AN INVESTIGATION OF A MULTI-OBJECTIVE GENETIC ALGORITHM APPLIED TO ENCRYPTED TRAFFIC IDENTIFICATION

by

Carlos Bacquet

Submitted in partial fulfillment of the
requirements for the degree of
Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2010

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "AN INVESTIGATION OF A MULTI-OBJECTIVE GENETIC ALGORITHM APPLIED TO ENCRYPTED TRAFFIC IDENTIFICATION" by Carlos Bacquet in partial fulfillment of the requirements for the degree of Master of Computer Science.

Dated: August 10, 2010

Supervisors:	Professor Nur A. Zincir-Heywood
	Professor Malcolm I. Heywood (co-supervisor)
Reader:	Professor Denis Riordan

Dedicated to

my friends (you know who you are)

and to my mother Neyda who has always been there for me,

gracias mama.

Table of Contents

List of Tables

List of Figures

Abstract

An important part of network management requires the accurate identification and classification of network traffic for decisions regarding bandwidth management, quality of service, and security. Earlier attempts to identify network traffic used to rely on TCP port number inspection and/or on payload inspection. Both these methods do not work on encrypted traffic as the payload is encrypted, and as port number inspection is not considered reliable anymore. This work explores the use of a Multi-Objective Genetic Algorithm (MOGA) for both, feature selection and cluster count optimization, for an unsupervised machine learning technique, K-Means, applied to encrypted traffic identification. In this work, SSH is chosen as an example of an encrypted application. However, nothing prevents the proposed model to work with other types of encrypted traffic, such as SSL or Skype. This work explores whether it is possible to mimic the performance of a gold standard model (classifier type, label driven model), using a MOGA based on clustering objectives. Then the performance of the proposed model is benchmarked against other unsupervised learning techniques existing in the literature: Basic K-Means, semi-supervised K-Means, DBSCAN, and EM. Results show that the proposed MOGA, not only outperforms the other models, but also provides a good trade off in terms of detection rate, false positive rate, and time to built and run the model. This is a very desirable property for a potential implementation of an encrypted traffic identification system. Finally, a hierarchical version of the proposed model is implemented, to observe the gains, if any, obtained by increasing cluster purity by means of a second layer of clusters. Results show that with the Hierarchical MOGA, significant gains are observed in terms of the classification performances of the system.

Acknowledgements

This work was supported by ISSNet - NSERC Strategic Network and the CFI new opportunities program. My thanks to Dr. Nur Zincir-Heywood and to Dr. Malcolm Heywood for all their guidance and time dedicated. My thanks to Chris Maxwell and David Green from the Faculty of Computer Science Tech Support, and also thanks to the Dalhousie UCIS team for providing the anonymozied Dalhousie traffic traces.

All research was conducted at the Dalhousie Faculty of Computer Science NIMS Laboratory, http://www.cs.dal.ca/projectx.

Chapter 1

Introduction

An important part of network management requires the accurate identification and classification of network traffic [5], [3]. Network administrators are normally interested in identifying application types for decisions regarding both bandwidth management and quality of service [3]. Traffic identification can also have potential applications in network security, in particular in the field of forensic analysis. Earlier attempts to identify network traffic used to rely on TCP port number inspection, i.e., inferring an application's protocol based on the port number it uses. However, this approach became increasingly inaccurate as users started to send traffic through non-standard port numbers, and as peer to peer (P2P) applications started to hide behind well known port numbers to avoid detection [29]. An alternative to port number inspection is payload inspection, which consists of inferring the type of the application by searching for protocol specific behavior or data inside the TCP or UDP payloads [39]. However, deep packet inspection cannot be performed when the packets are encrypted, as the payload is obscured. In addition, "governments may impose privacy regulations constraining the ability of third parties to lawfully inspect payloads at all" [39]. Consequently, the two traditional approaches to identify network traffic are unable to deal with the identification of encrypted traffic.

Given the limitations of the aforementioned methods, some attention has been given to the use of machine learning techniques to identify network traffic. These techniques normally employ statistics of the data, like packet length and packet inter arrival time related features. A number of these attempts have employed supervised learning methods, however, these classifiers have uncertain generalization properties when faced with new data. An alternative to classifiers is the use of unsupervised machine learning methods, specifically, clustering mechanisms. Unlike classifiers, clustering algorithms identify the natural classes existing in the data, i.e., a data driven approach. Thus, unsupervised machine learning techniques have the capability of identifying new behaviors in the data. This makes them particularly suitable to traffic analysis, given that new applications constantly populate the Internet without any warning for existing security mechanisms to adapt. The purpose of this work is to investigate the implementation of a genetic algorithm to work with an unsupervised machine learning technique to identify encrypted traffic. A Multi-Objective Genetic Algorithm (MOGA) will be used for the dual goal of (i) identifying the appropriate (flow) attribute/feature subspace and (ii) identifying traffic types via clustering. Such a MOGA based approach for traffic identification was first employed by the author in [8], under the assumption that the resulting clusters partition traffic into encrypted/not encrypted. The proposed MOGA will evolve four predefined clustering objectives that will aim to maximize inter and intra-cluster distance, as well as decrease the number of employed features and clusters. It is believed that higher inter and intra-cluster distance would lead to better quality of clusters, and that lower number of features and clusters will have a positive impact on the computational cost of training and running the model.

Thus, the first hypothesis of this work is whether the MOGA is capable of selecting an appropriate subset of features to partition the data into encrypted versus non encrypted traffic. To test the effectivity of the proposed method as a feature selection technique, the performance of the MOGA will be benchmarked against a gold standard model for feature selection. Such a model will also be a machine learning technique, however, the learning will be guided with classification objectives rather than with clustering objectives (chapter 7).

The second hypothesis of this work is whether it is possible to obtain gains in both, computational performance and classification rates with the proposed model. To test this hypothesis, the performance of the proposed MOGA will be benchmarked against the basic K-Means, but without using the MOGA optimization. Furthermore, the performance of the MOGA will also be benchmarked against other unsupervised learning techniques existing in the literature, such as semi-supervised K-Means, DBSCAN, and EM (chapter 8).

Finally, this work will explore the effects of increasing cluster purity in the classification performance of the system. It is believed that a higher cluster purity will lead to better classification rates. To test this hypothesis, a hierarchical version of

MOGA that will further partition the data will be implemented, aiming to increase cluster purity. Then, the performance of this model will be compared against the performance of the original MOGA (chapter 9).

With regards to the traffic type to analyze, Secure Shell (SSH) was chosen as an example encrypted application. While SSH is typically used to remotely access a computer, it can also be utilized for "tunneling, file transfers and forwarding arbitrary TCP ports over a secure channel between a local and a remote computer" [3]. These properties of SSH make it an interesting encrypted application to focus on, given that it shows similar behavior to popular encrypted applications such as Skype. However, unlike Skype, SSH is an open source protocol. This ensures that the ground truth is known regarding the traffic tested.

In the following, chapter 2, Background Information, will provide a brief description of unsupervised machine learning techniques, as well as a basic description of genetic algorithms. Chapter 2 will also describe the encrypted protocol to identify, SSH. Chapter 3, Previous Work, will describe the work existing in the literature where unsupervised machine learning techniques have been employed to identify network traffic. Special attention will be given to the specific algorithms employed, as well as to the feature selection process. Chapter 4, Algorithm Methodology, will describe the proposed model, going into the details of the implemented genetic algorithm, fitness function, and the feature selection and cluster count optimization process. Chapter 5, Experimental Methodology, will describe the data employed for the training and testing of the model, as well as the data pre-processing (flow generation process). In order to test the proposed methodology, chapter 6, Analysis of Clustering Objectives, will examine the effectivity of the MOGA in evolving the four predefined clustering objectives. Chapter 7, Analysis of the Unsupervised Learning Model, will compare the performance of the proposed MOGA against a gold standard model, to assess the suitability of the clustering objectives to the task of feature selection and encrypted traffic identification (first hypothesis). The work presented in this chapter can also be found in [9]. Chapter 8, MOGA vs Other Unsupervised Learning Techniques, will benchmark the proposed MOGA against four other unsupervised learning techniques existing in the literature (second hypothesis). The work presented in this chapter can also be found in [7]. Chapter 9, Hierarchical MOGA, will expand the proposed model into a hierarchical unsupervised learning algorithm, to analyze the gains, if any, obtained by increasing cluster purity (third hypothesis). Finally, chapter 10, Conclusions, will summarize the presented work, detailing the outcome of the conducted experiments and their significance in terms of the three hypotheses analyzed. Chapter 10 will also outline the future work derived from the observations and conclusions obtained from this thesis work.

Chapter 2

Background Information

2.1 Introduction

This chapter introduces the basic concepts employed in this work, where it is intended to identify encrypted network traffic by means of an unsupervised learning technique (K-Means algorithm), wrapped with a multi-objective genetic algorithm, MOGA, to address the dual problem of feature selection and cluster count optimization. Thus, it becomes necessary, before describing the details of the methodology, to explain the theory behind unsupervised machine learning algorithms, as well as the essence of genetic algorithms, and the encrypted protocol to identify, Secure Shell (SSH). This chapter will specifically describe the unsupervised machine learning technique employed by the proposed model, K-means, as well as two other common unsupervised techniques found in the literature, DBSCAN and EM. This is not a comprehensive description of these topics, the interested reader should refer to the material cited in each section.

2.2 Unsupervised Learning and K-Means Algorithm

Discriminating encrypted versus non-encrypted network traffic can be essentially addressed as a problem of pattern recognition. The objective is to determine to which class an entity belongs, based on its characteristics or features. Accordingly, the first step will be to identify the features that will allow proper distinction between in-class (encrypted traffic) and out-class (non-encrypted traffic). The feature selection process employed in this research is detailed in section 5.3, and again revisited in chapter 7.

In the field of machine learning, the classification between in-class and out-class is normally performed with either supervised learning methods, or with unsupervised learning methods. In the case of supervised learning, every instance in the training data has an associated label or class tag, indicating the class the entity belongs to.

These labels guide the learning of the model. Consequently, the learning phase is constrained to generate models that will only be able to recognize entities belonging to the classes that the classifier was trained with. When faced with new unseen data, classifiers tend to have uncertain classification properties. Unsupervised learning methods, on the other hand, guide the training of the model without any labels or class tags. Because these methods are not constrained by the classes used for training, unsupervised techniques can discover new behaviors in the data, potentially leading to the discovery of new classes [17]. This is particularly interesting for network traffic analysis, where new applications constantly populate the network without any warning for existing classifiers to adapt. Furthermore, not requiring labelled training data is also an advantage in terms of training data generation, as labelled samples tend to be expensive to generate [17]. Again, this is particularly appealing for encrypted network traffic, where the data is normally represented by log files without any payload information to facilitate labeling.

