MINETRAC: Mining Flows for Unsupervised Analysis & Semi-Supervised Classification

Pedro Casas[1,2], Johan Mazel[1,2], and Philippe Owezarski[1,2]
[1]CNRS; LAAS; 7 avenue du colonel Roche, F-31077 Toulouse Cedex 4, France
[2]Universite de Toulouse; UPS, INSA, INP, ISAE; UT1, UTM, LAAS; F-31077 Toulouse Cedex 4, France
Email: {pcasashe, jmazel, owe}@laas.fr

***Abstract*—Driven by the well-known limitations of port-based and payload-based analysis techniques, the use of Machine Learning for Internet traffic analysis and classification has become a fertile research area during the past half-decade. In this paper we introduce MINETRAC, a combination of unsupervised and semi-supervised machine learning techniques capable of identifying and classifying different classes of IP flows sharing similar characteristics. The unsupervised analysis is accomplished by means of robust clustering techniques, using Sub-Space Clustering, Evidence Accumulation, and Hierarchical Clustering algorithms to explore inter-flows structure. MINETRAC permits to identify natural groupings of traffic flows, combining the evidence of data structure provided by different partitions of the same set of traffic flows. Automatic classification is performed by means of semi-supervised learning, using only a small fraction of ground-truth flows to map the identified clusters into their associated most-probable originating network service or application. We evaluate the performance of MINETRAC using real traffic traces, additionally comparing its performance against previously proposed clustering-based flow analysis methods and supervised/semi-supervised classification approaches.**

***Index Terms*—Unsupervised Traffic Analysis, Semi-Supervised Traffic Classification, Sub-Space Clustering, Evidence Accumulation, Hierarchical Clustering.**

I. Introduction

Knowing and understanding the traffic that flows today's Internet is a critical need for Internet Service Providers. Network operators need to know what is flowing over their networks to perform a wide range of network monitoring tasks such as anomaly detection, traffic control, network security management, etc. This practical need has motivated an extensive development of the automatic network traffic classification field, being nowadays a very active research domain.

The objective of automatic traffic classification is to associate a flow of packets to the particular network service or application that generated them. Commonly deployed traffic classification methods rely on port and payload-based analysis techniques, both well-known in the field of network traffic classification. These techniques present important limitations that highly reduce their effectiveness, particularly due to the emergence of new dynamic applications and the widespread use of encryption, tunneling, and protocol obfuscation.

To alleviate the shortcomings of port and payload based traffic classification, Machine Learning (ML) techniques have been extensively applied to the problem. ML-based techniques analyze traffic flows by studying statistical patterns in payload-independent traffic features such as packet length and inter-arrival times. ML techniques are roughly classified into two different categories: *supervised* ML and *unsupervised* ML. Traffic classification methods are developed with *supervised* ML techniques, using a model construction or training step in which a mapping between a set of known traffic categories and their corresponding payload-independent traffic features is established. Supervised learning requires a set of labeled traffic flows to construct such a mapping model, which are generally unavailable and difficult to produce. On the other hand, *unsupervised* ML provides an autonomous approach to partition a set of unknown traffic flows into classes of similar characteristics, without relying on labeled traffic flows or training. This breakdown of traffic flows into a reduced number of classes permits to simplify traffic analysis tasks, as it dramatically reduces the number of flows to study.

In this paper we introduce MINETRAC, a novel unsupervised learning approach to identify different classes of flows sharing similar payload-independent characteristics. The unsupervised analysis is accomplished by means of clustering. The objective of clustering is to partition a set of unlabeled instances into homogeneous groups of similar characteristics, based on some similarity measure. In particular, we present a *divide & conquer* clustering approach, in which we combine the evidence of inter-flows structure provided by multiple independent partitions of the same set of flows to build a new inter-flows similarity measure. As we shall explain, the main advantage of this new similarity measure is that it better reflects natural groupings. The clustering approach combines the notions of Sub-Space Clustering [2], Evidence Accumulation Clustering [3], and Hierarchical Clustering [1] to build the new similarity measure and to produce the corresponding clusters.

This clustering technique is further used to build an automatic flow classification model, using a *semi-supervised* ML approach. Semi-supervised learning uses a small amount of labeled instances together with a large amount of unlabeled instances to train a classifier. MINETRAC uses just a small fraction of labeled traffic flows to label the clusters produced by the unsupervised approach. We shall see that even a very small fraction of labeled flows per cluster is good enough to build an accurate classification model. Once the clusters have been labeled any unknown flow can be classified, based on its distance towards the different clusters.

The remainder of the paper is organized as follows: Section II presents a brief state of the art in the field of automatic traffic analysis and classification trough ML, additionally describing

our main contributions. In section III we introduce the core of MINETRAC, describing the different clustering techniques used to retrieve natural groupings. Section IV describes the semi-supervised learning technique used by MINETRAC to construct a fast and accurate traffic classification model. Section V describes the experimental approach taken in the evaluation of MINETRAC. Section VI evaluates MINETRAC in real traffic traces from two different networks, labeled by Ground-Truth techniques [15]. In this section we also compare the performance of our methods against previous proposals for unsupervised traffic analysis, as well as against a rich set of well-known supervised classification approaches. Finally, section VII concludes this work.

II. Related Work & Contributions

The field of automatic traffic analysis and classification trough ML techniques has been extensively studied during the last half-decade. A standard non-exhaustive list of supervised ML-based approaches includes the use of Bayesian classifiers [4], linear discriminant analysis and k-nearest-neighbors [5], decision trees and feature selection techniques [6], and support vector machines [7]. Unsupervised and semi-supervised learning techniques have also been used before for traffic analysis and classification, including the use of k-means, DBSCAN, and AutoClass clustering [8], and a combination of k-means and maximum-likelihood clusters labeling [9]. We refer the interested reader to [10] for a detailed survey on the different ML techniques applied to automatic traffic classification.

