigned to a cluster C_i iff,

$$\left(H(C_i + x) - H(C_i)\right) < \left(H(C_k + x) - H(C_k)\right), k \neq i \tag{4}$$

where $H(C_k)$ denotes the entropy of cluster C_k. This method is referred to as differential entropy clustering [32]. We compute $H(C_k)$ and $H(C_i, C_j)$ for within-cluster and between-cluster entropy as follows.

$$H(C_k) = -\ln \frac{1}{N_k^2} \sum_{i=1}^{N_k} \sum_{j=1}^{N_k} G\left(x_i - x_j, 2\sigma^2\right) \tag{5}$$

$$H(C_i, C_j) = -\ln \frac{1}{N_i N_j} \sum_{p=1}^{N_i} \sum_{q=1}^{N_j} G\left(x_p - x_q, 2\sigma^2\right) \tag{6}$$

The main goal of our technique is to identify a dominating feature set with the least redundancy and the most relevancy. Initially, we assume that each cluster contains two feature subsets:

(a) the selected or relevant subset and
(b) the non-selected or irrelevant subset.

The selected cluster is the dominating feature set and the nonselected cluster is the irrelevant feature set. The method starts with a single feature object C_s, and assigns another object to it by computing Renyi's entropy (using Equations (4), (5) and (6)) w.r.t. a threshold η_1, otherwise it creates a new cluster, C_{ns} known as the non-selected cluster. It adaptively assigns each candidate feature object to C_s or C_{ns} w.r.t. threshold η_1 and the threshold for intra cluster entropy η_2. The threshold values of η_1 and η_2 are also chosen based on a heuristic approach.

4. *Outlier score:* Here, we use our own outlier identification algorithm, RODD [42] to compute a score for each instance with reference to the normal profiles. A graph is plotted based on sorted outlier ranking against those instances as shown in Figure 4, and from the graph, a cutoff is decided to distinguish the normal from anomalous instances. We see in the graph that for any two-class combination such as (normal, DoS), (normal, probe), (normal,U2R), or (normal, R2L) with various proportions, it is still possible to distinguish the normal from the rest.

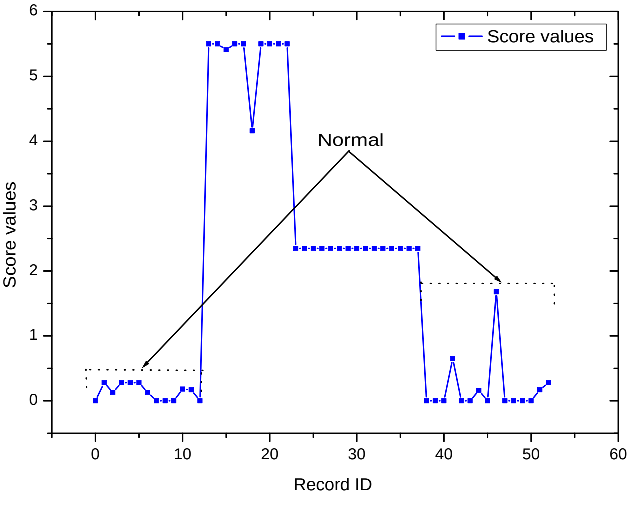

Figure 4. Identification of normal ranges using outlier score ranking over intrusion datatset

On the basis of cluster size, compactness, dominating features identified using UReFT and interval of outlier score rank values, we label each cluster as anomalous or normal w.r.t. the thresholds. We obtained the best result for labelling each cluster as anomalous with matching probability ≤ 0.63 w.r.t. the above measures. The CLUSSLab algorithm is given as Algorithm 3. It is a multi-objective technique to label each cluster as normal or anomalous. UReFT is the unsupervised Renyi's entropy based feature clustering technique to identify the relevant features set for each cluster. It matches the existing class specific feature set while labelling.

4.4 Complexity Analysis

As discussed, the proposed method is works in two phases. The first phase is subspace clustering technique, i.e., the TreeCLUSS. We assume that k clusters are obtained from n data objects. During cluster formation, TreeCLUSS takes $O(n \log k)$ time and for stability analysis, it takes $O(k \log k)$ time. Hence, the total computational complexity of TreeCLUSS is $O(n \log k)$.

The second phase is multi-objective cluster labelling technique, i.e., the CLUSSLab. It is again comprised of four sub-modules viz., cluster size, compactness, dominating feature subset (DFS) and outlier score (OS). To compute, compactness, dominating feature subset and outlier score, it takes $O(n \log n)$, $O(n)$, and $O(kn)$ time, respectively. Hence, the total time complexity of CLUSSLab is $O(n \log n + kn)$

The time complexity for each stage of our unsupervised network anomaly detection method is linear w.r.t. the size of dataset, the number of features, the number of clusters and the labelling of each clusters. Hence, it is effective in detecting known as well as unknown attacks with the least amount of false alarms.

Algorithm 3 : CLUSSLab($C_k, \xi_1, \xi_2, \xi_3, \xi_4$)

Input: C_k represents a cluster obtained from TreeCLUSS, ξ_1 is the number of instances in a cluster, ξ_2 is the cluster compactness score, ξ_3 is the matching probability of features of a cluster with a specific class and ξ_4 is the outlier score value of each instance of a cluster.

Output: Label clusters $C_1, C_2, C_3, \ldots C_k$ as normal or anomalous.

for $i \leftarrow 1$ to k **do**
    $S[i] = |C_i|$
    M[i] = call StableCLUSS(C_i)
end for
function UREFT(C_k)
    for $i \leftarrow 1$ to k **do**
        for $j \leftarrow 1$ to S_i **do**
            if $(H(C_s)) \leq \eta_1$ && $(H(C_s, C_{ns})) \leq \eta_2$ **then**
                $C_s[z] \leftarrow f_z,\ z = 1, 2, \ldots d$
            else
                $C_{ns}[z] \leftarrow f_z,\ z = 1, 2, \ldots d$
            end if
        end for
    end for
end function
for $i \leftarrow 1$ to k **do**
    if $S[i] \leq \xi_1$ && $M[i] < \xi_2$ && $C_i \geq \xi_4$ **then**
        if $P(|C_s[z]|, |MMIFS[z]|) \leq \xi_3$ **then**
            $anomalous \leftarrow C_i$
        else
            $normal \leftarrow C_i$
        end if
    end if
end for

5 EXPERIMENTAL ANALYSIS

In this section, we present experimental analysis and results of the unsupervised network anomaly detection method using several real world datasets from the UCI machine learning repository and datasets prepared at the TUIDS testbed at both packet and flow levels [33]. The network laboratory layout where we capture network traffic for the TUIDS intrusion dataset is shown in Figure 5. The network has 32 subnets including a wireless network, 4 routers, 3 wireless controllers, 8 L3 switches, 15 L2 switches and 300 hosts. A DHCP server is set up inside the main network for wireless network. Each router can be controlled to connect other networks as well as to route packets to specific networks. The datasets used in this paper to evaluate the proposed method and experimental results are discussed below.

