On the Use of Sub-Space Clustering & Evidence Accumulation for Traffic Analysis & 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, France
[2]Universite de Toulouse; UPS, INSA, INP, ISAE; LAAS; F-31077 Toulouse, France
Email: {pcasashe, jmazel, owe}@laas.fr

***Abstract*—Driven by the well-known limitations of port 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 a novel unsupervised approach to identify 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. Our approach 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. The technique is further used to build an automatic flow classification model, using a semi-supervised-learning-based approach. The approach uses only a reduced fraction of labeled flows to map the identified clusters into their associated most-probable originating application, which strongly simplifies its calibration. We evaluate the performance of our techniques using real traffic traces, additionally comparing their performance against previously proposed clustering-based classification methods.**

***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. Traffic classification methods are developed with *supervised* ML techniques, using a model construction or learning 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. Another category of ML techniques is represented by *unsupervised* learning, an autonomous approach that permits to partition a set of unknown traffic flows into classes of similar characteristics, without relying on labeled traffic flows or learning. 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 a novel unsupervised learning approach to identify different classes of flows sharing similar payload-independent features. 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* learning-based approach. Semi-supervised learning uses a small amount of labeled instances together with a large amount of unlabeled instances to train a classifier. In our particular application, we use a small fraction of labeled traffic flows to label the clusters produced by our 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 our proposal, describing the different clustering techniques

used to accurately retrieve natural groupings. Section IV describes the semi-supervised learning technique that we use to construct a fast and accurate traffic classification model. Section V evaluates our proposals in real network traffic from the public UNIBS traffic repository [14], accurately labeled by Ground-Truth techniques [13]. In this section we also compare the performance of our methods against previous proposals for unsupervised analysis. Finally, section VI 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 reader to [10] for a detailed survey on the different ML techniques applied to automatic traffic classification.

Our approach presents several advantages w.r.t. current state of the art in automatic traffic analysis and classification: firstly, it permits to analyze traffic shares 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 our 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 unknown flows in real-time, as it only needs to compute the distance between the new flow and the labeled clusters.

III. Unsupervised Traffic Analysis

The unsupervised traffic analysis algorithm 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 $\mathbf{Y} = \{\mathbf{y}_1, .., \mathbf{y}_n\}$ 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 $\mathbf{X} = \{\mathbf{x}_1, .., \mathbf{x}_n\} \in \mathbb{R}^{n \times m}$ the complete matrix of features, referred to as the *feature space*.

Our 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 information provided by multiple partitions of $\mathbf{X}$ to improve clustering accuracy?. Let us briefly introduce the notion of clustering ensemble. A clustering ensemble $\mathbf{P}$ 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 highly impacts 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-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 pro-

vide better results in low-dimensional spaces, because high-dimensional spaces are usually sparse, making it difficult to distinguish between high and low density regions. In general, working in low-dimensional spaces avoids sparsity issues such as masking, density-distortion, etc.. 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, the question now is how to use the information provided by the obtained clusters. 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 for our clustering approach:

(1) In the **split** step, the N partitions P_i of the same dataset are generated, which in our case they correspond to the partitions obtained by SSC and DBSCAN.

(2) 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.

(3) 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 together. Cutting the tree at a given height t_h produces the final partition P^* of our 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 minimum neighborhood-density distance that permits to group flows into the same cluster. In line 5, the C_k corresponds to the different clusters contained in P_t.

Algorithm 1 SSC-EA-based Clustering

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^* = CUT (SLT, t_h)

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.

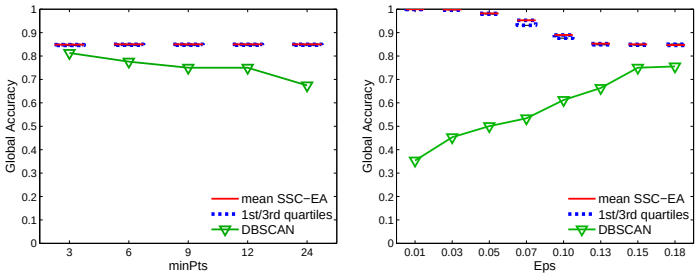
Semi-supervised learning works in a similar way to supervised learning, using a training set to construct a classification model. However, for the same size of training set, semi-supervised needs only a small fraction of labeled flows to construct the model, which represents a paramount advantage w.r.t. supervised-learning, where all the flows of the training set must be labeled.

Given an unlabeled training dataset $\mathbf{X}_{\text{train}}$, the classification model is built as follows: (i) in the first step, $\mathbf{X}_{\text{train}}$ is decomposed in clusters C_k, using SSC-EA clustering; (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 used to classify each of the clusters C_k, taking the most frequent traffic class per cluster as label for the corresponding C_k; (iv) in the last step, the centroid $\mathbf{o}_k$ of each C_k is computed, obtaining a classification model in the form of $\{\mathbf{o}_k, l_k\}$, where $l_k \in \{\text{app}_1, \text{app}_2, .., \text{app}_M\}$ is to the label associated to cluster C_k, and app_i corresponds to one of the M traffic classes present in the flows described by $\mathbf{X}_{\text{train}}$.