The use of unsupervised learning techniques, specifically clustering approaches, to classify data normally consists of a model building phase, and a test phase. In the model building phase training data is clustered, or partitioned, based on some criteria of similarity. Once the clusters of similar data have been formed, they are associated with a label, which represents the labels of the data grouped in that particular cluster. This cluster labeling can be implemented in a number of ways, but the basic idea is that the label has to represent the data contained in the cluster. The test phase, on the other hand, consists of classifying data by means of the built clusters. This is done by predicting the class of each entity in the test data, based on the label of the cluster that the entity is more similar to. Naturally, this criteria of similarity has to be the same as the one used to build the clusters.

2.2.1 K-Means Algorithm

The unsupervised algorithm utilized in this work is the K-Means clustering algorithm [32]. This algorithm seeks to group the data into K clusters or partitions, based on the distance between the data points. The idea is to minimize the distance between the data points and their closest cluster centroid. “Stated informally, the

[K-Means] procedure consists of simply starting with [K] groups each of which consists of a single random point, and thereafter adding each new point to the group whose mean the new point is nearest. After a point is added to a group, the mean of that group is adjusted in order to take account of the new point. Thus at each stage the [K-Means] are, in fact, the means of the groups they represent (hence the terms [K-Means])" [32]. Whereas this algorithm can at times converge to an optimal solution, that will not be the general case [32]. Algorithm 1 outlines the K-Means pseudo code [17], where *ui* represent the cluster's means.

Algorithm 1 K-Means Algorithm

```
begin initialize n, k, u1, u2, ... uk
    do classify n samples according to nearest ui
        recompute ui
    until no change in ui
    return n, k, u1, u2, ... uk
end
```

The computational complexity of this algorithm is *O(ndkT)*, where d is the number of features and T is the number of iterations.

An immediate concern with the K-Means algorithm is that it is required to know before hand the number of K clusters. Hence, part of this research will be to find the optimum number of clusters, which will directly depend on the employed features or attributes [19]. While a certain number of clusters k might lead to optimum results with a subset of features *f1*, that same value of k might perform very poorly with a different subset of features *f2*. In addition, the number of clusters alone has a great impact on the performance of the algorithm. If the number of clusters is too small, then the algorithm might not be able to tell the differences between classes. If, on the other hand, the number of clusters is too large, then the algorithm will simply memorize the training data, and its performance will be poor in front of new unseen data, i.e., over-fitting [32].

2.2.2 DBSCAN Algorithm

DBSCAN is a density based algorithm, so it regards "clusters as dense areas of objects that are separated by less dense areas" [20]. It was first proposed by Ester *et al.* [24], as an attempt to generate a clustering algorithm that required minimal prior knowledge about the data, as it does not require to know before hand the number of natural clusters in the data. It was also part of the motivation to generate an algorithm, that unlike existing techniques like K-Means, was not limited to find clusters of spherical shape. DBSCAN partitions the data based on areas of higher density of points. Clustered groups have, as expected, a higher density in points than areas outside of the clusters. These dense area clusters can follow any arbitrary shape.

Clusters are built with the premise that for each point p inside a cluster, "the neighborhood of a given radius has to contain at least a minimum number of points, i.e., the density in the neighborhood has to exceed some threshold" [24]. The *Eps-neighbohood* of a point p is defined by all the points q such that the distance between p and q, *dist(p,q)* is less than a specific distance (*Eps* distance). Then, for every point p that is part of a cluster, there has to be a point q such that p is in the q *Eps-neighborhood*, and that the *Eps-neighborhood* of q has al least a certain number of points (*MinPts*). Thus, by enforcing a specific distance, *Eps*, and a minimum number of points, *MinPts*, the algorithm ensures a minimum density threshold inside the resulting clusters.

Formally, a point p is *directly density-reachable* from a point q if it belongs to q's *Eps-neighbohood*, and if the number of points in that *Eps-neighbohood* is larger than *MinPts*. Then a point p is *density-reachable* from a point q if there are points $\{p1, p2, ..., pn\}$, such that *pi* is directly reachable from *pi+1*. A point p is *density-connected* to a point q if there is a point o such that both p and q are *density-reachable* from o. Then, a cluster will be a set of *density-connected* points [24]. The clusters are build in two phases. First an arbitrary point p is identified that satisfies the core point condition. Then, all the point that are *density-reachable* from that point are retrieved, generating a cluster. Points that are not part of any cluster are identified as noise.

The input for the algorithm is only the (*Eps*) distance, and the number of minimum points (*MinPts*). Given that the *Eps-Neighborhoods* are relatively small in comparison to the n size of the data to cluster, the time complexity of creating a neighborhood can be approximated to *O(log n).* Then, because there would always be at most one query per point n, the time complexity of the entire algorithm is *O(n* log n)* [24].

2.2.3 EM Algorithm

The Expectations Maximization (EM) algorithm [16] works with the probabilities of each instance belonging to each cluster [42]. Unlike distance based clustering algorithms, like K-Means, EM works with the statistical models that describe the data to be clustered. The algorithm has two phases, an expectation phase and a maximization phase. The parameters used by the algorithm that model the characteristic probability distribution of each cluster are estimated during the expectation phase, and are continually re-estimated during the maximization phase [20].

The idea behind the algorithm is to identify the data based on its density functions. Specifically, using a probability density function like the mixture model, "which asserts that the data is a combination of K individual component densities, corresponding to the K clusters" [14]. The mixture model seeks to identify the clusters in the data and to "provide a model (density distribution) of each of the populations" [14]. Then, the EM algorithm is used to estimate the parameters for the mixture model, to fit the model to the particular data to cluster. "The EM algorithm iteratively refines an initial cluster model to better fit the data and terminates at a solution which is locally optimal" [14]. The algorithm iterates until a stopping criteria is satisfied. At each iteration, the EM algorithm computes the membership probability of each point in the data set, and then it updates the model parameters accordingly.

With regards to its time complexity, to compute the membership probability of the points in the data, it is necessary to do one full scan of the data for each iteration. The total number of iterations will depend on the initial parameter values and on the distribution of the data. A more detailed explanation of the algorithm can be found in [16] and also in [1].

2.3 Genetic Algorithms

The idea behind genetic algorithms consist of bringing concepts from genetic evolution, as they occur in nature, to solve optimization problems in the engineering sciences [37]. In the words of genetic algorithms creator, John Holland, “Living organisms are consummate problem solvers. They exhibit a versatility that puts the best computer programs to shame” [25]. Genetic algorithms are particularly useful when the space of possible solutions to a problem is too large to be searched efficiently with exhaustive methods.

The basic genetic algorithm framework consists of encoding possible solutions to a problem into strings of ones and zeros, which for the purpose of genetic evolution become chromosomes. Still, from a computer perspective, they are simply encoded solutions for the problem at hand. Several of these possible solutions are then grouped together into a population of chromosomes, which is subsequently genetically evolved. What this means is that employing techniques like crossover and mutation, these chromosomes are combined imitating the interchange of genetic material as it happens in nature when species breed. Specifically, crossover means that at a certain position in the string, the material between two chromosomes is exchanged, generating offspring. Mutation, on the other hand, means that some of the bits in the offspring chromosome are flipped [37]. The power of genetic algorithms comes from the fact that parental selection is proportional to the fitness of the individuals (chromosomes). That is, the fittest individuals are more likely to generate offspring than the least fit ones. It will be critical, therefore, to define a fitness function that allows discrimination between the individuals that better solve the problem at hand (fittest), and the individuals that don’t (less fit). Algorithm 2, which has been taken and summarized from [37], outlines a simple genetic algorithm framework.

One iteration of this process is called a generation, and a set of generations is called a run. A run can consist of anywhere from 50 to 500 or more generations [37]. Given the stochastic nature of the process, it is normally necessary to combine several runs to obtain consistent results. At the end of a run, the population will ideally contain fitter individuals, thus, better solutions to the original problem. The key concept behind genetic algorithms is the idea of building blocks, or schemes. That is, better solutions are achieved by combining other potentially good solutions, or solutions

Algorithm 2 Simple Genetic Algorithm

```
1  Start with a randomly generated population of $n$ $l$-bit chromosomes (candidate solutions to a problem)
2  Calculate the fitness $f(x)$ of each chromosome $x$ in the population
3  Repeat the following steps until $n$ offspring have been created:
     a  Select a pair of parent chromosomes from the current population, the probability of selection is proportional to the fitness of the individual.
     b  With probability $pc$, "crossover probability", crossover the pair at a randomly chosen point to form two offspring.
     c  Mutate the two offspring at each bit (locus) with probability $pm$, "mutation probability", and place the resulting chromosomes in the new population.
4  Replace the current population with the new population.
5  Go to step 2
```

that already contain good schemes, or building blocks. As these individuals with good schemes are combined, they could eventually lead to better solutions.

2.3.1 Multi-Objective Genetic Algorithms

A particularly interesting type of genetic algorithms are the multi-objective optimization algorithms, where the purpose is to address problems that target more than one objective. The idea is to consider the multiple objectives simultaneously, searching for a solution space that is optimum in terms of all the objectives, compromising among its values. This leads to a set of potential solutions to the problem at hand, which will be a subset of the Pareto Front. The Pareto Front is conformed by the set of non-dominated individuals. An individual dominates another if it has a higher value in at least one of the objectives, and is at least as good in all the others. Because the solution will consist of a set of possible solutions (members of the Pareto Front), often a decision maker entity is needed to select a final solution. Multi-objective evolutionary algorithms need to ensure that the evolution is guided towards Pareto optimality, while at the same time maintaining diversity among the solutions, and preventing the loss of non-dominated solutions [36]. The most representative multi-objective algorithms are:

Strength Pareto Evolutionary Algorithm (SPEA), is a generational based genetic algorithm that maintains a solution archive with the non-dominated individuals. The idea is to avoid the elimination of the non-dominated individuals. After each generation, the non-dominated individuals are added to this archive, "culling dominated and indifferent members accordingly" [36].

Nondominated Sorting Genetic Algorithm (NSGA), implements a "non-dominated ranking scheme as a means of influencing fitness assignment" [36]. In its original version the fitness is assigned based on the ranks, favoring the non-dominated individuals. In its second version, this was improved by including "diversity maintenance schemes" [36].

Multi-Objective Genetic Algorithm (MOGA), where every member in the population is assigned with a rank, which indicates how many individuals in the population dominate that particular individual. The best individuals, the non-dominated ones, will be those whose rank equals one and will receive the highest fitness. The remaining individuals receive fitness as a "linear function of rank, scaled by objective space density in the form of niche counts according to sharing distance" [36].

Pareto Converging Genetic Algorithm (PCGA), which assigns ranks similarly as done by the MOGA. The PCGA, however, introduces elitism with a steady state algorithm. Thus, the model does not "employ a diversity mechanism explicitly, arguing that the compressed range of ranks due to the elitist steady-state strategy of PCGA produces the same effects as fitness sharing" [36].

2.4 The SSH Protocol

This section describes the encrypted protocol that is going to be identified by means of an unsupervised machine learning technique, K-Means, wrapped with a MOGA for feature selection and cluster count optimization.