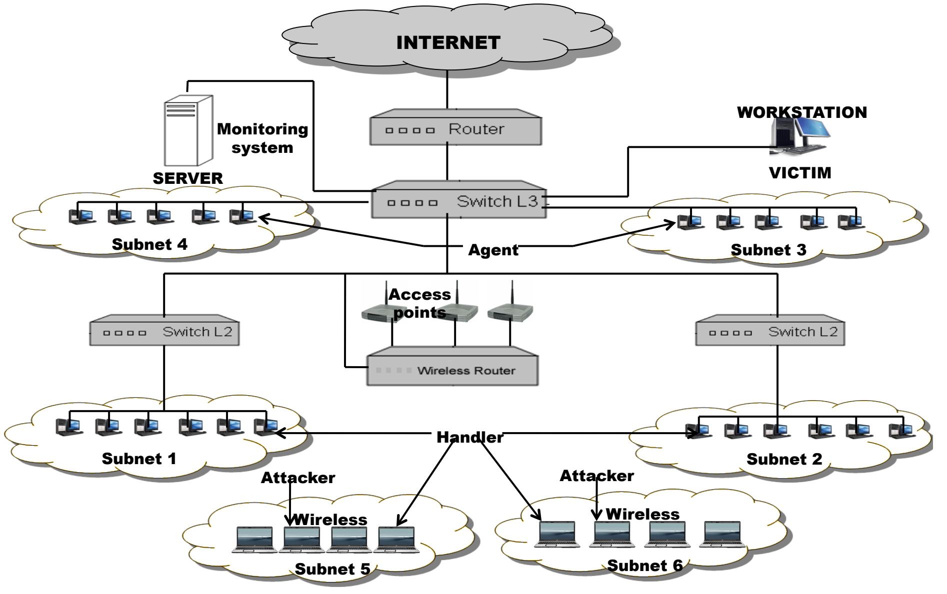

Figure 5. TUIDS testbed: All TUIDS datasets are prepared using this testbed with a number of configurations for the network as well as capturing tools

5.1 Datasets Used

We use two sets of datasets, namely:

1. nonintrusion datasets taken from UCI ML repository for initial evaluation and establishment of the proposed algorithms and
2. intrusion datasets.

5.1.1 Nonintrusion Datasets

We use ten nonintrusion datasets [34]: Zoo, Glass, Abalone, Shuttle, Wine, Lymphography, Heart, Pima, Vehicle and Poker Hand to initially validate clusters generated by TreeCLUSS. Table 6 describes the details of the nonintrusion datasets and their characteristics.

5.1.2 Intrusion Datasets

We use five different real life intrusion datasets. These are

1. TUIDS coordinated scan dataset,
2. TUIDS intrusion dataset,

Non-intrusion Datasets (NID)	Datasets	Dimension	No. of instances	No. of classes
NID1	Zoo	18	101	7
NID2	Glass	10	214	6
NID3	Abalone	8	4 177	29
NID4	Shuttle	9	14 500	3
NID5	Wine	13	178	3
NID6	Lymphography	18	148	4
NID7	Heart	13	270	2
NID8	Pima	8	768	2
NID9	Vehicle	18	846	4
NID10	Poker Hand	10	25 010	10

Table 6. Characteristics of real-life nonintrusion datasets

3. TUIDS DDoS dataset,
4. NSL-KDD dataset and
5. KDDcup99 dataset.

The attacks used to generate traffic and prepare labeled intrusion datasets are shown in Table 7. We capture, preprocess, and extract features in both packet and flow level network traffic. GULP (Lossless Gigabit Remote Packet Capture With Linux)[1] is used to capture the packet level traffic with launched attacks as well as normal traffic while NFDUMP[2] and NfSen[3] are used to capture and visualize flow level network traffic. The lists of extracted features in both packet and flow level intrusion datasets are presented in Table 8 and Table 9, respectively. More details of the TUIDS datasets can be found in [33]. Next, we describe each dataset in brief.

1. *TUIDS real-time Coordinated scan dataset:* We launched attacks numbered 12-17 (as given in Table 7) in a coordinated mode using the *rnmap*[4] tool to generate the traffic including normal traffic. We captured the traffic in both packet and flow levels to prepare the dataset. Characteristics of this dataset are given in Table 11.
2. *TUIDS real-time intrusion dataset:* This dataset is prepared by launching 20 different attacks with normal traffic connections. It contains 15 DoS attacks and 5 probe attacks. Characteristics of this datasets are given in Table 11.
3. *TUIDS real-time DDoS dataset:* It is prepared using the same TUIDS testbed with three different flooding attacks (viz., attacks numbered 18, 21 and 22 in Table 7) launched in amplification mode while capturing the traffic at flow level

[1] `http://staff.washington.edu/corey/gulp/`
[2] `http://nfdump.sourceforge.net/`
[3] `http://nfsen.sourceforge.net/`
[4] `http://rnmap.sourceforge.net/`

Attack name	Attack generation tools	Attack name	Attack generation tools
1. Bonk	targa2.c	2. Jolt	targa2.c
3. Land	targa2.c	4. Saihyousen	targa2.c
5. TearDrop	targa2.c	6. Newtear	targa2.c
7. 1 234	targa2.c	8. Winnuke	targa2.c
9. Oshare	targa2.c	10. Nestea	targa2.c
11. SynDrop	targa2.c	12. WindowScan	Nmap/Rnmap
13. SynScan	Nmap/Rnmap	14. XmassTreeScan	Nmap/Rnmap
15. NULLScan	Nmap/Rnmap	16. UDPScan	Nmap/Rnmap
17. FINScan	Nmap/Rnmap	18. Smurf	smurf4.c
19. OpenTear	opentear.c	20. LinuxICMP	linux-icmp.c
21. Fraggle	fraggle.c	22. Synflood	synflood.c

Table 7. Attacks used with their tools in TUIDS dataset preparation

only. Characteristics of this dataset are given in Table 11. A brief description of DDoS attacks we launched is given below.