MINETRAC presents several advantages with respect to current state of the art in automatic traffic analysis and classification: firstly, it permits to analyze traffic flows in a completely unsupervised fashion, which means that it can be directly plugged-in to any monitoring system and start to work from scratch, without any kind of calibration and/or training step. Secondly. it uses robust clustering techniques to identify natural groupings and avoid general clustering problems such as sensitivity to initialization, specification of number of clusters, detection of particular cluster shapes, or structure-masking by irrelevant features. Thirdly, it performs clustering in very-low-dimensional spaces, avoiding sparsity problems when working with high-dimensional data [1].

The semi-supervised model built on top of the unsupervised approach additionally permits to classify the pre-processed set of flows, using only a reduced fraction of ground-truth flows to label the complete set of unlabeled flows. Finally, the obtained model is extremely simple and permits to classify new flows in real-time, as it only needs to compute the distance between the new flow and the labeled clusters.

III. Unsupervised Traffic Analysis

MINETRAC takes as input a set of n unlabeled traffic flows. Flows are identified by a traditional 5-tuple hashing-key composed of source and destination IP addresses, source and destination ports, and protocol. Let $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$ be this set of n flows. Each flow $\mathbf{y}_i \in \mathbf{Y}$ is described by a set of m payload-independent traffic descriptors or *features*. Let $\mathbf{x}_i \in \mathbb{R}^m$ be the corresponding vector of traffic features describing flow $\mathbf{y}_i$, and $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^{n \times m}$ the complete matrix of features, referred to as the *feature space*.

The unsupervised algorithm is based on clustering techniques applied to $\mathbf{X}$. The objective is to breakdown the set of flows $\mathbf{Y}$ into homogeneous groups of similar characteristics. Unfortunately, even if hundreds of clustering algorithms exist [1], it is very difficult to find a single one that can handle all types of cluster shapes and sizes, or even decide which algorithm would be the best for our particular problem. Different clustering algorithms produce different partitions of data, and even the same clustering algorithm provides different results when using different initializations and/or different algorithm parameters. The lack of robustness is in fact one of the major drawbacks of current clustering techniques. To avoid such a limitation, we have developed a divide and conquer clustering approach, using the notions of *clustering ensemble* and *combination of multiple clusterings* [11]. The idea is novel and appealing: why not taking advantage of the different grouping information provided by multiple partitions of $\mathbf{X}$ to improve clustering accuracy?. Let us briefly introduce the notion of clustering ensemble. A clustering ensemble $\mathbf{P} = \{P_1, \ldots, P_N\}$ consists of a set of multiple partitions P_i produced for the same data. Each of these partitions provides a different and independent evidence of data structure, which can be combined to construct a new measure of similarity that better reflects natural groupings. There are many different ways to produce a clustering ensemble. For example, multiple partitions can be obtained by using different clustering algorithms, or by applying the same clustering algorithm with different parameters and/or initializations. In our approach, we use Sub-Space Clustering (SSC) [2] to produce multiple data partitions, applying the same clustering algorithm to N different sub-spaces $\mathbf{X}_i \subset \mathbf{X}$ of the original space.

A. Building Partitions through Sub-Space Clustering

Each of the N sub-spaces $\mathbf{X}_i \subset \mathbf{X}$ is obtained by selecting r features from the complete set of m attributes. To deeply explore the complete feature space, the number of sub-spaces N that are analyzed corresponds to the number of r-combinations-obtained-from-m. Each partition P_i is obtained by applying DBSCAN [12] to sub-space $\mathbf{X}_i$. DBSCAN is a powerful density-based clustering algorithm that discovers clusters of arbitrary shapes and sizes [1], and it is probably one of the most common clustering algorithms along with the widely known k-means. DBSCAN fits a-priori our unsupervised traffic analysis paradigm, as it is not necessary to specify difficult to set parameters such as the number of clusters to identify. However, it still requires to tune-up two important parameters that define its notion of density, which strongly impact its performance. We shall come back to this issue in the evaluations section. To set the number of dimensions r of each sub-space, we take a very useful property of monotonicity in clustering sets, known as the downward closure property, which basically states that if a collection of instances is a cluster in a r-dimensional space, then it is also part of a cluster in any $(r-1)$ projections of this space. This implies that dense regions of $\mathbf{X}$ will tend to be present in its lowest-

Algorithm 1 SSC-EA Clustering in MINETRAC

1: **Initialization:**
2: Set similarity matrix S to a null $n \times n$ matrix.
3: **for** $t = 1 : N$ **do**
4: P_t = DBSCAN $(\mathbf{X}_t, n_{\min}, \epsilon)$
5: Update $S(i,j)$, $\forall$ pair $\{\mathbf{x}_i, \mathbf{x}_j\} \in C_k$ and $\forall\, C_k \in P_t$:
6: $S(i,j) \leftarrow S(i,j) + \frac{1}{N}$
7: **end for**
8: Transform S into a distance matrix: $S \leftarrow S - 1$.
9: Build a SL tree from S: SLT = SINGLE-LINKAGE (S)
10: Cut SLT at height t_h: P^* = $\{C_k^*\}$ = CUT (SLT, t_h)

dimensional sub-spaces. Using small values for r provides important advantages: firstly, doing clustering in low-dimensional spaces is more efficient and faster than clustering in higher dimensions [1]. Secondly, density-based clustering algorithms such as DBSCAN provide better results in low-dimensional spaces, because high-dimensional spaces are usually sparse, making it difficult to distinguish between high and low density regions. We therefore use $r = 2$ in our SSC algorithm, which gives $N = m(m-1)/2$ partitions.