When data travels through the Internet in plain text, it can be easily intercepted, captured, and viewed by third parties [10]. Secure Shell (SSH) is a protocol that allows users to send messages encrypted through the network with the purpose of avoiding interception. The data is encrypted with an RSA key exchange, allowing secure communication over unsecured channels [2]. SSH enables users to log into a

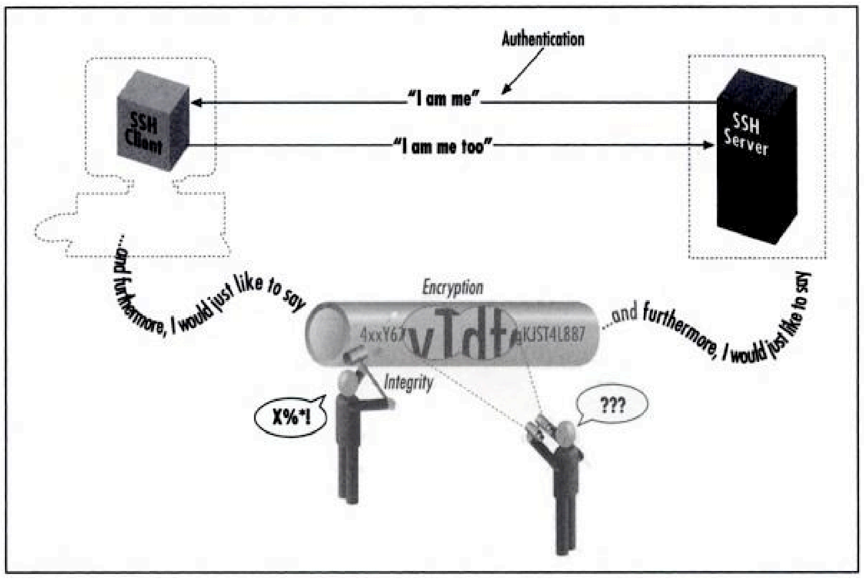

Figure 2.1: SSH Tunnel

computer remotely through the network, executing commands, moving files, etc [2]. It also allows the integration of security to other existing insecure protocols, such as Telnet, FTP, among others [10]. One of the main advantages of using SSH is that it is practically imperceptible for the user, and that it can operate with most operating systems [10]. The data encryption happens before any message left a computer, and the data is automatically decrypted when it reaches its destination, before the receiver gets to see the message. Thus, the notion of tunneling, where users can enjoy a secure interchange of information over the Internet without being aware of this encryption process, as it all happens behind the scenes. Figure 2.1, taken from [10], displays the concept of an SSH tunnel.

As it can be observed from figure 2.1, the protocol is organized around a client-server architecture. The client requests connections to the server, which accepts or rejects the logging, and then evaluates any other request like files request, command execution requests, etc [10]. In order to set up the encrypted channel, an SSH authentication process precedes the data exchange. At a high level, the SSH authentication, or SSH handshake, consists of a host authentication, and a user authentication (figure 2.2 taken from [18]).

Finally, it is important to note that SSH can not only be used to hide legitimate users information from third parties on the network. SSH can also be used by banned

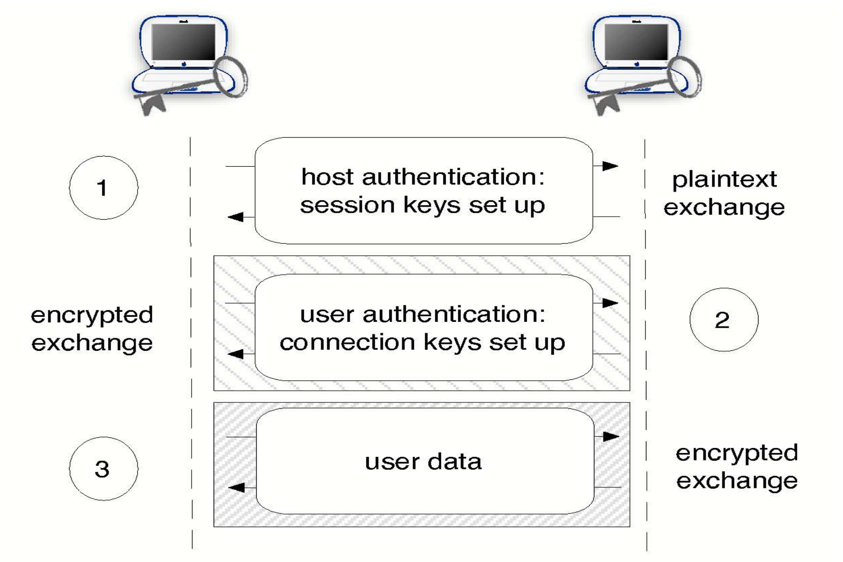
1
host authentication: session keys set up
plaintext exchange
encrypted exchange
user authentication: connection keys set up
2
3
user data
encrypted exchange

Figure 2.2: SSH Handshake

applications to circumvent firewalls and security mechanisms. Applications like peer to peer, or any other type of blocked or controlled application, can bypass firewalls by encrypting their content under SSH tunnels [18]. With the content obscured, a firewall has little chance to discriminate between legitimate and banned or malicious traffic, likely letting the encrypted content going through. Going any deeper into this argument would be beyond the scope of this work. Still, the point has been to bring attention to the importance of identifying encrypted traffic of any kind, in this case, SSH.

2.5 Conclusions

This chapter has briefly reviewed the basic knowledge needed to understand the work presented in this thesis. In summary, the unsupervised learning method, K-Means, will be optimized by means of a genetic algorithm to solve the dual problem of feature selection and cluster count optimization. This model will then be employed to identify encrypted network traffic, specifically, SSH. The following section will review the previous work existing in the literature, where similar unsupervised learning techniques have been employed for network traffic classification, including K-Means, DBSCAN, and EM.

Chapter 3

Previous Work with Unsupervised Learning Techniques applied to Network Traffic Identification

3.1 Introduction

Network traffic identification consists of separating the data going through the network by protocols or application types. Earlier attempts to identify network traffic used to rely on TCP port number inspection, i.e., inferring an application's protocol based on the port number used to communicate it. The use of standard port numbers allowed such association. For instance, if an application used port number 80, then it could be confidently assumed that the used protocol was HTTP [26]. However, this approach became increasingly inaccurate as users started to send traffic over non-standard port numbers, and as peer to peer (P2P) applications started to hide behind well known port numbers to avoid detection [29]. An alternative to port number inspection is payload inspection, which consists of inferring the application types by searching for protocol specific behavior or data inside the TCP or UDP payloads [39]. Thus, it is necessary to know before hand the mapping between the application behaviors (syntax) and the applications to identify [39]. These methods can be extremely accurate when the payload can be accessed [4]. However, deep packet inspection cannot be performed when the packets are encrypted, as the payload is obscured. In addition, "governments may impose privacy regulations constraining the ability of third parties to lawfully inspect payloads at all" [39]. Given these limitations, several previous attempts to identify encrypted traffic have worked with statistics of the data, like packet length and packet inter arrival time related features. A number of these attempts have employed supervised machine learning techniques, however, these classifiers have uncertain generalization properties when faced with new data. An alternative to classifiers is the use of unsupervised machine learning techniques, specifically, clustering mechanisms (section 2.2). This chapter presents a summary of the previous attempts existing in the literature, where unsupervised machine learning techniques have been employed to identify network traffic.

3.2 Previous Work

One of the earliest works in the literature in this field was presented by Paxson in [41]. In this work, a number of analytic models were presented to describe the features associated with TELNET, NNTP, SMTP, and FTP connections. Three million TCP connections gathered at seven different sites were analyzed, focusing on attributes like bytes transfered and duration. In doing so, it was noticed that in the case of attributes with large ranges of values, it was more meaningful to analyze the data after applying a logarithmic transformation to it. Specifically, the significance of statistics such as mean and standard deviation were skewed towards attributes with the greater ranges. Given that there were several orders of magnitude between these values, applying a log transformation reduced the corresponding dynamic range to the attributes with the most dissimilar ranges. This logarithmic normalization can be observed in several subsequent traffic classification approaches existing in the literature, as a measure to obtain better data representation. Section 7.3, chapter 7, will evaluate the effects of applying this logarithmic transformation to the data, prior to applying the proposed model.

McGregor *et al.* [35] presented an unsupervised approach using Expectation Maximization (EM) clustering algorithm, to classify network traffic represented by a set of flow attributes. The authors first examined plots of packet size against packet inter-arrival time, where they observed that these plots exhibited a number of characteristic shapes that were believed to indicate application types like, HTTP, FTP, and SMTP. After this analysis, flows were generated with the following attributes: packet size statistics, inter-arrival statistics, byte counts, connection duration, number of transitions between transaction mode and bulk transfer mode, and the time spent idle, in bulk transfer, and in transaction mode. Attributes that did not have an impact on the classification were identified and discarded. Using the Auckland VI trace they observed that the clustering algorithm showed some capabilities in grouping flows together by traffic type, but that more work needed to be done to derive better features to increase performance. This characterization of applications based on flow attributes alone, served as a promising indication that a purely data driven flow based approach could perform a good degree of traffic classification.

Zander *et al.* [45] also proposed the use of an unsupervised machine learning technique (autoclass) for network traffic identification. The authors employed the Auckland VI data set, and the NZIX-II and Leiozig traces. The flow feature selection process was based on a sequential forward selection, where the algorithm starts with the single attribute that generates the best results. Then, attributes were added one at a time, based on the results they generated in combination with the already selected attributes. This process was repeated until no more improvements were achieved. The quality of the clusters was evaluated in terms of the intra-class homogeneity. The idea was to maximize this variable to generate a good cluster separation, under the assumption that this would lead to a better application differentiation. The authors were able to achieve an average accuracy across all traces of 86.5%. It was identified that together with good cluster separation, it would also be desirable to minimize "the number of classes to improve speed of the learning and classification" and minimize "memory requirements". To this end, the model proposed in this thesis addresses this issue by having one of the objectives to be evolved minimize the number of employed clusters (section 4.3). The authors also mention that better, new flow attributes, among others idle time, would be considered for future experiments. This issue has also been considered in the proposed model by including in this thesis several idle time related features (section 5.3). Finally, the authors left to "quantify the performance in terms of processing time and memory consumption and to investigate the trade-off between the approach's accuracy and processing overhead" for their future work. To that end, the proposed model has been benchmarked against other unsupervised approaches existing in the literature, not only in terms of classification accuracy, but also in terms of processing time (chapter 8).