- In Smurf attack, the attacker sends packets to a network amplifier (a system supporting broadcast addressing), with the return address spoofed to the victim's IP address. It uses ICMP ECHO packets and as a result, the original packet spoofs tens or even hundreds of times to the victim host.
- The Fraggle attack is similar to a Smurf attack in that the attacker sends packets to a network amplifier but uses UDP ECHO packets instead of ICMP ECHO packets. The UDP ECHO packets are sent to the port that supports character generation (chargen, port 19 in Unix systems), with the return address spoofed to the victim's echo service (echo, port 7 in Unix systems) creating an infinite loop.
- The SYN flooding attack exploits the TCP's three-way handshake mechanism and its limitation in maintaining half-open connections. So, it drops more packets while sending from source to destination.

4. *NSL-KDD intrusion dataset:* NSL-KDD[5] is an enhanced version of the KDDcup99 dataset. This is a well-known dataset for intrusion detection system evaluation. The dataset is described in Table 11.
5. *KDDcup99 intrusion dataset:* This is the most well-known and the most popular intrusion dataset used for evaluation of any intrusion detection system. It contains training data processed into about five million network connection records. A connection record is a sequence of TCP packets with well-defined starting and ending times. Each connection record is unique in the dataset with 41 continuous and nominal features plus one class label. The features available

[5] `http://www.iscx.ca/NSL-KDD/`

Label/feature name	Type*	Description
Basic features		
1. Duration	C	Length (number of seconds) of the connection
2. Protocol-type	D	Type of protocol, e.g., tcp, udp, icmp
3. Src-ip	C	Source host IP address
4. Dest-ip	C	Destination IP address
5. Src-port	C	Source host port number
6. Dest-port	C	Destination host port number
7. Service	D	Network service on the destination e.g., http, telnet
8. num-bytes-src-dst	C	The number of data bytes flowing from source to destination
9. num-bytes-dst-src	C	The number of data bytes flowing from destination to source
10. Fr-no	C	Frame number
11. Fr-len	C	Frame length
12. Cap-len	C	Captured frame length
13. Head-len	C	Header length of the packet
14. Frag-off	D	Fragment offset '1' for the second packet overwrite everything '0' otherwise
15. TTL	C	Time to live '0' discards the packet
16. Seq-no	C	Sequence number of the packet
17. CWR	D	Congestion window record
18. ECN	D	Explicit congestion notification
19. URG	D	Urgent TCP flag
20. ACK	D	Acknowledgement flag value
21. PSH	D	Push TCP flag
22. RST	D	Reset TCP flag
23. SYN	D	Syn TCP flag
24. FIN	D	Fin TCP flag
25. Land	D	1 if connection is from/to the same host/port; 0 otherwise
Content-based features		
26. Mss-src-dest-requested	C	Maximum segment size from source to destination requested
27. Mss-dest-src-requested	C	Maximum segment size from destination to source requested
28. Ttt-len-src-dst	C	Time to live length from source to destination
29. Ttt-len-dst-src	C	Time to live length from destination to source
30. Conn-status	C	Status of the connection (e.g., '1' for complete, '0' for reset)
Time-based features		
31. count-fr-dest	C	Number of frames received by unique destination in the last T seconds from the same source
32. count-fr-src	C	Number of frames received by unique source in the last T seconds to the same destination
33. count-serv-src	C	Number of frames from the source to the same destination port in the last T seconds
34. count-serv-dest	C	Number of frames from destination to the same source port in the last T seconds
35. num-pushed-src-dst	C	The number of pushed packets flowing from source to destination
36. num-pushed-dst-src	C	The number of pushed packets flowing from destination to source
37. num-SYN-FIN-src-dst	C	The number of SYN/FIN packets flowing from source to destination
38. num-SYN-FIN-dst-src	C	The number of SYN/FIN packets flowing from destination to source
39. num-FIN-src-dst	C	The number of FIN packets flowing from source to destination
40. num-FIN-dst-src	C	The number of FIN packets flowing from destination to source
Connection-based features		
41. count-dest-conn	C	Number of frames to unique destination in the last N packets from the same source
42. count-src-conn	C	Number of frames from unique source in the last N packets to the same destination
43. count-serv-srcconn	C	Number of frames from the source to the same destination port in the last N packets
44. count-serv-destconn	C	Number of frames from the destination to the same source port in the last N packets
45. num-packets-src-dst	C	The number of packets flowing from source to destination
46. num-packets-dst-src	C	The number of packets flowing from destination to source
47. num-acks-src-dst	C	The number of acknowledgement packets flowing from source to destination
48. num-acks-dst-src	C	The number of acknowledgement packets flowing from destination to source
49. num-retransmit-src-dst	C	The number of retransmitted packets flowing from source to destination
50. num-retransmit-dst-src	C	The number of retransmitted packets flowing from destination to source

Table 8. List of packet level features in the TUIDS intrusion dataset. C and D in the second column represent continuous and discrete features, respectively.

Label/feature name	Type*	Description
Basic features		
1. Duration	C	Length (number of seconds) of the flow
2. Protocol-type	D	Type of protocol, e.g., TCP, UDP, ICMP
3. Src-ip	C	Source host IP address
4. Dest-ip	C	Destination IP address
5. Src-port	C	Source host port number
6. Dest-port	C	Destination host port number
7. ToS	D	Type of service
8. URG	D	TCP urgent flag
9. ACK	D	TCP acknowledgement flag
10. PSH	D	TCP push flag
11. RST	D	TCP reset flag
12. SYN	D	TCP SYN flag
13. FIN	D	TCP FIN flag
14. Src-bytes	C	Number of data byte transfer from source to destination
15. Dest-bytes	C	Number of data byte transfer from destination to source
16. Land	D	1 if connection is from/to the same host/port; 0 otherwise
Time-based features		
17. count-dest	C	Number of flows to unique destination IP in the last T seconds from the same source
18. count-src	C	Number of flows from unique source IP in the last T seconds to the same destination
19. count-serv-src	C	Number of flows from the source to the same destination port in the last T seconds
20. count-serv-dest	C	Number of flows from the destination to the same source port in the last T seconds
Connection-based features		
21. count-dest-conn	C	Number of flows to unique destination IP in the last N flows from the same source
22. count-src-conn	C	Number of flows from unique source IP in the last N flows to the same destination
23. count-serv-srcconn	C	Number of flows from the source IP to the same destination port in the last N flows
24. count-serv-destconn	C	Number of flows to the destination IP to the same source port in the last N flows

Table 9. List of flow level features in the TUIDS intrusion dataset. C and D in the second column represent continuous and discrete features, respectively.

in the KDDcup99 dataset are given in Table 10. A detailed description of the dataset is also given in Table 11.