In order to classify a new unknown traffic flow $\mathbf{x}$ we use a distance-based classification rule, associating to $\mathbf{x}$ the traffic class of the closest cluster, using the standard euclidean distance d: $\text{class}(\mathbf{x}) = \text{class}(\arg\min_k d(\mathbf{x}, \mathbf{o}_k))$.

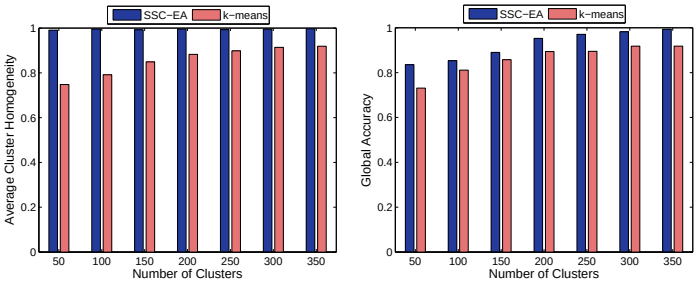
V. Evaluation and Discussion

In this section we evaluate the SSC-EA clustering technique and the semi-supervised traffic classification model. We use real traffic from the public UNIBS-2009 traffic-traces repository [14]; traces were collected at the edge router of the campus network of Brescia's University between the 30/09 and the 02/10, 2009. Traffic mainly consists of HTTP, Mail (SSL mainly), P2P (BitTorrent, Edonkey), and VoIP (Skype), and was generated by a set of workstations running a Ground Truth (GT) traffic classifier [13]. A GT classifier is a software tool that accurately associates traffic flows with the corresponding application that generated them, probing the kernel of the machine to obtain information on open IP sessions. The dataset that we use in the following evaluations consists of 2000 flows taken from the first day of traffic. We randomly sample 500 flows for each of the four aforementioned traffic classes: HTTP, SSL, P2P, and VoIP. Similar to [8], we use an equal



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

Figure 1. GA for SSC-EA and DBSCAN, changing $n_{\min}$ and ϵ.



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

Figure 2. ACH and GA for SSC-EA vs k-means, for different number of identified clusters.

number of flows for each application to fairly evaluate the clustering ability of our algorithm, avoiding a biased analysis due to highly unbalanced cluster sizes.

A. Flows and Features

We use NetMATE [15] to process packet traces, identify flows, and compute feature values. Flows are identified by the traditional 5-tuple hashing-key $\{\mathrm{IP}_{\mathrm{src/dst}} : \mathrm{Port}_{\mathrm{src/dst}} : \mathrm{Protocol}\}$. Flows are bidirectional and have a limited duration. UDP flows are terminated by a flow timeout. TCP flows are terminated upon proper connection tear-down or after a timeout. We consider a 600 second flow timeout, a default value used in previous work [6]. We consider only 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. We use the same set of 22 payload-independent traffic features previously used in [6], as these are simple and well understood traffic descriptors. The list includes protocol, flow duration, flow volume in bytes and packets, packet length (minimum, mean, maximum, and standard deviation), inter-arrival time between packets (minimum, mean, maximum, and standard deviation). As traces contain both directions of the flows, features are computed in both directions.

B. Evaluation Criteria

In order to asses the quality of the clustering results produced by SSC-EA, we employ two traditionally used indexes: the Global Accuracy (GA) and the Average per-Cluster Homogeneity (ACH). 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, we simply take the most frequent traffic class among all of its flows. GA indicates the percentage of correctly classified flows among the total number of flows n. ACH indicates the average percentage of correctly classified flows per cluster. If we define $TP(k)$ as the number of correctly classified flows in cluster C_k, $n(k)$ as the size of C_k, and n_{clusts} as the total number of clusters, then we may express GA and ACH as:

$$\mathrm{GA} = \frac{\sum_k TP(k)}{n}, \mathrm{ACH} = \frac{\sum_k \frac{TP(k)}{n(k)}}{n_{\mathrm{clusts}}}, \mathrm{R}_i = \frac{TP_i}{n_i}, \mathrm{P}_i = \frac{TP_i}{TP_i + FP_i}$$

To evaluate the semi-supervised classification model, we consider two additional per-class indexes: Recall R_i is the number of flows from class i 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).

Besides evaluating the quality of the SSC-EA algorithm, we shall compare its performance against two well-known clustering approaches, previously used for traffic classification in [8]: DBSCAN and k-means.