B. Combining Partitions through Evidence Accumulation

Having produced the N partitions P_i, the question now is how to use the information provided by the obtained clusters $C_k \in P_i$. A possible answer is provided in [3], where authors introduced the idea of multiple-clusterings Evidence Accumulation (EA). EA uses the clustering results of multiple partitions P_i to produce a new inter-patterns similarity measure which better reflects natural groupings. The algorithm follows a **split-combine-merge** approach to discover the underlying structure of data. Let us briefly describe the three steps of this algorithm, particularly adapted to our clustering problem; (i) in the **split** step, the N partitions P_i of the same set of flows are constructed, which in our case they correspond to the partitions obtained by SSC and DBSCAN applied to $\mathbf{X}_i$. (ii) In the **combine** step, a new measure of similarity between flows is produced, using an *association* mechanism; the underlying assumption in EA is that flows belonging to a natural cluster are likely to be co-located in the same cluster in different partitions. Taking the membership of pairs of flows to the same cluster as weights for their association, the N partitions are mapped into a $n \times n$ similarity matrix S, such that $S(i,j) = n_{ij}/N$. The value n_{ij} corresponds to the number of times that the pair of flows described by $\{\mathbf{x}_i, \mathbf{x}_j\}$ was assigned to the same cluster along the N partitions. (iii) In the final **merge** step, any clustering algorithm can be applied to matrix S to obtain a final partition of $\mathbf{X}$ in natural clusters. In our approach we use a simple Hierarchical Clustering (HC) algorithm known as Single-Linkage (SL). HC creates a hierarchy of clusters that can be represented in a tree structure. The root of the tree consists of a single cluster containing all the instances, and the leaves correspond to individual instances. SL builds this tree in an agglomerative fashion: at each step, the algorithm joins together the two clusters which are closest. Cutting the tree at a given height t_h produces the final partition P^* of the approach.

A pseudo-code of the complete unsupervised clustering algorithm is provided in algorithm 1. From now on, we shall refer to this approach as the SSC-EA clustering algorithm. In line 4, the parameter $n_{\min}$ specifies the minimum number of flows that can be classified as a cluster by DBSCAN, and ϵ defines the maximum neighborhood-density distance that permits to group flows into the same cluster. In line 5, C_k corresponds to the different clusters contained in P_t.

IV. Semi-Supervised Traffic Classification

The unsupervised separation of flows into multiple classes simplifies traffic analysis tasks, but it can not be used to automatically recognize the network-service or application that generated each class of flows without any additional information. In this section we develop an automatic flow classification model, using a semi-supervised-learning-based approach on top of the SSC-EA clustering algorithm.

Regarding traffic classification, let us assume a set of M known network applications $L = \{l_1, l_2, \ldots, l_M\}$, where each l_k corresponds to the class or label of each application, such as VoIP, eMail, P2P, etc.. The idea is to build a classification model $\mathcal{F}(\cdot)$ such that, given a flow $\mathbf{y}_i$ described by $\mathbf{x}_i$, the model assigns an estimated label $\hat{l}_k \in L$ to this flow: $\hat{l}_k = \mathcal{F}(\mathbf{x}_i)$. In both supervised and semi-supervised learning for traffic classification, the model $\mathcal{F}(\cdot)$ is constructed by some sort of training step, using a training set of labeled traffic flows. In the case of supervised learning, the training set must be fully labeled, which means that the label of each flow must be known. Supervised-learning-based methods are highly efficient for traffic classification [6]. However, their main drawback is that of requiring a fully labeled set of flows for training. Semi-supervised learning works in a similar way to supervised learning, using a training set to construct a classification model. Nevertheless, for the same size of training set, semi-supervised needs only a small fraction of labeled flows to construct the model, which represents a paramount advantage with respect to supervised-learning.

Figure 1 depicts a complete diagram of MINETRAC. MINETRAC uses a semi-supervised learning approach for traffic classification. Given an **unlabeled** set of flows $\mathbf{Y}$, the model is built as follows: (i) in the first step, $\mathbf{X}$ is computed and then decomposed in clusters C_k^*, using algorithm 1; (ii) in the second step, a fraction λ of flows per-cluster is randomly selected; (iii) in the third step, the labels of only these selected flows are computed and used to classify each of the clusters C_k^*, taking the most frequent traffic class per cluster as label for the corresponding C_k^*. As the number of sampled flows to analyze is very small (as we shall see, using as little as $\lambda = 1\%$ can be good enough), their labels can be obtained by Deep Packet Inspection (DPI) techniques, or even by manual inspection in small network-scenarios. In the fourth and last step, (iv) the centroid $\mathbf{o}_k$ of each cluster C_k^* is computed, obtaining a classification model in the form of $\{\mathbf{o}_k^*, l_j\}$. The reader should note that this model construction through clustering is done in an off-line fashion, and thus it

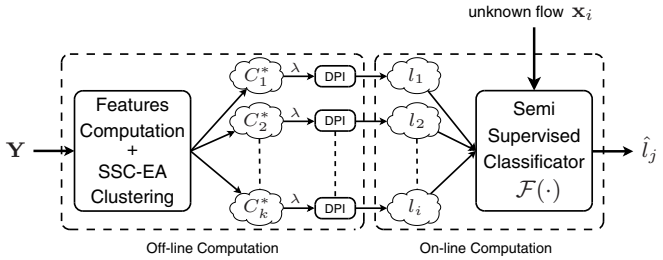


Figure 1. MINETRAC uses SSC-EA to separate flows in clusters, and a small fraction λ of labeled flows to classify each cluster (both steps are computed off-line). The obtained semi-supervised classification model is then used to automatically classify unknown flows in an on-line basis.

does not impact the use of MINETRAC for on-line traffic classification. In order to classify a new unknown traffic flow $\mathbf{x}_i$ we use a distance-based classification rule, associating to $\mathbf{x}_i$ the traffic label $\hat{l}_j$ of the closest cluster:

$$\hat{l}_j = \mathcal{F}(\mathbf{x}_i) = \text{label}\left(\arg\min_k d(\mathbf{x}_i, \mathbf{o}_k^*)\right) \quad (1)$$

where d is the standard euclidean distance. Note now that this classification can be easily achieved in an on-line fashion, as it is only necessary to compute the distance to the centroid of each of the clusters previously labeled.