5.2 Results and Discussion

In this section, we report the performance of the proposed method using real-life and benchmark datasets. The method does not use any class information when it processes a dataset for anomaly detection. We measure the accuracy of the algorithms using the following metric.

- Detection rate = True Positive/(True Positive + False Negative)
- False positive rate = False Positive/(False Positive + True Negative)

Label/feature name	Type*	Description
Basic features		
1. Duration	C	Length (number of seconds) of the connection
2. Protocol-type	D	Type of protocol, e.g., tcp, udp, etc.
3. Service	D	Network service on the destination, e.g., http, telnet etc.
4. Flag	D	Normal or error status of the connection
5. Src-bytes	C	Number of data bytes from source to destination
6. Dst-bytes	C	Number of data bytes from destination to source
7. Land	D	1 if connection is from/to the same host/port; 0 otherwise
8. Wrong-fragment	C	Number of "wrong" fragments
9. Urgen	C	Number of urgent packets
Content-based features		
10. Hot	C	Number of "hot" indicators (hot: number of directory accesses, create and execute program)
11. Num-failed-logins	C	Number of failed login attempts
12. Logged-in	D	1 if successfully logged-in; 0 otherwise
13. Num-compromised	C	Number of "compromised" conditions (compromised condition: number of file/path not found errors and jumping commands)
14. Root-shell	D	1 if root-shell is obtained; 0 otherwise
15. Su-attempted	D	1 if "su root" command attempted; 0 otherwise
16. Num-root	C	Number of "root" accesses
17. Num-file-creations	C	Number of file creation operations
18. Num-shells	C	Number of shell prompts
19. Num-access-files	C	Number of operations on access control files
20. Num-outbound-cmds	C	Number of outbound commands in an ftp session
21. Is-host-login	D	1 if login belongs to the "hot" list; 0 otherwise
22. Is-guest-login	D	1 if the login is a "guest" login; 0 otherwise
Time-based features		
23. Count	C	Number of connection to the same host as the current connection in the past 2-second
24. Srv-count	C	Number of connections to the same service as the current connection in the past 2-second (same-host connections)
25. Serror-rate	C	% of connections that have "SYN" errors (same-host connections)
26. Srv-serror-rate	C	% of connections that have "SYN" errors (same-service connections)
27. Rerror-rate	C	% of connections that have "REJ" errors (same-host connections)
28. Srv-rerror-rate	C	% of connections that have "REJ" errors (same-service connections)
29. Same-srv-rate	C	% of connections to the same service (same-host connections)
30. Diff-srv-rate	C	% of connections to different services (same-host connections)
31. Srv-diff-host-rate	C	% of connections to different hosts (same-service connections)
Connection-based features		
32. Dst-host-count	C	Count for destination host
33. Dst-host-srv-count	C	Srv_count for destination host
34. Dst-host-same-srv-rate	C	Same_srv_rate for destination host
35. Dst-host-diff-srv-rate	C	Diff_srv_rate for destination host
36. Dst-host-same-src-port-rate	C	Same_src_port_rate for destination host
37. Dst-host-srv-diff-host-rate	C	Diff_host_rate for destination host
38. Dst-host-serror-rate	C	Serror_rate for destination host
39. Dst-host-srv-serror-rate	C	Srv_serror_rate for destination host
40. Dst-host-rerror-rate	C	Rerror_rate for destination host
41. Dst-host-srv-rerror-rate	C	Srv_rerror_rate for destination host

Table 10. List of features in the KDDcup99 intrusion dataset. C and D in the second column represent continuous and discrete features, respectively.

Intrusion Datasets (ID)	Connection type	Dimensions	No. of instances	No. of classes
ID1	*TUIDS coordinated scan packet level*			
	Normal		106 380	1
	Probe	50	14 423	6
	Total		120 803	7
ID2	*TUIDS coordinated scan flow level*			
	Normal		36 033	1
	Probe	25	15 654	6
	Total		51 687	7
ID3	*TUIDS packet level*			
	Normal		47 895	1
	DoS	50	30 613	15
	Probe		7 757	5
	Total		86 265	21
ID4	*TUIDS flow level*			
	Normal		16 770	1
	DoS	25	14 475	15
	Probe		9 480	5
	Total		40 725	21
ID5	*TUIDS DDoS flow level*			
	Normal	25	43 252	1
	Flooding attacks		22 707	3
	Total		65 959	4
ID6	*NSL-KDD packet level*			
	Normal		9 711	1
	DoS		7 460	11
	Probe	41	2 421	6
	R2L		2 753	12
	U2R		199	8
	Total		22 544	38
ID7	*KDDcup99 corrected packet level*			
	Normal		60 593	1
	DoS		229 853	12
	Probe	41	4 166	6
	R2L		16 189	12
	U2R		228	6
	Total		311 029	37

Table 11. Distribution of Normal and Attack connection instances in real time TUIDS Coordinated scan (packet and flow), TUIDS (packet and flow), TUIDS DDoS flow level, NSL-KDD packet level and KDDcup99 packet level intrusion datasets

5.2.1 Nonintrusion Datasets

The method was initially tested using nonintrusion datasets. We label each cluster obtained by TreeCLUSS using our CLUSSLab cluster labelling technique. We compare performance in terms of detection rate (DR) and false positive rate (FPR). Detailed results are given in Table 12.