Bernaille *et al.* [12] proposed the use of an unsupervised (K-Means) online approach, based solely on the packet size of the first p packets. The main motivation behind this work was that "state-of-the-art techniques cannot determine the application before the end of the TCP flow" [12]. Whereas the idea here was to achieve online classification. Their approach aimed at identifying the application associated with a TCP flow as early as possible, using the size of the first data packets, ignoring TCP control packets (SYN, ACK's, etc). The premise was that the "size of the first few packets is a good predictor of the application associated with a flow because it captures the application's negotiation phase, which is usually a pre-defined sequence of messages and distinct among applications" [12]. The authors observed that unsupervised learning is more appropriate for traffic classification than supervised learning, because it does not rely on predefined classes. This argument was further investigated by presenting an analysis of the multiple behaviors an application can have. Specifically, they plot the different behaviors of the FTP protocol: control flows, download flows, and upload flows. They found many other multi-modal applications in their data, which again, should be easier to model with unsupervised methods. With their proposed model, the best results were obtained employing the first 5 packets, with 50 clusters for the K-Means algorithm. The proposed model correctly classified more than 80% of the flows of almost all of the tested applications. The authors did mention a few limitations of the proposed method, among others, the fact that the packets might arrive out of order or may appear more than once. This would directly affect the results, however, in their studied network this only happened with less than 4% of the TCP flows. They also mentioned that two applications might start with the same packet sizes, which would also affect the results. To solve this they would look into heuristics like port number and inter-arrival times. To that end, in the work proposed in this thesis the use of port numbers has been avoided as it can be misleading [29].

In [13], Bernaille *et al.* further elaborated their approach, working on traces collected at eight different networks. In this work, they explored their proposed approach with K-Means and Gaussian Mixture Models, GMM, on an Euclidean space, and Spectral clustering on Hidden Markov Models, HMM. Their results showed that with the first four packets of a TCP connection they could obtain an accuracy above 90%. Their classification phase used the clusters defined in the training phase, plus heuristics relative to the port number. A 4-dimensional space gave the best results for the three clustering algorithms, employing 40 clusters for the K-Means algorithm, 30 clusters for the HMM, and 45 clusters for the GMM. Their results also showed that "even though the HMM representation is richer, the quality of the clustering is comparable to the simpler Euclidean representation when using GMM clustering" [13].

Their model correctly classified over 98% of known applications in the studied payload traces when employing the GMM clustering combined with TCP port numbers. Again, for the work proposed in this thesis, any heuristics related to port numbers is avoided. Finally, in [11], Bernaille *et al.* extended their work to identify applications encrypted in SSL connections. Their method used the size of the first few packets of an SSL connection to recognize applications. The method was tested on two campus traces, and on a manually-encrypted trace, being able to recognize the applications in SSL connections with more than 85% accuracy. To separate the SSL traffic from the rest of the trace they used their GMM model, employing the first three packets and 35 clusters. The protocol was then identified by analyzing the subsequent first application packets. It is interesting to notice that online traffic classification, or near real-time traffic classification, is not necessarily impossible to achieve with models that employ complete flows, as opposed to only the first few packets as proposed here. Near real-time traffic classification could potentially still be achieved with the entire flows, and without the need of port number related heuristics. However, an online approach would imply the availability of the computational power to do so.

Erman *et al.* presented several unsupervised approaches for traffic classification [20, 21, 22, 23]. In [21] the authors compared an unsupervised approach using an Expectation Maximization (EM) based clustering algorithm (AutoClass), against a supervised approach that used a Naive Bayes Classifier. Both methods were tested on subsets of the traces Auckland IV and Auckland VI from the University of Auckland. The data was represented by the following flow features: Total Number of Packets, Mean Packet Size, Mean Data Packet Size, Flow Duration, and Mean Inter-Arrival Time of Packets. Because of the heavy tail distributions on many of these features, they used the logarithms of the features. Their results showed that the unsupervised technique had an accuracy of up to 91%, outperforming the supervised technique by up to 9%. Furthermore, they found that the unsupervised technique was also able to discover traffic from previously unknown applications. Then in [20], they presented an evaluation of three clustering algorithms: K-Means, which is a partition based algorithm, DBSCAN, which is a density based algorithm, and the previously used AutoClass. K-Means and DBSCAN were chosen in particular because of their superior clustering speed in comparison to AutoClass. For this work the authors used the Auckland IV data set, as well as the locally collected Calgary trace. The data was represented by the following flow features: total number of packets, mean packet size, mean payload size, number of bytes transferred, and mean inter-arrival time of packets. Like in their previous work, they applied a logarithmic transformation to the data. The authors found that with K-Means the overall accuracy steadily improved as the number of clusters was increased. This continued until K was around 100 with the overall accuracy being 79% and 84% on each data set respectively. In comparison to the other tested algorithms, K-Means had better accuracy than DBSCAN. Autoclass slightly outperformed K-Means, but K-Means had a much faster building time. Thus, K-Means seemed to provide the best mix of properties.

Finally, in [22], Erman *et al.* proposed a semi-supervised method, in which the training was done with only a small percentage of labeled and a high percentage of unlabeled flows. They obtained high flow and byte accuracy, greater than 90%, clustering the data with a K value of 400 for the K-Means algorithm. Using a backward greedy feature selection method, they chose the following eleven features: total number of packets, average packet size, total bytes, total header (transport plus network layer) bytes, number of caller to callee packets, total caller to callee bytes, total caller to callee payload bytes, total caller to callee header bytes, number of callee to caller packets, total callee to caller payload bytes, and total callee to caller header bytes. They observed that flow features that have time components should be avoided as they are more prone to suffer variations across different networks. With regards to these approaches, in chapter 8 the performance of the model proposed in this thesis is benchmarked against the three clustering algorithms evaluated in [20] (K-Means, DBSCAN, and EM), and also against the semi-supervised approach presented in [22].

Siqueira *et al.* presented an unsupervised approach in [28], focusing on P2P traffic. They used a hierarchical clustering technique, in which 249 features were analyzed, including packet length and packet inter-arrival time statistics. The feature selection was based on the Ratio F, which uses a ratio of two estimates, "dividing the variance mean of intra-group elements" and "the mean variance of inter-group elements". From original 249 features, the best five discriminators were: Port Server, Maximum Window Advertisement Client to Server, Maximum Window Advertisement Server to Client, Minimum Window Advertisement Server to Client, Minimum segment size

Client to Server. They were able to achieve 86.12% in trust, and 96.79% in accuracy. It is important to notice, however, that the Port server was included as one of the selected features, which as already explained, was avoided during the research presented in this thesis.

Yingqiu *et al.* also presented a flow based clustering approach, using K-Means to build the clusters with features previously identified as the best discriminators [44]. The data used was collected at a research facility, and for the feature selection process they used several techniques, like: CFS, Consistency-based subset evaluation, Information gain attribute evaluation for feature selection, backward and forward greedy search, greedy Best First, and Ranker for searching. The premise was that the "more frequent a discriminator appears in the selected feature subsets, the better at discriminating classes". The best features were: "the number of total packets-b-a, the number of actual data bytes-b-a, the number of pushed data pkts-a-b, the number of the pushed data pkts-b-a, size of the mean IP packets-a-b, size of the max IP packets-a-b, variant of the IP packet size-a-b, size of the mean IP packet-b-a, size of the max-IP packet-b-a, variant of the IP packet size-b-a, and duration, where 'a' is the client and 'b' is the server". For the K number of clusters, they used values from 20 to 200, and they applied a log transformation that generated a 10% increase in accuracy, reaching an overall accuracy level of up to 90% when utilizing $K = 80$ clusters. This gave them a good trade off since they observed that greater values of K would take more time to form a model, and it would also have a greater risk of over-fitting. The authors concluded that the logarithmic transformation was very useful for traffic classification. With that regard, section 7.3 evaluates the effectiveness of applying a logarithmic transformation to the model proposed in this thesis.

Yang *et al.* [43] employed a DBSCAN clustering algorithm for traffic classification. The authors employed a wrapper methodology for feature selection, i.e., features were selected based on the effect that they had on the classification accuracy. The five features employed were: mean inter arrival time, mean IP packet size, total number of packets, total number of unique bytes sent, and connection duration. They were able to achieve an accuracy of about 87%, concluding that this algorithm not only had a good classification performance, but that it also had advantages in terms of the reduced time required to build the model, and in terms of the little domain knowledge that was required to set the input parameters (*minPts* and *eps*). Furthermore, they also noticed that this algorithm was capable of building clusters of arbitrary shapes, which resulted in better clusters. To this end, chapter 8 will benchmark the performance of the DBSCAN algorithm against the proposed model in terms of both, classification performance and computational time required.

Maiolini *et al* [33] presented an online K-Means based classifier. The authors analyzed statistical features of the first packets of each connection, such as arrival times, directions and packet sizes. The data was preprocessed by removing from each flow the packets related to the first two packets of the three way TCP handshake. Furthermore, the data was normalized with an "affine" normalization, consisting of taking the difference between the value x and the minimum value of x, divided by the difference between the maximum and minimum values of x. The K number of clusters was selected by means of an *a priori* cross validation performed in the training data. The best results were obtained analyzing the first six packets of each flow, supported by 23 clusters. They focused on identifying applications like HTTP, FTP, POP3, and SSH; and also on identifying applications within the SSH tunnels. Using locally captured and locally generated traffic they were able to achieve an SSH average accuracy of 98.4%. The authors also tried the CAIDA trace, yet, they concluded that "more traces would help assessing the robustness of [the] methodology".

3.3 Conclusions

This chapter has presented a review of the most relevant work existing in the literature related to unsupervised learning algorithms applied to network traffic classification. The employed unsupervised learning algorithms have been emphasized, as well as the feature selection process. From the feature selection perspective, it can be observed that while some authors favored packet length related features, others have resorted into both, packet length as well as inter-arrival time related features. To that end, that decision has been left in the hands of the algorithm, which selects the most appropriate feature set out of a poll of both packet length as well as inter-arrival time related features (section 5.3). In addition, several authors made use of logarithmic transformations to obtain better data representations. Chapter 7 evaluates the effects of applying a logarithmic transformation to the proposed model. With regards to the different algorithms employed, it can be observed that most authors used variations of either K-Means algorithm, DBSCAN algorithm, or EM algorithm. The performance of these techniques is benchmarked against the proposed model in chapter 8 in terms of both classification performances as well as computational times required to build and test the models. Finally, to the best of the author's knowledge the proposed model in this thesis is the first work that performs encrypted traffic identification by means of a multi-objective genetic algorithm without using any payload information or port numbers or IP addresses.

Chapter 4

Algorithm Methodology

4.1 Introduction

This chapter introduces the algorithm implemented during this thesis work, in which a genetic algorithm is utilized for feature selection and cluster count optimization for an unsupervised machine learning technique. Details about the specific genetic algorithm to be employed are reviewed, making emphasis on the individual representation as well as the fitness function. No details are provided regarding the particular data or application where this algorithm will be employed, as those are left for chapter 5, Experimental Methodology.