V. Experimental Approach

A. Traffic Traces

A comprehensive evaluation of MINETRAC is done with real traffic traces from two different traffic repositories: the public UNIBS-2009 traffic-traces repository [17], and the reference traffic repository built in [18] and available upon request. In both cases, traffic was generated by controlled workstations running a Ground Truth (GT) traffic classifier [15], [18]. A GT classifier is a software tool that associates traffic flows with the corresponding application that generated them, probing the kernel of the machine to obtain information on open IP sessions.

The first dataset consists of traces collected at the edge router of a campus network between the 30/09 and the 02/10, 2009. Traffic mainly consists of HTTP, eMail (SSL mainly), P2P (BitTorrent, Edonkey), and VoIP (Skype). The dataset that we use consists of 2000 flows taken from the first day. We randomly sample 500 flows for each of the four traffic classes: HTTP, SSL, P2P, and VoIP. Similar to [8], we use an equal number of flows for each application to fairly evaluate the clustering ability of the SSC-EA algorithm, avoiding a biased analysis due to highly unbalanced representation of classes. We refer to this dataset as the UNIBS dataset.

The second dataset consists of 43 hours of traffic captured in a controlled and separated access network. The dataset contains traffic from different P2P clients (Emule, LimeWire, Azureus), File Hosting/Download applications, VoIP (Skype), HTTP, eMail (POP3 mainly), as well as monitoring traffic from network devices. As before, we sample 500 flows per each of the 8 traffic applications, obtaining a 4000 flows dataset. We shall refer to this dataset as the VALTC dataset.

B. Flows and Features

We use NetMATE [19] to process packet traces, identify flows, and compute feature values. Flows are identified by the traditional 5-tuple hashing-key $\{\text{IP}_{\text{src/dst}} : \text{Port}_{\text{src/dst}} : \text{Proto}\}$. Flows are bidirectional and have a limited duration. We consider UDP and TCP flows with at least one packet in each direction and at least one byte of payload. This excludes flows without payload and requests without responses. UDP flows are terminated by a flow timeout. TCP flows are terminated upon proper connection tear-down or after a timeout. We take a 600 sec. flow timeout, a default value used in previous work [6].

We use the same set of basic 22 payload-independent traffic features previously used in [6]. The list includes protocol, flow duration, flow volume in bytes and packets, packet length (minimum, mean, maximum, and standard deviation), and inter-arrival time (minimum, mean, maximum, and standard deviation). As traces contain both directions of the flows, features are computed for both directions.

C. Evaluation Algorithms

We evaluate both components of MINETRAC, namely the SSC-EA clustering technique and the semi-supervised classification model. Regarding clustering, we study the performance and accuracy of SSC-EA to produce homogeneous clusters, and compare its performance against two well-known clustering approaches previously used for traffic analysis in [8]: DBSCAN and k-means.

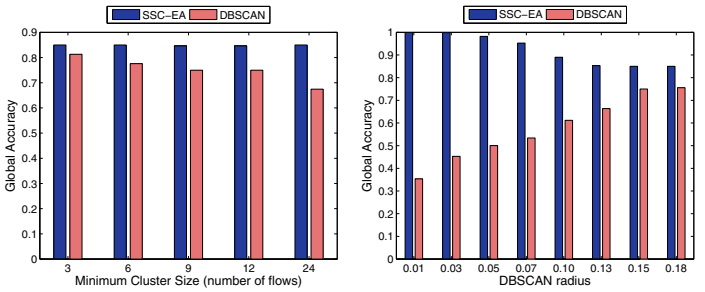
As regards traffic classification, we compare MINETRAC against two similar semi-supervised classification models built upon DBSCAN and k-means, and against five standard supervised-learning-based approaches previously used in the literature: Multi-Layer Perceptron (MLP) Neural Networks, Naive Bayes (NB), Decision Trees (C45), Support Vector Machines (SVM), and Locally-Weighted-based Learning (LWL). We use the Weka Machine-Learning software tool [16] to calibrate these five learning-based algorithms and to perform the evaluations. Even though we do not explain the particular details of each of these algorithms, we address the interested reader to the survey [10] and to the Weka documentation [16] for additional information.

D. Evaluation Criteria

To asses the quality of the clustering results provided by SSC-EA, we employ two traditionally used performance metrics [8]: Global Accuracy (GA) and Average per-Cluster Homogeneity (ACH):

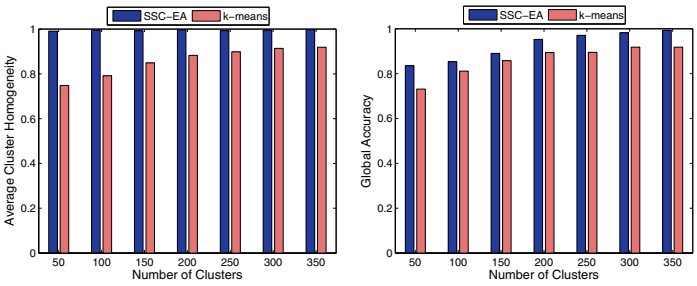
$$\text{GA} = \frac{\sum_{k=1}^{n_{\text{cls}}} TP(k)}{n}, \quad \text{ACH} = \frac{\sum_{k=1}^{n_{\text{cls}}} \frac{TP(k)}{n(k)}}{n_{\text{cls}}} \quad (2)$$

Both criteria determine how accurate is the algorithm to produce homogeneous clusters, i.e., clusters that contain a single traffic class. To label a cluster and only for evaluation purposes, we take the most frequent traffic label among all of its flows. GA indicates the percentage of correctly classified



(a) SSC-EA vs DBSCAN ($\epsilon = 0.15$). (b) SSC-EA vs DBSCAN ($n_{\min} = 9$).