5.2.2 Intrusion Datasets

In these experiments, we test our method for network anomaly detection using TUIDS, NSL-KDD and KDDcup99 network intrusion datasets. It converts all categorical attributes into numeric form and then computes $log_b(x_{ij})$ to normalize larger attribute values, where x_{ij} is a large attribute value and b depends on the attribute values. Nominal features such as protocol (e.g., *tcp, udp, icmp*), service type (e.g.,

Dataset	No. of clusters	Correctly detected	Mis-detected	Detection rate (%)	False positive rate (%)
NID1	8	95	6	94.06	0.0594
NID2	9	206	8	96.26	0.0373
NID3	22	4 002	175	95.81	0.0418
NID4	3	14 296	204	98.59	0.0141
NID5	3	174	4	97.75	0.0121
NID6	5	135	13	91.22	0.0471
NID7	2	266	4	98.51	0.0522
NID8	2	761	7	99.08	0.0125
NID9	5	809	36	95.62	0.0613
NID10	12	24 867	143	99.42	0.0018

Table 12. Experimental results with nonintrusion datasets

http, ftp, telnet) and TCP status flags (e.g., *sf, rej*) are converted into numeric features. We replace other categorical values by numeric values also. For example, in the protocol attribute, the value TCP is changed to 1, UDP is changed to 2 and ICMP is changed to 3.

We initially apply TreeCLUSS on a subset of relevant features extracted using the MMIFS algorithm [24] for all intrusion datasets to generate a stable number of clusters and label each cluster using CLUSSLab as normal or anomalous. Experiments used the following datasets:

1. TUIDS real-time Coordinated scan dataset,
2. TUIDS real-time intrusion dataset,
3. TUIDS real-time DDoS dataset,
4. NSL-KDD intrusion dataset and
5. KDDcup99 intrusion dataset.

Then, we apply the MMIFS algorithm to find the class specific relevant subspaces for all datasets. These class specific feature subsets are used during cluster formation. A list of relevant features for all datasets with their ranks in descending order are given in Table 13. Finally, experimental results of all datasets are given in Table 14.

5.2.3 Discussion

We achieve better results than competing algorithms for network anomaly detection in terms of detection rate and false positive rate. A comparison of our method with several competing algorithms, viz., C4.5 [39], ID3 [40], CN2 [41], CBUID [13], TANN [37], HC-SVM [38] using the TUIDS datasets and the KDDcup99 dataset is given in Figures 6 and 7, respectively. It can be easily seen from the figures that our method outperforms other competing algorithms [36, 13, 37, 38] in the terms

Datasets	#Features	Selected features
ID1		*packet level*
Normal	10	8, 33, 7, 9, 14, 28, 45, 1, 48, 2
Probe	15	45, 8, 34, 33, 49, 7, 14, 50, 44, 41, 39, 20, 2, 22, 30
ID2		*flow level*
Normal	11	14, 7, 18, 15, 19, 2, 22, 21, 25, 1, 4
Probe	14	7, 14, 11, 9, 25, 21, 24, 18, 15, 2, 6, 1, 12, 13
ID3		*packet level*
Normal	9	8, 33, 7, 9, 14, 28, 45, 1, 48, 2
DoS	10	8, 33, 7, 40, 38, 9, 2, 41, 49, 2
Probe	13	45, 8, 34, 33, 49, 7, 50, 44, 41, 39, 20, 2, 30
ID4		*flow level*
Normal	11	14, 7, 18, 15, 19, 16, 2, 22, 21, 25, 1
DoS	10	14, 18, 7, 24, 25, 2, 12, 16, 19, 22
Probe	13	7, 14, 11, 9, 16, 25, 21, 24, 18, 15, 2, 6, 1
ID5		*flow level*
Normal	9	8, 33, 7, 9, 14, 28, 45, 1, 48
Flooding attacks	12	8, 9, 31, 14, 33, 43, 49, 47, 7, 42, 1, 11
ID6		*packet level*
Normal	7	5, 3, 23, 6, 35, 1, 29
DoS	10	5, 23, 6, 24, 2, 24, 36, 41, 3, 25
Probe	15	40, 5, 23, 33, 4, 28, 3, 41, 35, 29, 27, 32, 6, 12, 24
U2R	10	5, 1, 3, 33, 24, 23, 14, 6, 32, 21
R2L	14	3, 6, 5, 13, 22, 23, 10, 35, 37, 24, 4, 1, 39, 38
ID7		*packet level*
Normal	6	5, 23, 3, 6, 35, 1
DoS	8	5, 23, 6, 2, 24, 41, 36, 3
Probe	13	40, 5, 33, 23, 28, 3, 41, 35, 27, 32, 12, 24, 28
U2R	10	5, 1, 3, 24, 23, 2, 33, 6, 32, 4, 14, 21
R2L	15	3, 13, 22, 23, 10, 5, 35, 24, 6, 33, 37, 32, 1, 37, 39, 22, 38, 10, 3

Table 13. Feature ranks for all classes in intrusion datasets. See Table 11 for ID numbers.

of detection rate and false positive rate, especially in case of probe, U2R, and R2L attacks.

TreeCLUSS depends on two main parameters, α and β, but users need to provide α value only. β can be derived from the α. Each is chosen using a heuristic approach for each dataset. Hence, our method is less dependent on input parameters compared to competing algorithms [36, 13, 37, 38, 16].

5.3 Statistical Significance Test

In addition to the evaluation based on real-life intrusion data, we also test statistical significance of our results using two well known statistical measures: chi-square test

Type of traffic	No. of clusters	Correctly detected	Mis-detected	Detection rate (%)	False positive rate (%)
ID1	*packet*	*level*			
Normal	7	105 121	1 259	98.81	0.0164
Probe	7	14 292	131	99.09	0.0017
Overall	14	119 413	1 390	98.95	0.0091
ID2	*flow*	*level*			
Normal	5	35 668	365	98.99	0.0153
Probe	7	15 519	135	99.13	0.0015
Overall	12	51 187	500	99.06	0.0084
ID3	*packet*	*level*			
Normal	5	47 109	786	98.35	0.0164
DoS	16	29 997	616	97.99	0.0166
Probe	5	7 637	120	98.45	0.0014
Overall	26	84 743	1 522	98.26	0.0114
ID4	*flow*	*level*			
Normal	3	16 486	284	98.30	0.0169
DoS	16	14 381	101	99.35	0.0167
Probe	4	9 225	255	97.31	0.0149
Overall	23	40 092	640	98.32	0.0161
ID5	*flow*	*level*			
Normal	2	43 104	148	99.65	0.0034
Flooding attacks	4	22 272	435	98.08	0.0195
Overall	6	65 376	583	99.11	0.0089
ID6	*packet*	*level*			
Normal	3	9 573	138	98.57	0.0147
DoS	12	7 391	69	99.08	0.0052
Probe	6	2 356	65	97.32	0.0182
R2L	11	2 367	386	85.97	0.1493
U2R	7	131	68	65.83	0.2050
Overall	39	21 818	726	89.35	0.0784
ID7	*packet*	*level*			
Normal	5	59 901	692	98.85	0.0113
DoS	14	229 796	57	99.97	0.0016
Probe	5	4 018	148	96.45	0.0160
R2L	13	14 007	2 182	86.52	0.1335
U2R	5	151	77	66.23	0.1973
Overall	42	307 873	3 156	98.98	0.0102