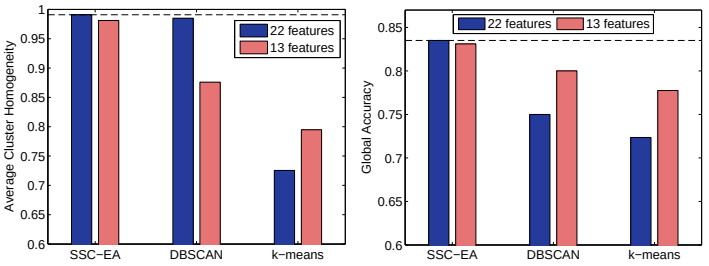
C. SSC-EA vs DBSCAN vs k-means

Figure 1 depicts the global accuracy obtained with SSC-EA and DBSCAN when changing the two input parameters of DBSCAN, namely $n_{\min}$ and ϵ. 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 therefore plot the mean value and the quartiles obtained for different heights t_h, going from the lowest ($t_h = 0.01$) to the highest ($t_h = 0.9$) values. The 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 1.(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 addition, the difference in accuracy when cutting the tree at different heights is negligible. Both observations provide a first evidence of strong robustness of the SSC-EA algorithm against some of its parameters.

Figure 1.(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 distances. Regarding DBSCAN, the algorithm identifies many outliers (flows outside the clusters) when changing both parameters, which certainly impacts the attained GA. We claim that better performance could be obtained with DBSCAN if outliers were assigned to its closest clusters, but this study is out of the scope of this paper.

Figure 2 depicts the average per-cluster homogeneity and the global accuracy obtained for 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 2.(a) shows that the average cluster homogeneity obtained with SSC-EA is almost perfect, independently of 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.



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

Figure 3. Impact of feature selection on accuracy and cluster homogeneity. The subset is obtained by Best-First search and Correlation-based evaluation.

D. Feature Selection

We now evaluate the impact of feature selection on the clustering algorithms. Using a large list of traffic features is not always the best choice, 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.

Figure 3 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 set of 13 features, obtained by Best First search and Correlation-based subset evaluation [6]. This approach basically selects features that are poorly correlated with each other and highly correlated to the classes of traffic. 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.

E. Semi-Supervised Traffic Classification

Let us now evaluate the semi-supervised classification model, built on top of the SSC-EA algorithm. To train and to test the classification model, we separate the complete set of flows 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.

Figure 4.(a) depicts the global accuracy of the three classification models as a function of the fraction λ of flows used for training. 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 $\lambda = \{1, 0.5, 0.1, 0.05, 0.01\}$. Depicted results include the obtained mean global accuracy, as well as the minimum and maximum values. The first interesting observation is that the semi-supervised approach performs with high accuracy even when using a fraction as small as 1% of labeled flows per cluster. At the same time, the advantages of the SSC-EA clustering algorithm previously evidenced also provide a better classification performance than traditional approaches.

To conclude with the evaluation section, figures 4.(b) and 4.(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%. P2P traffic is systematically misclassified by the three models, obtaining a lower true positives rate (recall). Note however that both the SSL and the VoIP traffic are accurately classified, obtaining precision and recall values close to 100%. Finally, HTTP traffic presents a relatively better performance than P2P traffic, but still provides poor results. A deeper evaluation of these results is part of our ongoing work.

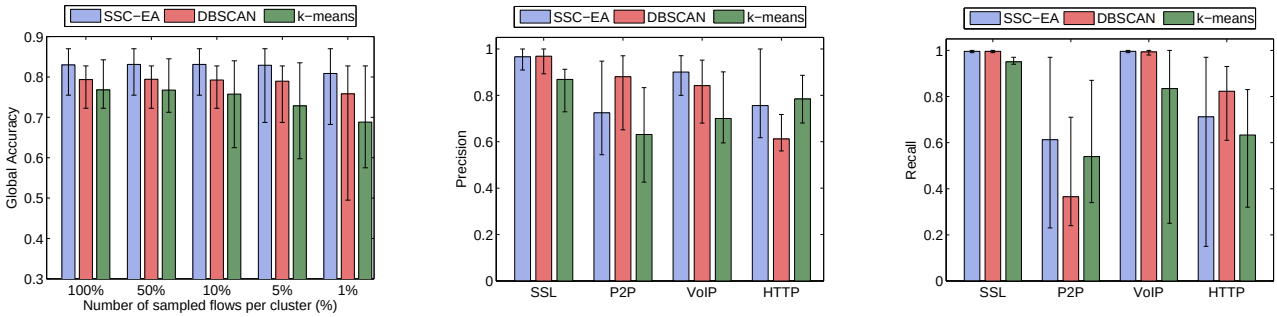
VI. Concluding Remarks

In this paper we addressed the problem of unsupervised and semi-supervised traffic analysis and classification via clustering. We introduced the SSC-EA clustering algorithm, a novel clustering approach which 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. The semi-supervised model built on top of the SSC-EA algorithm additionally showed high performance classification, using just a 5% of labeled flows for training issues. We claim that this classification model permits to classify new unknown flows in real-time, as it only needs to compute about 30 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.



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

Figure 4. Semi-supervised classification using SSC-EA vs DBSCAN vs k-means. Classification Accuracy, Precision, and Recall for P2P (BitTorrent), VoIP (Skype), SSL (eMail), and HTTP traffic.

[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 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] 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.

[14] "The UNIBS Anonymized 2009 Internet Traces", at http://www.ing.unibs.it/ntw/tools/traces (accessed 01/11).

[15] S. Zander "NetMATE - Network Measurement and Accounting System", at http://sourceforge.net/projects/netmate-meter (accessed 01/11).