Figure 2. GA for SSC-EA and DBSCAN, changing $n_{\min}$ and ϵ.



(a) Average per-cluster homogeneity. (b) Global accuracy.

Figure 3. ACH and GA for SSC-EA vs k-means, for different number of identified clusters.

flows among the total number of flows n. ACH indicates the average percentage of correctly classified flows per cluster. In equation (2), $TP(k)$ corresponds to the number of correctly classified flows in cluster C_k (True Positives), $n(k)$ is the size of C_k, and n_{cls} is the total number of clusters.

To evaluate the semi-supervised classification model, we consider two additional per-class metrics: Recall and Precision. Recall R_i is the number of flows from class $i = 1, \ldots, M$ correctly classified (TP_i), divided by the number of flows in class i (n_i). Precision P_i is the percentage of flows correctly classified as belonging to class i among all the flows classified as belonging to class i, including true and false positives (FP_i). Recall and precision are two widely used performance metrics in classification. Precision permits to measure the fidelity of the classification model regarding each particular class, whereas recall measures the per-class accuracy. As a summary of classification performance of each algorithm, we shall use the Global Accuracy metric:

$$\text{R}_i = \frac{TP_i}{n_i}, \quad \text{P}_i = \frac{TP_i}{TP_i + FP_i}, \quad \text{GA} = \frac{\sum_{i=1}^{M} TP_i}{n} \tag{3}$$

E. Features Selection

Using an extensive list of traffic features is not always the best strategy, as it may negatively impact clustering and classification results. As we claimed before, using more features increments the dimensionality of the feature space, introducing sparsity issues. At the same time, using irrelevant or redundant features may diminish performance in the practice. We shall therefore evaluate the impact of feature selection on the SSC-EA clustering algorithm.

There are different search strategies and evaluation criteria to construct a sub-set of traffic features. Regarding search strategies, the idea is to test different sub-sets of features, studying local changes in the particular evaluation criterion when adding or removing features. The evaluation criterion permits to test the goodness of a particular sub-set.

In this paper we apply a widely used evaluation criterion to construct reduced sub-sets of features: correlation-based evaluation [14]. This approach basically selects sub-sets of features that are poorly correlated among each other, but highly correlated to the classes of traffic. As search strategy, we use Best-First (BF) search [6]; BF is similar to a standard greedy exploration, but it has the ability to do backtracking, i.e., it basically keeps the previously evaluated sub-sets so as to avoid local maximum/minimum results when there is no local improvement.

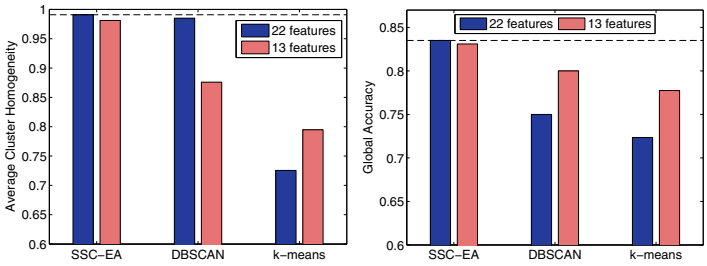
VI. Evaluation and Discussion

In the first part of the evaluation, we use the UNIBS dataset to study the accuracy, homogeneity, and sensitivity of the SSC-EA clustering approach when considering different number of clusters, different clustering parameters for the underlying density-based clustering algorithm (i.e., DBSCAN parameters $n_{\min}$ and ϵ), and feature selection techniques. To evidence the advantages of our SSC-EA algorithm with respect to previous work on clustering for traffic analysis [8], we compare obtained results against those obtained with k-means and DBSCAN.

A. SSC-EA vs DBSCAN vs k-means

We begin by studying the influence of $n_{\min}$ and ϵ in both the SSC-EA and the standard DBSCAN algorithm. In the case of SSC-EA we have an additional parameter to vary, which corresponds to the height t_h where the tree is cut to obtain the final clusters. We do not present the evaluation results obtained with SSC-EA when varying t_h, but we report that both accuracy and clusters homogeneity are hardly impacted by t_h: indeed, the variation is almost negligible when moving t_h from the lowest ($t_h = 0.01$) to the highest ($t_h = 0.9$) values. We believe that this effect is due to the high clusters homogeneity achieved with SSC-EA, but a deeper evaluation of this phenomenon is part of our ongoing work.

Figure 2 depicts the global accuracy obtained with SSC-EA and DBSCAN when changing $n_{\min}$ and ϵ. The standard DBSCAN algorithm does not necessary assign every instance to a cluster; in order to evaluate its performance, we follow the same approach used in [8], where every flow that is not assigned to a cluster is considered as noise. Figure 2(a) shows that the SSC-EA algorithm is immune to $n_{\min}$, as the obtained GA remains constant at near 85% for all the considered range (both variation ranges for $n_{\min}$ and ϵ were taken from [8]). In the case of DBSCAN, it is clear that producing outliers (i.e., flows outside the clusters) has a negative impact on its performance, as these outliers can not be directly tied to any particular traffic class. We claim that better performance could be obtained with DBSCAN if outliers were assigned



(a) Average per-cluster homogeneity. (b) Global accuracy.

Figure 4. Impact of feature selection on accuracy and cluster homogeneity. The subset is obtained by Best-First search and Correlation-based evaluation.

to its closest clusters, but this study is out of the scope of current paper. All we can say for sure is that the SSC-EA algorithm provides strong robustness with respect to $n_{\min}$, at least in the considered variation range. Figure 2(b) shows a marked variation of accuracy for the SSC-EA algorithm when using very small values for ϵ. However, its value remains constant at near 85% for bigger radius. Regarding DBSCAN, the algorithm identifies many outliers when changing either $n_{\min}$ or ϵ, which certainly impacts the attained global accuracy.