4.2 Genetic Algorithm for Feature Selection and Clustering

The work proposed in this thesis consists of implementing a Multi-Objective Genetic Algorithm (MOGA) for the dual problem of feature selection and cluster count optimization, applied to encrypted traffic identification. Like most Genetic Algorithms (GA), MOGA starts with a population of individuals (potential solutions to a problem), and incrementally evolves that population into better individuals, as established by the fitness criteria. Fitness is naturally relative to the population. This work followed the MOGA framework proposed by Kim *et al.* [30], but modifying the evolutionary component to follow the Pareto Converging Genetic Algorithm proposed by Kumar *et al.* [31]. The latter converges towards the Pareto-front (set of non-dominated solutions) without any complex sharing/niching mechanism (section 2.3.1). One specific property of this GA is the utility of a steady-state GA, thus, only two members of the population are replaced at a time under an elitist replacement model. What this means is that instead of replacing the entire population after each generation (section 2.3), every time two individuals are combined (bred), the resulting two offspring replace the two worse performing individuals in the population.

Each of these combination and replacement cycles is denominated an epoch.

Two important issues that need to be addressed when using GAs are individual representation, i.e., how will the individuals in the population represent solutions for the problem at hand, and the fitness function, i.e., how will the fitter individuals (the ones that better solve the problem at hand), be distinguished from the less fit individuals (those that perform worse at solving the problem at hand). The fitness function will be explained in section 4.3, and further explored in chapter 7. With regards to the individual representation, on the other hand, each individual in the population will represent a subset of features *f*, which will take care of the feature selection, and a number of clusters K, which will take care of the cluster count optimization. Specifically, an individual is an l bit binary string, where bits between the first bit and the D'th bit represent the features to include, and the remaining bits represent the K number of clusters. The variable D corresponds to the dimension of the entire feature subspace that is being reduced. Bits of the individuals in the initial population are initialized with a uniform probability distribution. For feature selection, a "one" implies that the feature at that index is included (from an array or table containing the features from 1 to D), and a "zero" ignores the feature. The K number of clusters, on the other hand, is obtained by counting the number of "ones" (as opposed to "zeros") contained between the D'th bit and the end of the individual. Clusters are identified using the standard K-Means algorithm, using the subset of features *f*, and the number of clusters K, as the input for the K-Means algorithm. Thus, each individual *ind_i* proposes to cluster the data using a certain subset of features *fi*, and with a certain number of clusters *ki*. Figure 4.1 displays an example of an individual and its feature selection and cluster count representation.

The fitness of the individual will depend on how well the resulting clusters perform in relation to four predefined clustering objectives: *Fwithin*, *Fbetween*, *Fclusters*, and *Fcomplexity* (section 4.3). Fitness evaluation assumes a multi-objective approach, typically resulting in the identification of the Pareto-front, or a set of non-dominated solutions. Informally, a solution is said to dominate another if it has higher values in at least one of the objectives, and is at least as good in all the others. After the objective values for each individual have been assessed, individuals are assigned with ranks, which indicate how many individuals dominate that particular individual.

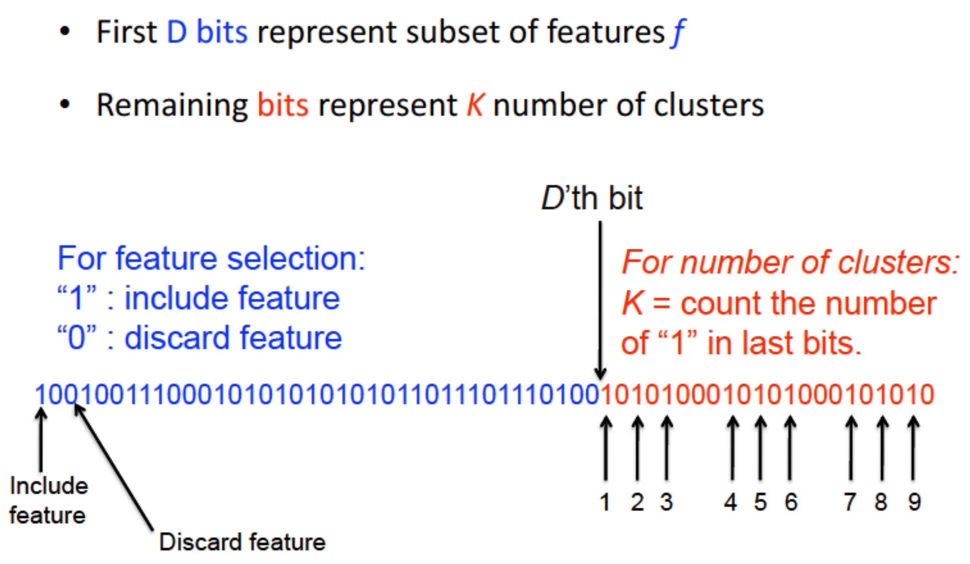

Figure 4.1: Individual Representation

Thus, fitness of the individuals is inversely proportional to their ranks, which is used to build a roulette wheel that is ultimately used for parental selection.

The population is evolved for a certain number of epochs, after which the set of non-dominated individuals (individuals whose ranks equal to 1) is identified. These individuals correspond to the set of potential solutions. The evolutionary component of the algorithm is then terminated and the best individual in the set of non-dominated solutions (the one that better identifies SSH traffic) is identified in the post-training phase. For each individual in the set of non-dominated solutions, *ind_i*, K-Means is applied with its proposed set of features *f_i* and number of clusters *k_i*. The resulting clusters are labelled as either SSH, or non-SSH. If the majority of the flows in a cluster have SSH labels, then that cluster is labeled as SSH, otherwise it is labeled as non-SSH. The post-training phase is then entered and consists of testing each of the non-dominated individual's proposed solutions in the training data (used to build the clusters on), to identify the solution with best classification rates. The system diagram is displayed in Figure 4.2, and its evolutionary component in Figure 4.3.

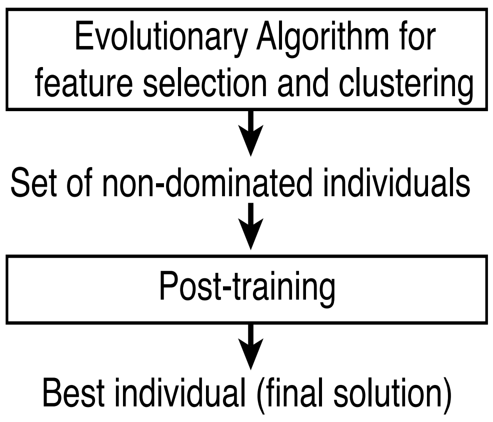
Evolutionary Algorithm for feature selection and clustering
Set of non-dominated individuals
Post-training
Best individual (final solution)

Figure 4.2: System Diagram

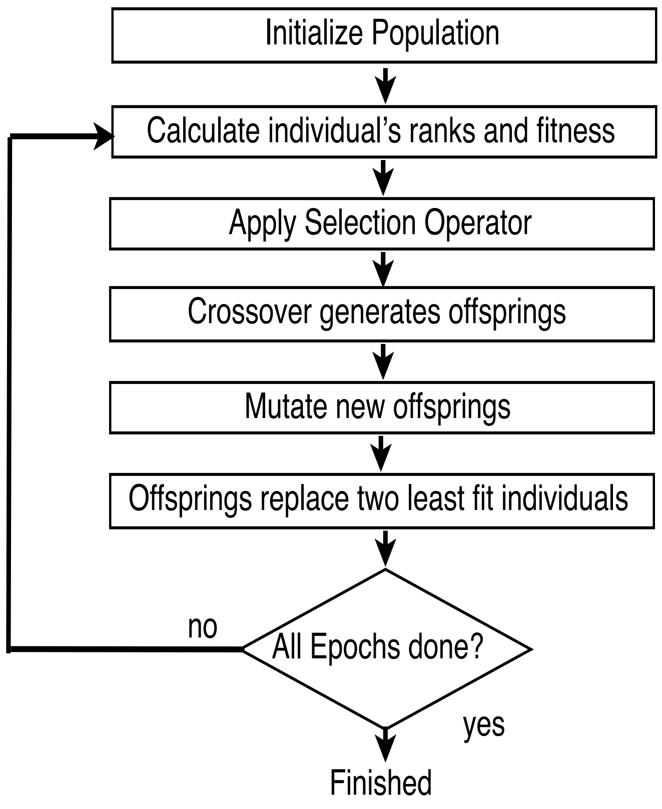
Initialize Population
Calculate individual's ranks and fitness
Apply Selection Operator
Crossover generates offsprings
Mutate new offsprings
Offsprings replace two least fit individuals
All Epochs done?
no
yes
Finished

Figure 4.3: Evolutionary Component Diagram

4.3 Fitness Function

The fitness of the individual will depend on how well the resulting clusters perform in relation to the following four predefined clustering objectives:

1. *Fwithin*: Measures cluster cohesiveness, the more cohesive the better. For this purpose the average standard deviation per cluster is assumed. That is, the sum of the standard deviations per feature over the total number of employed features. Then *Fwithin* will be the number of clusters in a solution, *K*, over the sum of all the clusters' average standards.

2. *Fbetween*: Measures how separate the clusters are from each other, the more separated the better. For each pair of clusters i and j, their average standard deviations are calculated, and the euclidean distance between their centroids is also calculated. Then, *Fbetween* for clusters i and j is:

$$Fbetween(i,j) = \frac{EuclideanDistanceFrom\_i\_to\_j}{\sqrt{(AveStdDev_i)^2 + (AveStdDev_j)^2}}$$

   Thus, *Fbetween* will be the sum of all pairs of cluster's *Fbetween(i,j)*, over *K*.

3. *Fclusters*: Measures the number of clusters *K*, "Other things being equal, fewer clusters make the model more understandable and avoid possible over fitting" [30].

$$Fclusters = 1 - \frac{K - Kmin}{Kmax - Kmin}$$

   Kmax and *Kmin* are the maximum and minimum number of clusters.

4. *Fcomplexity*: Measures the amount of features used to cluster the data, this objective aims at minimizing the number of selected features.

$$Fcomplexity = 1 - \frac{d-1}{D-1}$$

D is the dimensionality of the whole data set and d is the number of employed features.

In short, this model assumes that building fewer high quality clusters in terms of low intra-cluster distance and high inter-cluster distance, and selecting fewer features, will lead to a better data description. Conversely, post training performance evaluation is based on detection rate (DR) and false positives rate (FPR) defined by:

$$DR = 1 - \frac{\#false\_negatives}{total\_number\_of\_SSH\_flows}$$

$$FPR = \frac{\#false\_positives}{total\_number\_of\_non\_SSH\_flows}$$

where false_negatives means SSH traffic incorrectly classified as non-SSH traffic, and false positives means non-SSH traffic incorrectly classified as SSH traffic.

4.4 Parameters Setting

For all the experiments in this thesis, the K-Means implementation provided by Weka [42] is used. Also, the criteria of similarity employed for the K-Means algorithm was the Euclidean distance, defined by:

$$d(p,q) = \sqrt{\sum_{i=0}^{n}(pi - qi)^2}$$

With regards to the parameters for the MOGA, the initial population was set to 250 individuals, and it was evolved for 5000 epochs, with a mutation rate of 0.6% and a uniform crossover operator. These parameters were selected after several trial and error preliminary experiments. Chapter 6, Analysis of Clustering Objectives, will explore the effects of the 5000 MOGA epochs over the clustering objectives.