Table 14. Results with intrusion datasets using the proposed method

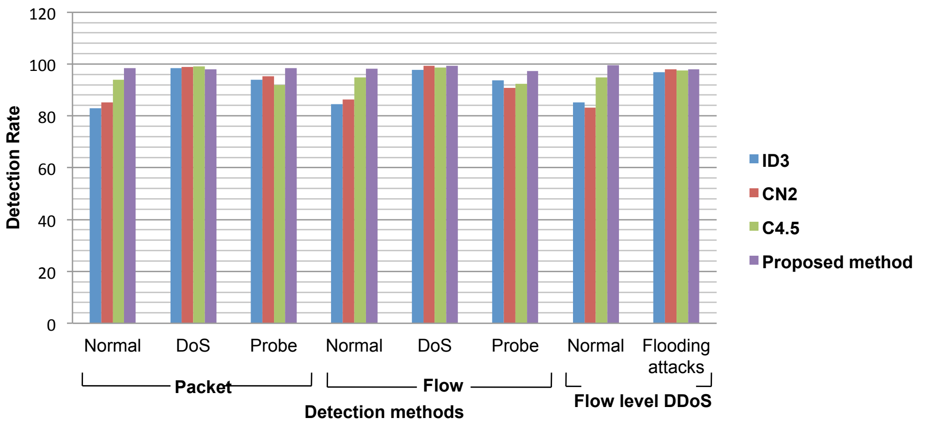

Figure 6. Comparison of our method with competing algorithms using TUIDS intrusion dataset

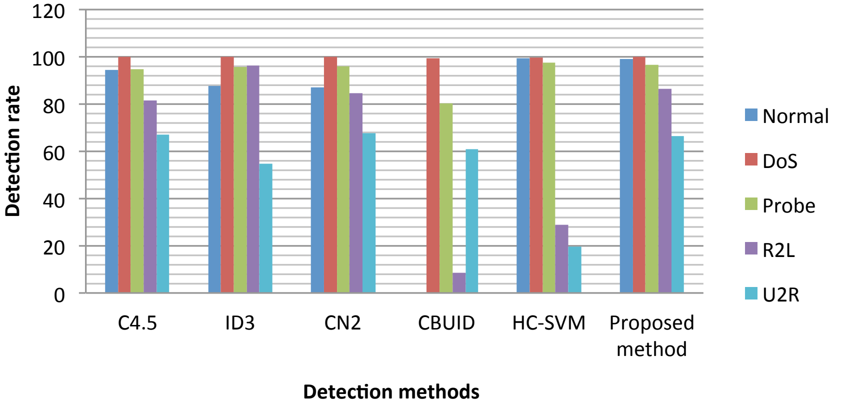

Figure 7. Comparison of proposed method with competing algorithms using KDDcup99 intrusion datasets

and t-test. The chi-square test is used to compute how significantly the observed values are different from the expected values of the distribution for a given sample [43]. We reject the null hypothesis if the chi-square value is greater than the tabulated value w.r.t. the degree of freedom and level of significance. We tested over seven network intrusion datasets mentioned above and obtained significance level $\alpha = 0.05$ in all datasets as shown Figure 8.

The t-test is used to find the difference between two means in relation to the variation in the data. If the computed t-value exceeds the tabulated value, we say that it is highly significant, so that we can reject the null hypothesis. We tested over seven intrusion datasets and obtained t-values as shown in Figure 9. Thus, for both statistical significance tests, we achieved higher significance level for differences between normal and anomalous samples.

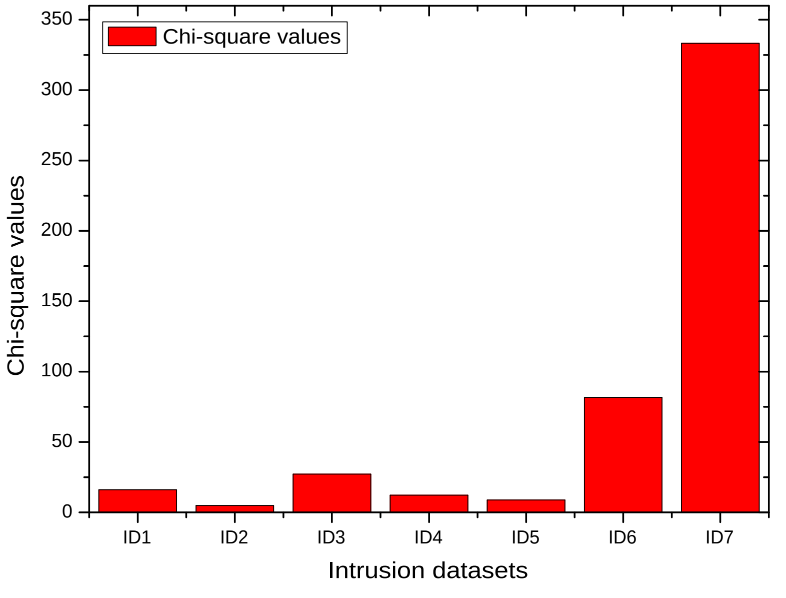

Figure 8. Chi-square test statistics for seven different intrusion datasets with significance level $\alpha = 0.05$ (min = 4.86, max = 333.28)

6 CONCLUSION

In this work, we present a tree based subspace clustering technique for unsupervised network anomaly detection in high dimensional datasets. It generates the approximate number of clusters without having any prior knowledge of the domain. We analyze cluster stability for each cluster by using an ensemble of multiple cluster indices. We also introduce a multi-objective cluster labelling technique to label each stable cluster as normal or anomalous. The major attractions of our proposed method are the following:

1. TreeCLUSS does not require the number of clusters a priori.
2. It is free from the restriction of using a specific proximity measure.
3. CLUSSLab is a multi-objective cluster labelling technique including an effective unsupervised feature clustering technique for identifying dominant feature subset for each cluster.