Figure 3 depicts the average per-cluster homogeneity and the global accuracy obtained by SSC-EA and k-means for different numbers of identified clusters. Changing the value of ϵ in SSC-EA permits to produce different clustering trees, which additionally permits to identify a different number of clusters. Figure 3(a) shows that the average cluster homogeneity obtained with SSC-EA is almost perfect, independently of the identified number of clusters. For k-means, clusters homogeneity strongly depends on the number of clusters to construct. Regarding accuracy, both algorithms improve results when using more clusters; however, using a large number of clusters is counterproductive, as it reduces the practical interest of doing clustering for traffic analysis. In any case, we can see that the SSC-EA algorithm still provides high accuracy for a limited number of clusters, about 85% with 50 clusters.

B. Feature Selection

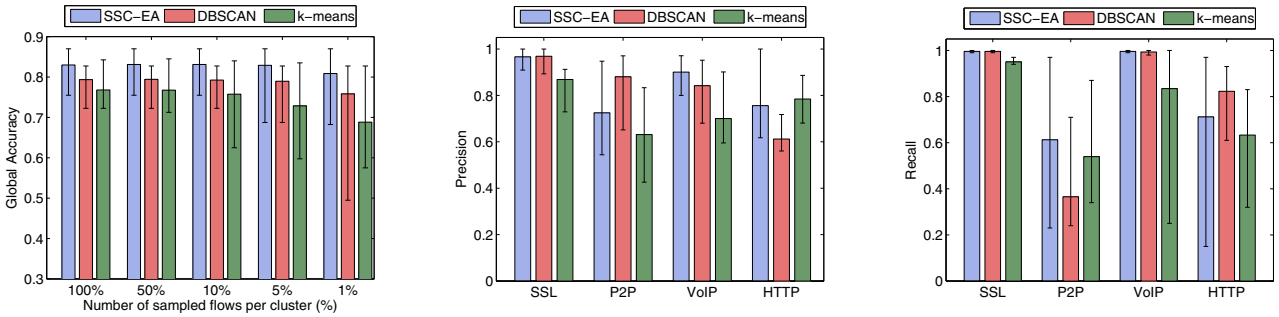
We now evaluate the impact of feature selection on the clustering algorithms. Figure 4 depicts the average per-cluster homogeneity and the global accuracy for the three clustering algorithms, using both the complete set of 22 features, and a reduced sub-set of 13 features, obtained by Best First search and Correlation-based subset evaluation. Selected features correspond mainly to flow volume and packet size features, which is not surprising, as inter-arrival times tend to be bundled to the particular condition of the transport network and are therefore less related to the traffic classes. For DBSCAN and SSC-EA, we take $n_{\min} = 9$ and $\epsilon = 0.15$, which produces a reasonable number of clusters, about 50 when using 22 features. For k-means we use therefore $k = 50$. Both accuracy and cluster homogeneity remain almost unchanged for the SSC-EA algorithm, while vary between 7% and 10% for DBSCAN and k-means. It is interesting to appreciate how the accuracy of both algorithms improves when removing irrelevant features. As we claimed before, the SSC-EA algorithm is more robust against irrelevant or redundant features than standard clustering algorithms. Another interesting observation is that the number of clusters obtained by SSC-EA and DBSCAN falls to about 30 clusters when using 13 features.

C. Semi-Supervised Traffic Classification

Let us now evaluate the semi-supervised classification model of MINETRAC, built on top of the SSC-EA algorithm. To train and to test the classification model, we separate the UNIBS dataset into a training and a testing set. The training set accounts for the 80% of the flows, while the remaining 20% is used as testing flows. All the evaluations presented in this subsection use 5-fold cross-validation, which means that we train and we test the model for 5 different training/testing sets. In addition, we use the reduced set of 13 features obtained by Feature Selection. As we explained before, using this reduced set additionally reduces the number of identified clusters to about 30. Finally, we compare the performance of our approach with that obtained with the same semi-supervised classification technique, but using DBSCAN ($\epsilon = 0.15$, $n_{\min} = 9$) and k-means ($k = 30$) to construct the classification model.

MINETRAC uses a reduced fraction λ of flows per cluster to establish their corresponding labels. We shall therefore evaluate the impact of λ as well. Figure 5(a) depicts the global accuracy of the three classification models as a function of λ. As the flows used for labeling each cluster are randomly chosen, we have run the classification algorithm 10 times for each of the 5-fold evaluation datasets and for each of the 5 different fraction values that we have considered, $\lambda = \{1, 0.5, 0.1, 0.05, 0.01\}$. Depicted results include the obtained mean global accuracy, as well as the minimum and maximum values. The first interesting observation is that MINETRAC performs with high accuracy even when using a fraction as little as 1% of labeled flows per cluster. At the same time, the advantages of the SSC-EA clustering algorithm previously evidenced provide a better classification performance than traditional approaches.

To conclude with this analysis, figures 5(b) and 5(c) present the values of precision and recall obtained with the three models, for each of the four different traffic classes. The fraction of sampled flows λ is 5%. As in figure 5(a), clustering parameters are $k = 30$, $\epsilon = 0.15$, and $n_{\min} = 9$. P2P traffic is systematically misclassified by the three models, obtaining a quite low recall. HTTP traffic presents a relatively better performance than P2P traffic, but still provides poor results. Note however that both the SSL and the VoIP traffic are accurately classified, obtaining precision and recall values close to 100%. The low recall and precision obtained for P2P traffic mean both that many P2P flows are misclassified, but also that non-P2P flows are classified as P2P traffic; the same happens with HTTP traffic. The main problem comes from the way that HTTP traffic is labeled; in fact, GT techniques keep only the name of the application that generated the flows, but do not distinguish among different protocols or services that may run on the same application. In the case of HTTP the application is a web browser, and many different types of traffic can be



(a) $k = 30$, $\epsilon = 0.15$, $n_{\min} = 9$ (b) Precision ($\lambda = 5\%$) (c) Recall ($\lambda = 5\%$)

Figure 5. Semi-supervised classification using MINETRAC (SSC-EA) vs DBSCAN vs k-means. Classification Accuracy, Precision, and Recall for P2P (BitTorrent), VoIP (Skype), SSL (eMail), and HTTP traffic.

used on top of a web browser, such as FTP, chat traffic, streaming traffic, etc.. There is therefore a certain mixture of traffic behaviors between P2P and HTTP traffic, which justifies the low performance for both classes. Nevertheless, results obtained by MINETRAC are slightly better than those obtained with traditional clustering techniques.