4.5 Conclusions

This chapter summarized the multi-objective genetic algorithm developed during this thesis work, where four clustering objectives are evolved aiming to approach a Pareto-front of non-dominated solutions. A post training phase selects, based on classification metrics (detection rate and false positive rates), the individual with the best classification performance, which becomes the final solution. Following, chapter 5, Experimental Methodology, will go into the application specific details of the proposed model.

Chapter 5

Experimental Methodology

5.1 Introduction

This chapter describes the data set and data pre-processing required to apply the algorithm described in chapter 4, to complete the proposed encrypted traffic identification model. First the data set is described, consisting of traces of network traffic, and then the flow generation process is outlined, needed to obtain a flow based representation of the data.

5.2 Data Sets

The training data set employed was sub-sampled from the network trace captured by the Dalhousie University Computing and Information Services Centre (UCIS) in January 2007 on the campus network between the university and the commercial Internet. Dalhousie University is one of the largest universities in the Atlantic region of Canada, with more than 15,000 students. Data privacy related issues required that the data was filtered to scramble the IP addresses and that each packet was further truncated to the end of the IP header so that all the payload was excluded. Furthermore, the checksums were set to zero since they could conceivably leak information from short packets. However, any length information in the packet was left intact. Dalhousie traces were labeled by UCIS with a commercial classification tool, PacketShaper, which is a deep packet analyzer, i.e., it analyzes the packet payload [40]. Given that the handshake part of SSH protocol is not encrypted, it can be confidently assumed that the labeling of the data set is completely accurate and provides the ground truth for testing purposes. Again, it is emphasized that this work did not consider any information from the handshake phase nor any part of the payload, IP addresses, or port numbers. Also, this work focuses on SSH as a case study, however, there is nothing in the approach that ties the proposed model to the

SSH protocol specifically. On the other hand, the fact that the SSH's handshake is not encrypted, allowed the comparison of the obtained results with those obtained through payload inspection. In order to build training data the Dalhousie trace was randomly sampled. The training data for all experiments consisted of 12250 flows, including SSH, MSN, HTTP, FTP, and DNS. The test data, on the other hand, was the entire data set (more than 18,500,000 flows), and consisted of flows from each of those applications, plus flows that belonged to any of the following additional applications: RMCP, Oracle SQL*NET, NPP, POP3, NETBIOS Name Service, IMAP, SNMP, LDAP, NCP, RTSP, IMAPS and POP3S.

In addition, in order to measure robustness across different networks, the performance of the system was also evaluated on the AMP, the DARPA [15], and the MAWI [34] traces (chapter 9). The AMP trace, consists of data collected during 2005 at the AMPATH location at the NAP of the Americas in Miami, and containing 21,097,425 flows. The MAWI [34] data set, on the other hand, contains 104,046,846 flows. The DARPA [15] data set contains traces with simulated attacks. For the purposes of this work, only the traces that were free of attacks (weeks one and three) were considered, as the interest is not in intrusion detection at this point, but on encrypted traffic identification [5].

5.3 Flow Generation

Flows are defined by sequences of packets that present the same values for source IP address, destination IP address, source port, destination port and type of protocol. In this work, each flow is described by a set of statistical features and associated feature values. A feature is a descriptive statistic that can be calculated from one or more packets. Thus, a flow can be thought of as a D'th dimensional vector, where D correspond of the total number of calculated features, and where each feature value in the vector represents a different statistic about the packets collected by that flow. NetMate [38], an open source tool, was used to generate flows, and to compute feature values. Table 5.1 shows the 38 features obtained from NetMate. Flows are bidirectional with the first packet determining the forward direction. Since flows are of limited duration, in this work UDP flows are terminated by a flow timeout, and TCP flows are terminated upon proper connection tear-down or by a flow timeout, whichever occurs first. A 600 second flow timeout value was employed here; where this corresponds to the IETF Realtime Traffic Flow Measurement working groups architecture [27]. It is important to mention that only UDP and TCP flows are considered. Specifically, flows that have no less than one packet in each direction, and transport no less than one byte of payload. Again, payload data and features like IP addresses and source/destination port numbers were excluded from the feature set to ensure that the results were not dependent on such biases.

Table 5.1: Flow Features Available for MOGA

ind.	Feature Name	Abbreviation
1	protocol (tcp, udp)	*proto*
2	total forward packets	*total_fpackets*
3	total forward volume	*total_fvolume*
4	total backward packets	*total_bpackets*
5	total backward volume	*total_bvolume*
6	min forward packet length	*min_fpktl*
7	mean forward packet length	*mean_fpktl*
8	max forward packet length	*max_fpktl*
9	std dev forward packet length	*std_fpktl*
10	min backward packet length	*min_bpktl*
11	mean backward packet length	*mean_bpktl*
12	max backward packet length	*max_bpktl*
13	std dev backward packet length	*std_bpktl*
14	min forward inter arrival time	*min_fiat*
15	mean forward inter arrival time	*mean_fiat*
16	max forward inter arrival time	*max_fiat*
17	std dev forward inter arrival time	*std_fiat*
18	min backward inter arrival time	*min_biat*
19	mean backward inter arrival time	*mean_biat*
20	max backward inter arrival time	*max_biat*
21	std dev backward inter arrival time	*std_biat*
22	duration of the flow	*duration*
23	min active	*min_active*
24	mean active	*mean_active*
25	max active	*max_active*
26	std dev active	*std_active*
27	min idle	*min_idle*
28	mean idle	*mean_idle*
29	max idle	*max_idle*
30	std dev idle	*std_idle*
31	sub flow forward packets	*sflow_fpackets*
32	sub flow forward bytes	*sflow_fbytes*
33	sub flow backward packets	*sflow_bpackets*
34	sub flow backward bytes	*sflow_bbytes*
35	forward push counter	*fpsh_cnt*
36	backward push counter	*bpsh_cnt*
37	forward urg counter	*furg_cnt*
38	backward urg counter	*burg_cnt*

The resulting flows are then integrated with the algorithm described in chapter 4, Algorithm Methodology, completing the proposed system. Figure 5.1 displays the complete system diagram, after integrating the flow generation step with the MOGA described in section 4.2.

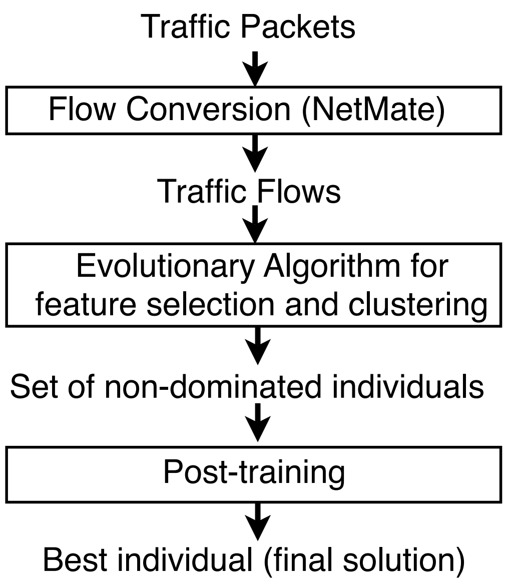
Traffic Packets
Flow Conversion (NetMate)
Traffic Flows
Evolutionary Algorithm for feature selection and clustering
Set of non-dominated individuals
Post-training
Best individual (final solution)

Figure 5.1: Complete System Diagram

5.4 Conclusions

This chapter has described the application specific details of the model proposed in this thesis. The network traces under a flow based representation are fed into the MOGA proposed in chapter 4, for encrypted traffic identification. The following chapter, Analysis of Clustering Objectives, will explore the effects of the MOGA over the clustering objectives.

Chapter 6

Analysis of Clustering Objectives

6.1 Introduction

This chapter examines the effects of MOGA over the four predefined clustering objectives described in chapter 4, *Fwithin*, *Fbetween*, *Fclusters*, and *Fcomplexity*. Given the stochastic component of the fitness evaluation (fitness is based on a roulette wheel proportional to individuals ranks), these objectives are not expected to necessarily increase in a hill climbing mode. Instead, it could be the case that values fluctuate as the MOGA explores the solution space. The average values should, however, observe a somewhat constant increase.

6.2 Clustering Analysis

To analyze the effects of the MOGA over the four predefined clustering objectives (*Fwithin*, *Fbetween*, *Fclusters*, and *Fcomplexity*), 5 runs of MOGA are set, plotting only the non-dominated individuals from each run. The non-dominated individuals from the five runs were combined, plotting their objective values every 1000 epochs. The idea was to monitor the effects of MOGA on each objective over time, from epoch 0 to epoch 5000. By doing so, it is possible to verify whether the population's fitness is enhanced over the epochs as expected.

6.2.1 Fwithin

Figure 6.1 shows the minimum, median and maximum values, as well as the first and third quartile of the *Fwithin* objective every 1000 epochs. Figure 6.2 scales the same diagram so that it is possible to better observe the increase in the median values. Figure 6.3 shows the average values of *Fwithin* every 1000 epochs. As it can be observed from these three diagrams, in the case of *Fwithin* for the five conducted runs, there is a steady increase in its values, more notoriously after epoch 3000.