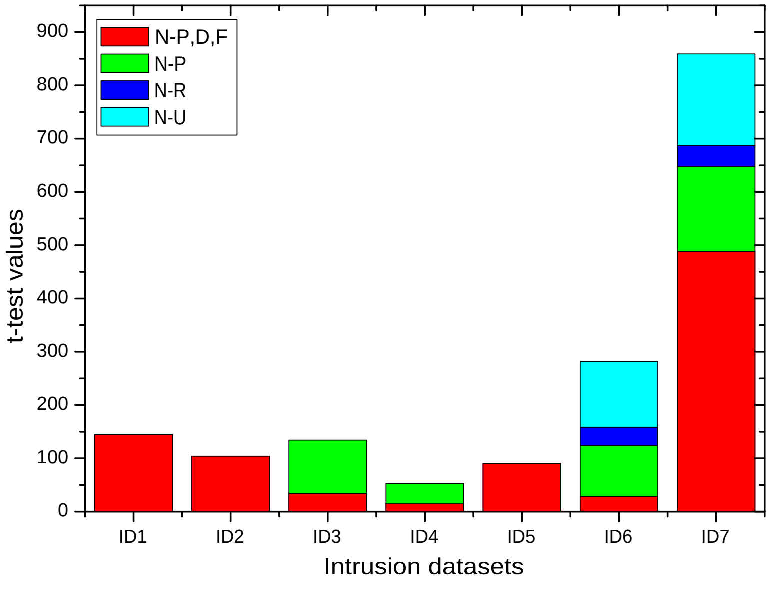

Figure 9. t-test statistics for seven different intrusion datasets with significance level $\alpha = 0.05$; N-P, D, F, R, U represents the normal, probe, DoS, flooding attacks, R2L and U2R respectively

4. TreeCLUSS exhibits a high detection rate and a low false positive rate, especially in case of probe, U2R, and R2L attacks.

Thus, we are able to establish the proposed method to be superior compared to competing network anomaly detection techniques. We also demonstrate that the results produced by our method are statistically significant. A faster, fuzzy incremental semi-supervised version of the proposed technique is underway for mixed type network intrusion data.

Acknowledgment

This work is supported by Department of Information Technology, MCIT and Council of Scientific & Industrial Research (CSIR), Government of India. The research is also funded by the National Science Foundation (NSF), USA under grants DUE-1154342 and CNS-0851783. The authors are thankful to the funding agencies.

REFERENCES

[1] Patcha, A.—Park, J. M.: An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends. Computer Networks, Vol. 51, 2007, No. 12, pp. 3448–3470.

[2] Su, M. Y.: Using Clustering to Improve the KNN-Based Classifiers for Online Anomaly Network Traffic Identification. Journal of Network and Computer Applications, Vol. 34, 2011, No. 2, pp. 722–730.

[3] Toosi, A. N.—Kahani, M.: A new Approach to Intrusion Detection Based on an Evolutionary Soft Computing Model Using Neuro-Fuzzy Classifiers. Computer Communication, Vol. 30, 2007, No. 10, pp. 2201–2212.

[4] Chandola, V.—Banerjee, A.—Kumar, V.: Anomaly Detection: A Survey. ACM Computing Survey, Vol. 41, 2009, No. 3, 2009, pp. 15:1–15:58.

[5] Bhuyan, M. H.—Bhattacharyya, D. K.—Kalita, J. K.: An Effective Unsupervised Network Anomaly Detection Method. In: Proceedings of the 1st International Conference on Advances in Computing, Communications and Informatics, Chennai, India 2012, pp. 533–539.

[6] SNORT: Open Source Network Intrusion Prevention and Detection System. Availaible on: `http://www.snort.org/`, 2010.

[7] BRO: Unix-Based Network Intrusion Detection System. Availaible on: `http://broids.org/`, 2011.

[8] Sperotto, A.—Schaffrath, G.—Sadre, R.—Morariu, C.—Pras, A.—Stiller, B.: An Overview of IP FlowBased Intrusion Detection. IEEE Communications Surveys & Tutorials, Vol. 12, 2010, No. 3, pp. 343–356.

[9] Tavallaee, M.—Stakhanova, N.—Ghorbani, A.: Toward Credible Evaluation of Anomaly-Based IntrusionDetection Methods. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Vol. 40, 2010, No. 5, pp. 516–524.

[10] Portnoy, L.—Eskin, E.—Stolfo, S.: Intrusion Detection with Unlabeled Data Using Clustering. In: Proceedings of ACM, CSS Workshop on Data Mining Applied to Security, Philadelphia 2001, pp. 5–8.

[11] Leung, K.—Leckie, C.: Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters. In: Proceedings of the 28th Australasian Conference on Computer Science, Volume 38, Darlinghurst, Australia 2005, pp. 333–342.

[12] Bhuyan, M. H.—Bhattacharyya, D. K.—Kalita, J. K.: NADO: Network Anomaly Detection Using Outlier Approach. In: Proceedings of the International Conference on Communication, Computing & Security, Rourkela (India) 2011, pp. 531–536.

[13] Jiang, S.—Song, X.—Wang, H.—Han, J. J.—Li, Q. H.: A Clustering-Based Method for Unsupervised Intrusion Detections. Pattern Recognition Letters, Vol. 27, 2006, No. 7, pp. 802–810.

[14] Song, J.—Takakura, H.—Okabe, Y.—Nakao, K.: Toward a More Practical Unsupervised Anomaly Detection System. Information Sciences, 2011, DOI: 10.1016/j.ins.2011.08.011.

[15] Noto, K.—Brodley, C.—Slonim, D.: FRaC: A Feature Modeling Approach for Semisupervised and Unsupervised Anomaly Detection. Data Mining and Knowledge Discovery, Vol. 25, 2012, No. 1, pp. 109–133.

[16] Casas, P.—Mazel, J.—Owezarski, P.: Unsupervised Network Intrusion Detection Systems: Detecting the Unknown Without Knowledge. Computer Communications, Vol. 35, 2012, No. 7, pp. 772–783.

[17] Lange, T.—Roth, V.—Braun, M. L.—Buhmann, J. M.: Stability Based Validation of Clustering Solutions. Neural Computing, Vol. 16, 2004, No. 6, pp. 1299–1323.