D. MINETRAC vs Supervised Traffic Classification

In the last part of the paper we compare the classification performance achieved by MINETRAC against the one obtained by the five supervised-learning based approaches: MLP (Neural Networks), NB (Naive Bayes), C45 (Decision Tree), SVM (Support Vector Machines), and LWL (Locally-Weighted-based Learning). As we did before, all the presented results in this subsection were obtained by 5-fold cross-validation. However, we shall depict only the average values and not the worst and best cases, so as to keep figures legible.

We shall begin by studying results in the UNIBS dataset. Figure 6(a) depicts the global accuracy obtained by the six approaches. We still use in this evaluation the sub-set of 13 features selected by BF search and Correlation-based evaluation. MLP and the C45 decision tree clearly outperform the rest of the methods, achieving global accuracy values close to 100%. SVM-based classification performance is also very high, about 90%. MINETRAC global accuracy is very close to that obtained with these three methods, and even though there is a performance gap of about 15% with respect to MLP and C45, the reader should remember that MINETRAC does not have access to a fully labeled set of flows. On the contrary, in these evaluations we take only a 5% of labeled flows as input for labeling the clusters produced by SSC-EA. NB and LWL perform worse than MINETRAC, even though they use a fully labeled training set of flows. While this poor performance for traffic classification has been evidenced before [6], we wanted to include these supervised approaches to show that MINETRAC is capable of outperforming fully supervised classification models.

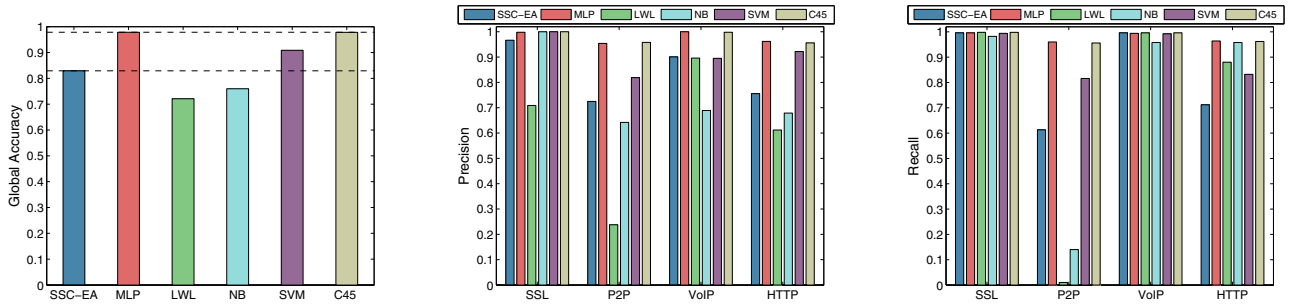
Regarding precision and recall depicted in figures 6(b) and 6(c), we appreciate as before lower performance for P2P and HTTP traffic, particularly for MINETRAC, NB, and LWL. However, both MLP and C45 classifiers are more robust and achieve proper separation between both classes of traffic. It is interesting to see that MINETRAC can perform very close to MLP and C45 for SSL and VoIP, which opens the question about the impact of the quality of the ground-truth generated by current GT techniques on traffic analysis.

Let us now perform the same evaluation and comparison in the VALTC dataset. The SSC-EA clustering algorithm uses now $\epsilon = 0.05$ and $n_{\min} = 9$, which produces a higher number of clusters, about 100. We perform selection of features also in this case, using exactly the same search strategy and evaluation criterion. It is not surprising that once again, packet sizes and flow volumes are prioritized over time-based information. The constructed sub-set has 14 features for VALTC, 10 of them match the list of 13 features selected in UNIBS.

Figure 7(a) shows a relatively lower accuracy for all the methods in the VALTC dataset. The relation among the different global accuracy values obtained by the different algorithms is the same as before, but the performance gap between the most accurate algorithm (C45) and MINETRAC is about 25% now. Figures 7(b) and 7(c) show that MINETRAC can pretty well distinguish between two out of the three P2P clients (Azureus, BitTorrent client and LimeWire, Gnutella client), while achieving poor performance for Emule traffic. It is very interesting to see that HTTP traffic in VALTC is poorly classified with all of the approaches, obtaining a slightly better performance with MINETRAC. We believe that this misbehavior of the six algorithms is a result of the GT technique, basically because there is an important mixture of traffic classes within the HTTP one. Finally, despite achieving poorer performance in VALTC than in UNIBS, we can appreciate a similar behavior of the different algorithms in both sets: MLP and C45 clearly outperform MINETRAC, SVM provides similar results, and NB and LWL provide worse performance.