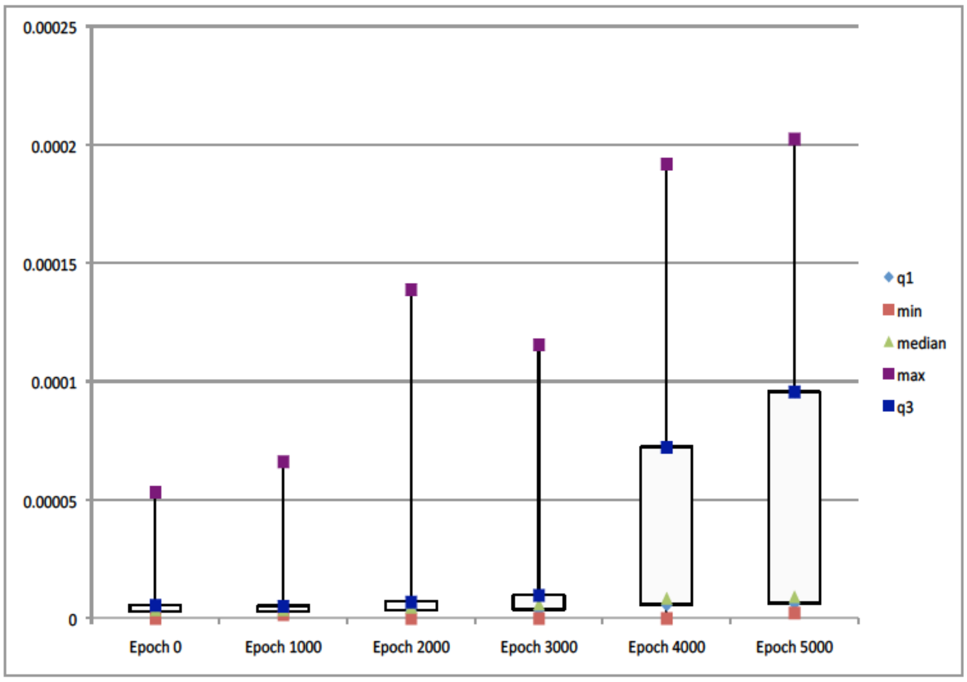

Figure 6.1: *Fwithin* Objective over 5000 Epochs

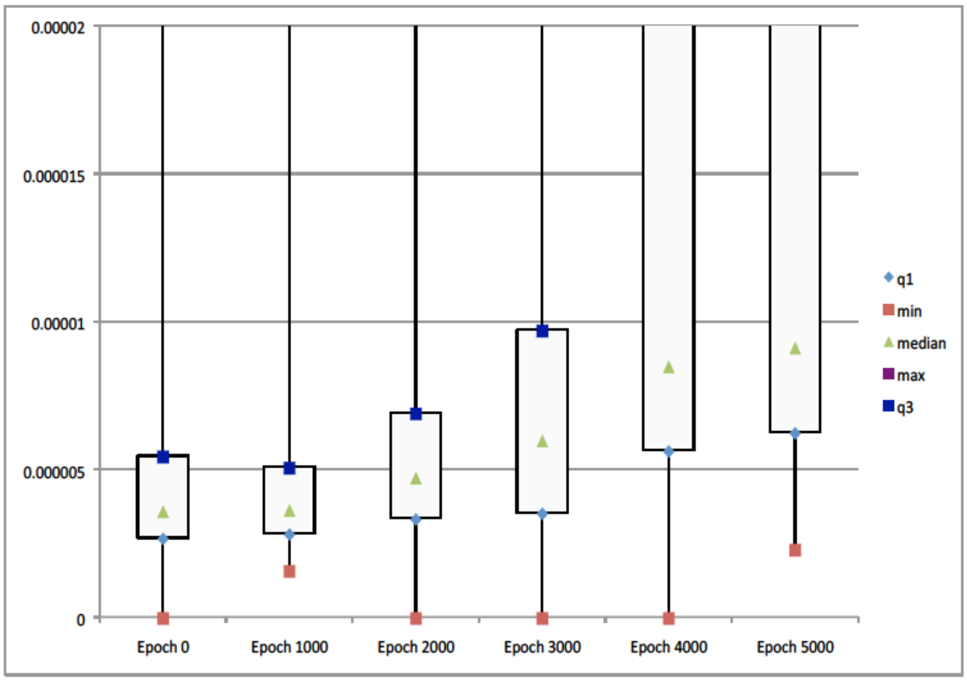

Figure 6.2: *Fwithin* Objective over 5000 Epochs, scaled

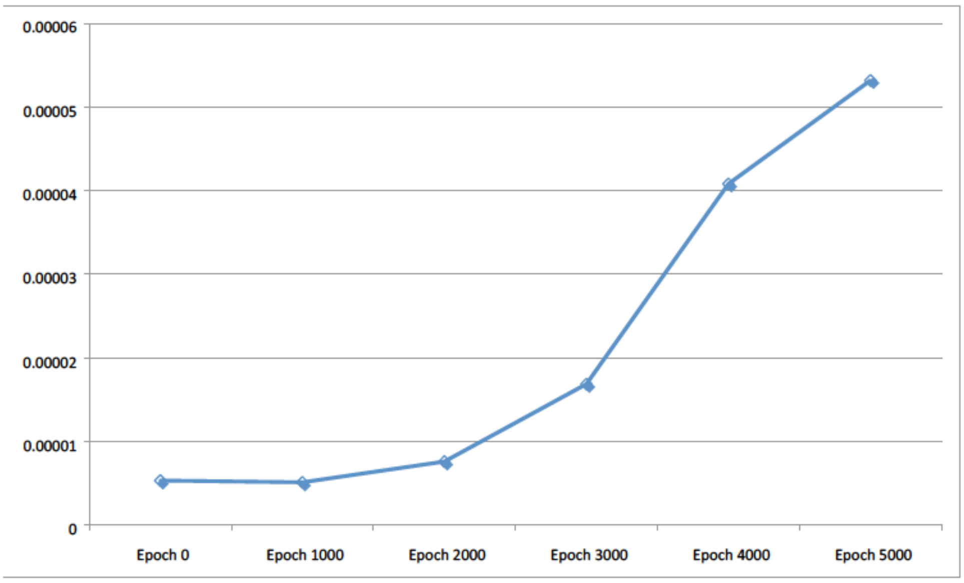

Figure 6.3: *Fwithin* Averages over 5000 Epochs

6.2.2 Fbetween

Figure 6.4 shows the minimum, median and maximum values, as well as the first and third quartile of the *Fbetween* objective every 1000 epochs. Figure 6.5 shows the same diagram, but scaled so that it is possible to better observe the increase in the median values. Figure 6.6 shows the average values of *Fbetween* every 1000 epochs. It is interesting to notice from Figure 6.4 that higher maximum values were achieved at earlier epochs (0, 1000, and 2000). However, for this analysis it is more interesting the steady increase of the median, and of the third quartile, as those values give an idea of where the majority of the higher values are located. As in the previous case, from these three diagrams it can be observed a steady increase in the median, third quartile, and also on the average values.

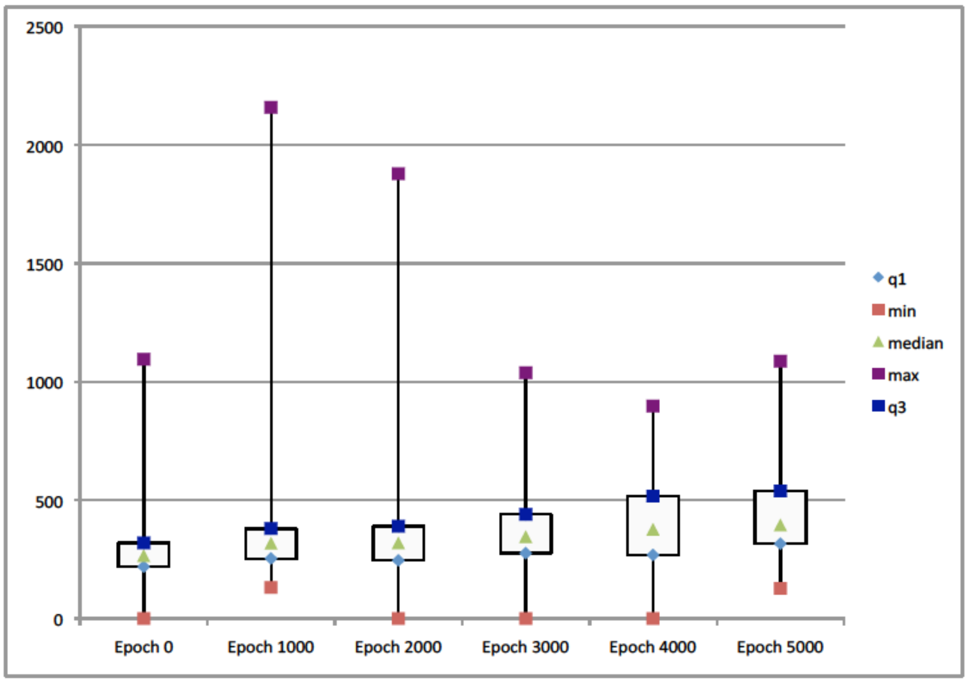

Figure 6.4: *Fbetween* Objective over 5000 Epochs

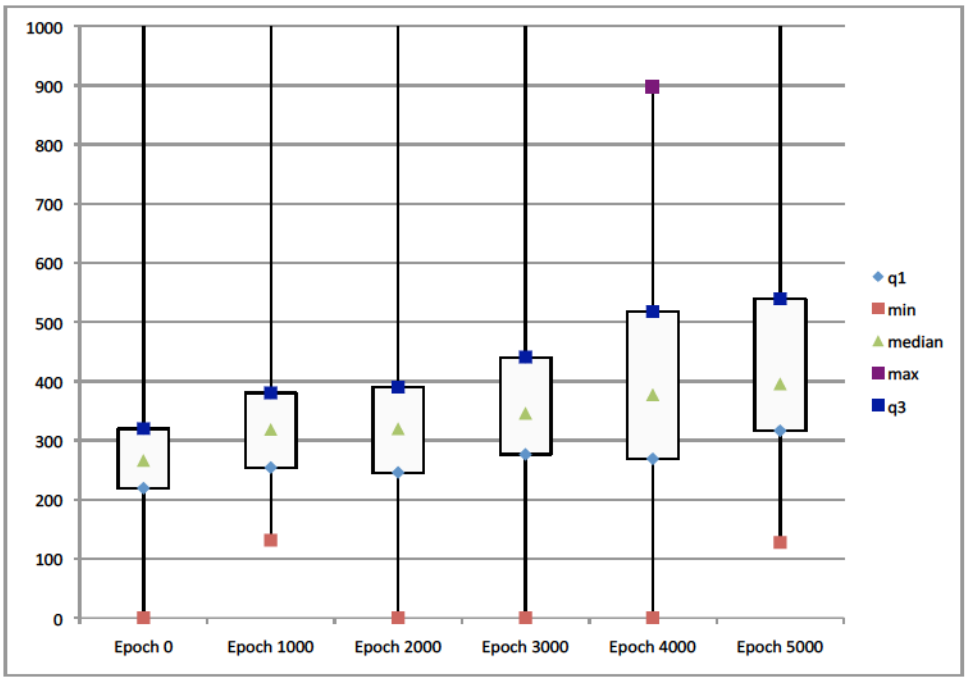

Figure 6.5: *Fbetween* Objective over 5000 Epochs, scaled

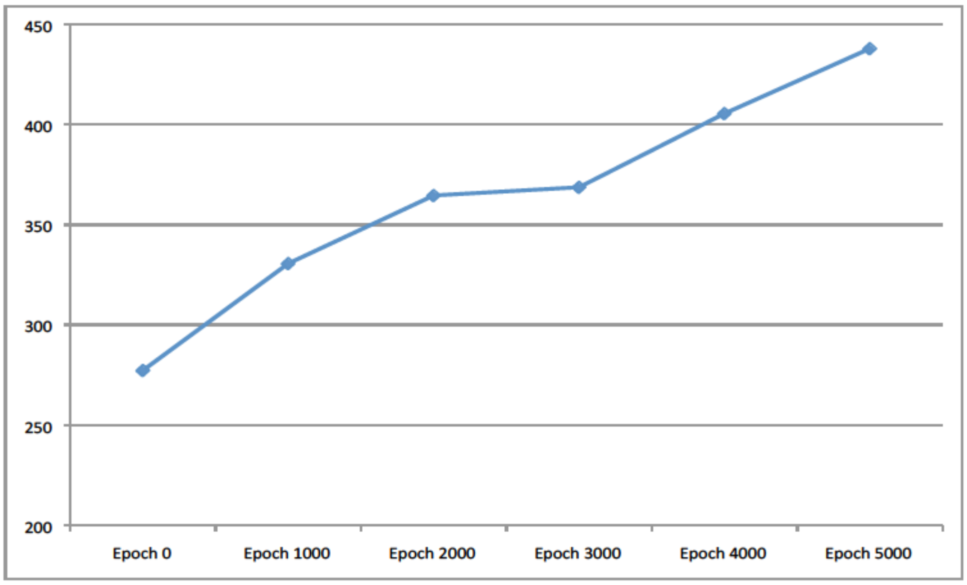

Figure 6.6: *Fbetween* Averages over 5000 Epochs

6.2.3 Fclusters

Figure 6.7 shows the minimum, median and maximum values, as well as the first and third quartile of the *Fclusters* objective every 1000 epochs. Figure 6.8 shows the average values of *Fclusters* every 1000 epochs. In this case, it can be observed a decrease in values until epoch 2000-3000, when the values start increasing. This initial decrease in values can be attributed to the fact that MOGA does not search the solution space in a hill climbing mode.