[18] Mufti, B. G.—Bertrand, P.—Moubarki, E. L.: Determining the Number of Groups from Measures of Cluster Stability. In: Proceedings of the International Symposium on Applied Stochastic Models and Data Analysis, ENST Bretagne, May 17–20, Brest (France) 2005, pp. 404–413.

[19] Ben-David, S.—von Luxburg, U.—Pál, D.: A Sober Look at Clustering Stability. Proceedings of the 19th Annual Conference on Learning Theory, June 22–25, 2006, LNCS Vol. 4005, Springer Verlag 2006, pp. 5–19.

[20] Das, A. K.—Sil, J.: Cluster Validation Method for Stable Cluster Formation. Canadian Journal on Artificial Intelligence, Machine Learning and Pattern Recognition, Vol. 1, 2010, No. 3, pp. 26–41.

[21] Halkidi, M.—Batistakis, Y.—Vazirgiannis, M.: On Clustering Validation Techniques. Journal of Intelligent Information Systems, Vol. 17, 2001, No. 23, pp. 107–145.

[22] Brock, G.—Pihur, V.—Datta, S.—Datta, S.: clValid: An R Package for Cluster Validation. Journal of Statistical Software, Vol. 25, 2008, No. 4, pp. 1–22.

[23] Jun, S.: An Ensemble Method for Validation of Cluster Analysis. International Journal of Computer Science Issues, Vol. 8, 2011, No. 6, pp. 26–30.

[24] Amiri, F.—Yousefi, M. M. R.—Lucas, C.—Shakery, A.—Yazdani, N.: Mutual Information-Based Feature Selection for Intrusion Detection Systems. Journal of Network and Computer Applications, Vol. 34, 2011, No. 4, pp. 1184–1199.

[25] Dunn, J.: Well Separated Clusters and Optimal Fuzzy Partitions. Journal of Cybernetics, Vol. 4, 1974, pp. 95–104.

[26] Hubert, L.—Schultz, J.: Quadratic Assignment as a General Data Analysis Strategy. British Journal of Mathematical and Statistical Psychology, Vol. 29, No. 2, 1976, pp. 190–241.

[27] Davies, D. L.—Bouldin, D. W.: A Cluster Separation Measure. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 1, 1979, No. 2, pp. 224–227.

[28] Rousseeuw, P. J.: Silhouettes: A Graphical aid to the Interpretation and Validation of Cluster Analysis. Journal of Computational and Applied Mathematics, Vol. 20, 1987, No. 1, pp. 53–65.

[29] Xie, X. L.—Beni, G.: A Validity Measure for Fuzzy Clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, 1991, No. 4, pp. 841–847.

[30] Renyi, A.: On Measures of Entropy and Information. Berkeley Symposium Mathematics, Statistics and Probability 1960, pp. 547–561.

[31] Principe, J. C.—Xu, D.—Fisher, J.: Information Theoretic Learning. Chapter: Unsupervised Adaptive Filtering. John Wiley and Sons 2000, pp. 265–319.

[32] Jenssen, R.—Ii, K. E. H.—Erdogmus, D.—Principe, J. C.—Eltoft, T.: Clustering Using Renyi's Entropy. In: Proceedings of the Joint International Conference on Neural Networks, Portland 2003, pp. 523–528.

[33] Gogoi, P.—Bhuyan, M. H.—Bhattacharyya, D. K.—Kalita, J. K.: Packet and Flowbased Network Intrusion Dataset. In: Proceedings of the 5th International Conference on Contemporary Computing, LNCSCCIS Vol. 306, Springer 2012, pp. 322–334.

[34] Frank, A.—Asuncion, A.: UCI Machine Learning Repository. `http://archive.ics.uci.edu/ml`, University of California, School of Information and Computer Sciences, Irvine, CA 2010.

[35] KDDCUP99: Winning Strategy in KDD99. `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`, 1999.

[36] Beghdad, R.: Critical Study of Supervised Learning Techniques in Predicting Attacks. Information Security Journal: A Global Perspective, Vol. 19, 2010, No. 1, pp. 22–35.

[37] Tsai, C. F.—Lin, C. Y.: A Triangle Area Based Nearest Neighbors Approach to Intrusion Detection. Pattern Recognition, Vol. 43, 2010, No. 1, pp. 222–229.

[38] Horng, S. J.—Su, M. Y.—Chen, Y. H.—Kao, T. W.—Chen, R. J.—Lai, J. L.—Perkasa, C. D.: A Novel Intrusion Detection System Based on Hierarchical Clustering and Support Vector Machines. Expert Systems with Applications, Vol. 38, 2011, No. 1, pp. 306–313.

[39] Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Francisco 1993.

[40] Quinlan, J. R.: Induction of Decision Trees. Mach. Learn., Vol. 1, 1986, No. 1, pp. 81–106.

[41] Clark, P.—Niblett, T.: The CN2 Induction Algorithm. Mach. Learn., Vol. 3, 1989, No. 4, pp. 261–283.

[42] Bhuyan, M. H.—Bhattacharyya, D. K.—Kalita, J. K.: RODD: An Effective Outlier Detection Technique for Large Datasets. In: Proceedings of the 1st International Conference on Computer Science and Information Technology (LNCSCCIS 2010), Bangalore, India 2010, pp. 76–84.

[43] Daniel, W. W.: Biostatistics: A Foundation for Analysis in the Health Sciences. John Wiley & Sons, New York 1987.

Monowar Hussain Bhuyan received his M. Tech. in information technology from the Department of Computer Science and Engineering, Tezpur University, Assam, India in 2009. Currently, he is working towards his Ph. D. in computer science and engineering at the same university. He is a life member of IETE, India. His research areas include machine learning, computer and network security. He has published fifteen papers in international journals and referred conference proceedings.

Dhruba Kr. Bhattacharyya received his Ph. D. in computer science from Tezpur University in 1999. He is a Professor in the Computer Science and Engineering Department at the same university. His research areas include data mining, network security and content based image retrieval. He has published 150+ research papers in leading international journals and conference proceedings. In addition, he has written/edited 8 books. He is a Programme Committee/Advisory Body member of several international conferences/workshops.

Jugal K. Kalita is a Professor of computer science at the University of Colorado at Colorado Springs. He received his Ph. D. from the University of Pennsylvania. His research interests include natural language processing, machine learning, artificial intelligence and bioinformatics. He has published 120 papers in international journals and referred conference proceedings and has written two books.