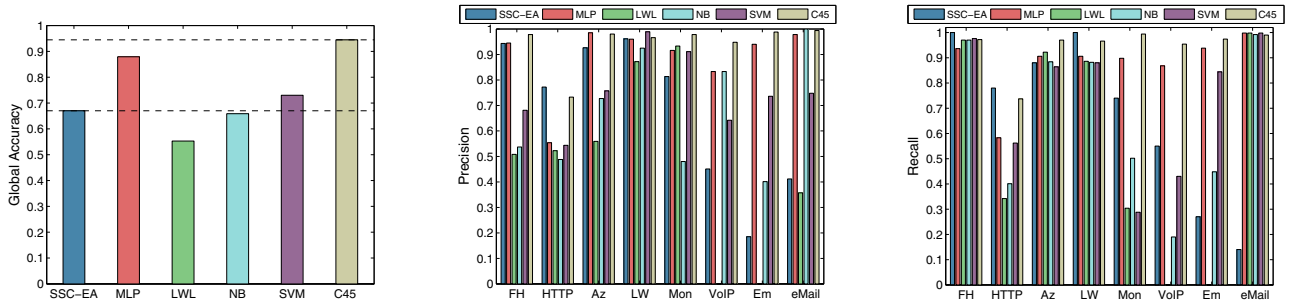
VII. Concluding Remarks

In this paper we introduced MINETRAC, a combination of unsupervised and semi-supervised learning algorithms for automatic traffic analysis and classification. Our novel SSC-EA clustering approach proved to be more robust, consistent, and accurate for traffic analysis and classification than two well-known clustering approaches previously used: DBSCAN and k-means, at least in our evaluations. In particular, the method showed to produce better and more stable results



(a) $\epsilon = 0.15$, $n_{\min} = 9$ (b) Precision ($\lambda = 5\%$) (c) Recall ($\lambda = 5\%$)

Figure 6. Semi-supervised classification using MINETRAC (SSC-EA) vs Supervised Learning Approaches. Classification Accuracy, Precision, and Recall for P2P (BitTorrent), VoIP (Skype), SSL (eMail), and HTTP traffic.



(a) $\epsilon = 0.05$, $n_{\min} = 9$ (b) Precision ($\lambda = 5\%$) (c) Recall ($\lambda = 5\%$)

Figure 7. Semi-supervised classification using MINETRAC (SSC-EA) vs Supervised Learning Approaches. Classification Accuracy, Precision, and Recall for File Hosting applications (FH), HTTP traffic, different P2P clients (Azureus, LimeWire, Emule), Wireless Monitoring Traffic (Mon), VoIP (Skype), and eMail (POP3).

when changing different core parameters. The semi-supervised model built on top of SSC-EA additionally showed good performance classification, comparable to that obtained with traditional fully supervised-based algorithms, but using just a 5% of labeled flows for training issues.

Our evaluations showed that clustering-based approaches and semi-supervised learning techniques are an attractive yet not deeply explored alternative for automatic classification. Even though we have not evaluated this possibility, we claim that the use of unsupervised traffic analysis algorithms is fundamental if we want to develop classification models that highly adapt to a wide variety of network scenarios. For example, we could envisage running periodically MINETRAC to *track* the evolution of the different identified clusters. There is an interesting literature in clustering evolving data [13], which could be exploited in this direction.

Our evaluations also revealed a possible problem with current Ground Truth techniques: as it was evidenced for the case of HTTP traffic, keeping only the application that generated the flows, without discriminating among the different protocols or services that may run on the same application produces bad-quality ground-truth data.

Finally, we claim that after trained, MINETRAC permits to classify new flows in real-time, as it only needs to compute a reduced number of distance values to provide a verdict.

REFERENCES

[1] A. K. Jain, "Data Clustering: 50 Years Beyond K-Means", in *Pattern Recognition Letters*, vol. 31 (8), pp. 651-666, 2010.

[2] L. Parsons et al., "Subspace Clustering for High Dimensional Data: a Review", in *ACM SIGKDD Expl. Newsletter*, vol. 6 (1), pp. 90-105, 2004.

[3] A. Fred et al., "Combining Multiple Clusterings using Evidence Accumulation", in *IEEE Trans. Patt. Anal. & Mach. Intell.*, vol. 27 (6), 2005.

[4] A. Moore, D. Zuev, "Internet Traffic Classification using Bayesian Analysis Techniques", in *Proc. ACM SIGMETICS*, 2005.

[5] M. Roughan et al., "Class-of-Service Mapping for QoS: a Statistical Signature-Based Approach to IP Traffic Classification", in *IMW*, 2004.

[6] N. Williams el al., "A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification", in *ACM SIGCOMM Computer Communication Review*, vol. 36 (5), 2006.

[7] S. Valenti et al., "Accurate, Fine-Grained Classification of P2P-TV Applications by Simply Counting Packets", in *Proc. 1st TMA Workshop*, 2009.

[8] J. Erman et al., "Traffic Classification using Clustering Algorithms", in *Proc. MineNet*, 2006.

[9] J. Erman et al., "Semi-Supervised Network Traffic Classification", in *Proc. ACM SIGMETRICS*, 2007.

[10] T. Nguyen et al., "A Survey of Techniques for Internet Traffic Classification using Machine Learning", in *IEEE Comm, Surv. & Tut.*, 2008.

[11] A. Strehl et al., "Cluster Ensembles - a Knowledge Reuse Framework for Combining Multiple Partitions", in *J. Mach. Lear. Res.*, vol. 3, 2002.

[12] M. Ester et al., "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", in *Proc. ACM SIGKDD*, 1996.

[13] C. Aggarwal, J. Han, J. Wang, P. Yu, "A Framework for Clustering Evolving Data Streams", in *Proc. VLDB*, 2003.

[14] M. Hall, "Correlation-based Feature Selection for Machine Learning", *PhD Dissertation*, Waikato University, 1998.

[15] F. Gringoli et al., "GT: Picking Up the Truth from the Ground for Internet Traffic", in *ACM Comp. Comm. Review*, vol. 39 (5), pp. 13-18, 2009.

[16] "Weka: Data Mining Software in Java", at http://www.cs.waikato.ac.nz/ml/weka/ (accessed 02/11).

[17] "The UNIBS Anonymized 2009 Internet Traces", at http://www.ing.unibs.it/ntw/tools/traces (accessed 02/11).

[18] G. Szabo et al., "On the Validation of Traffic Classification Algorithms", in *Proc. 9th PAM*, 2008.

[19] S. Zander "NetMATE - Network Measurement and Accounting System", at http://sourceforge.net/projects/netmate-meter (accessed 02/11).