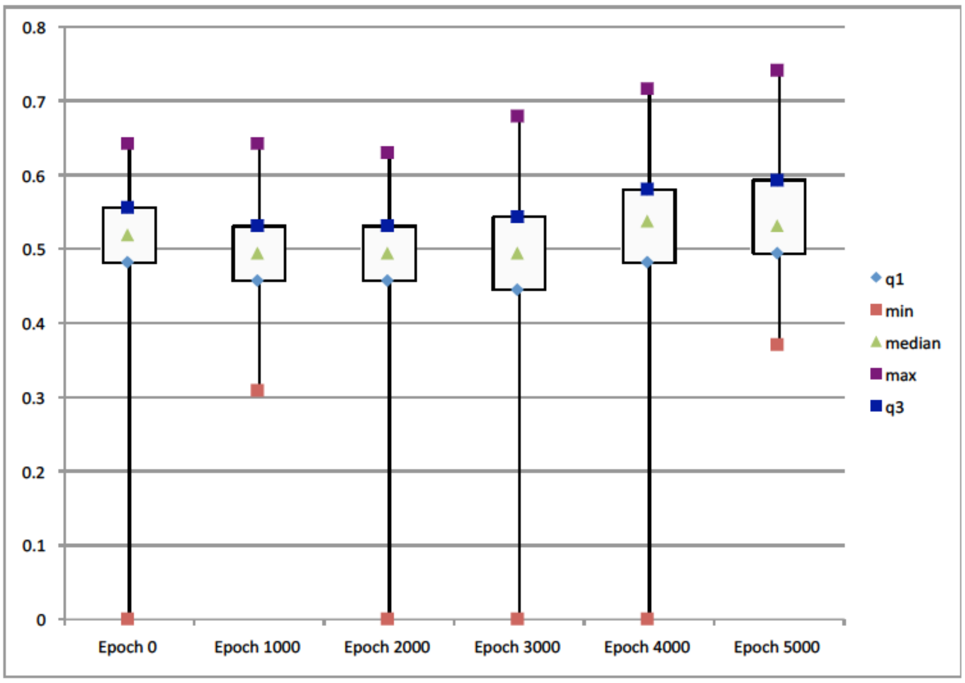

Figure 6.7: *Fclusters* Objective over 5000 Epochs

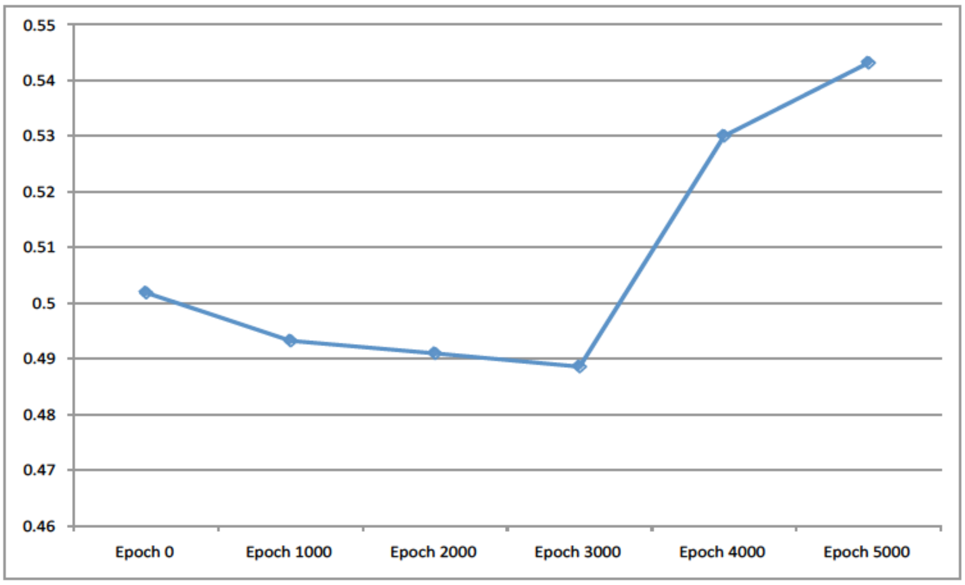

Figure 6.8: *Fclusters* Averages over 5000 Epochs

6.2.4 Fcomplexity

Figure 6.9 shows the minimum, median and maximum values, as well as the first and third quartile of the *Fcomplexity* objective every 1000 epochs. Figure 6.10 shows the average values of *Fcomplexity* every 1000 epochs. In this case, it can be observed a somewhat steady increase of the objective values over the 5000 epochs.

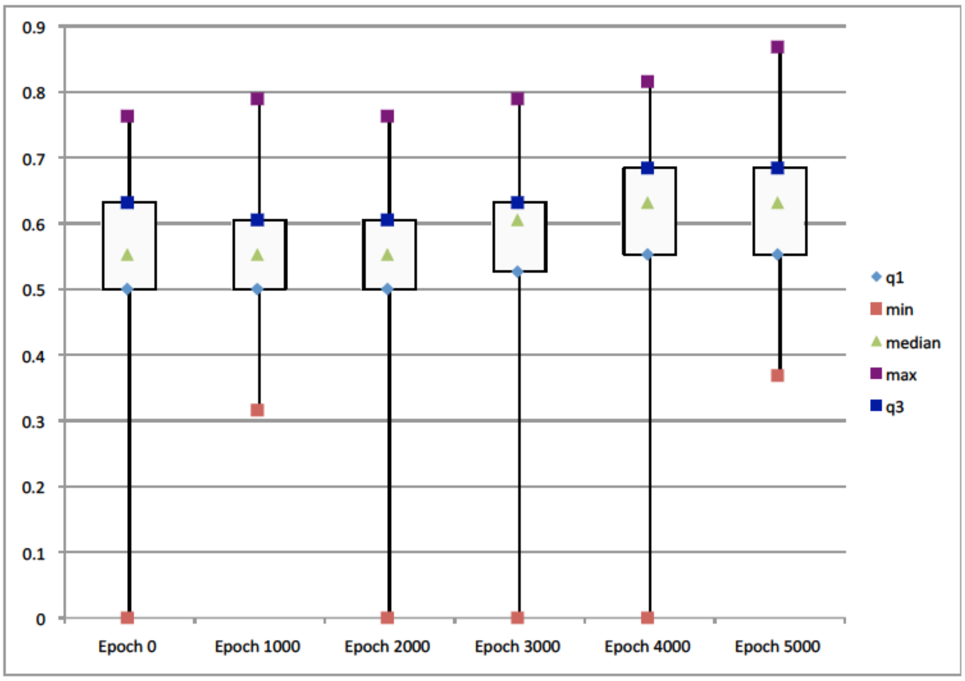

Figure 6.9: *Fcomplexity* Objective over 5000 Epochs

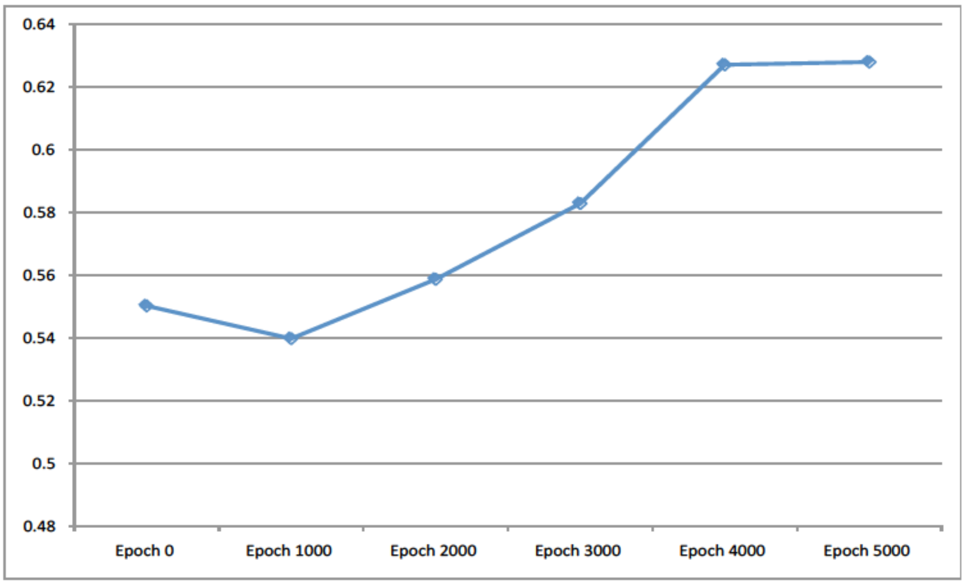

Figure 6.10: *Fcomplexity* Averages over 5000 Epochs

6.3 Conclusions

This chapter has reviewed the effects of the MOGA over the four objectives to be optimized. It can be observed a somewhat steady increase in the value of the objectives, with some fluctuation in their values as the MOGA explores the solution space. Also, this analysis has been based on five independent runs. Different runs could show different fluctuations, however, it should still be possible to observe a similar overall rise of the objective values through the epochs. In short, in this chapter it has been demonstrated that the implemented algorithm effectively enhances the objective values. The next step will be to asses the effectiveness of these objectives for the intended purpose of traffic classification.

Chapter 7

Analysis of the Unsupervised Learning Model

7.1 Introduction

This chapter investigates to what extend the clustering objectives introduced in chapter 4, guide the learning of the MOGA towards achieving good classification rates. This analysis is relevant because the use of clustering objectives only indirectly pertains to the purpose of traffic identification, i.e., clustering data description is not necessarily the same as classification. Thus, instead of using clustering objectives, label information could have been used during training, which would have directly guided the learning towards achieving good classification rates. However, the use of labels during the learning phase is computationally much more expensive, as it involves iterative calculations of classification metrics. Moreover, the use of labels implies the availability of labeled training data in the first place, which is expensive to generate. Still, to evaluate how effective the proposed clustering objectives are in guiding the learning of the model towards achieving good classification rates, a second model was implemented in which the learning was driven by labels. This second model was used as a gold standard against which the performance of the proposed model was compared. Thus, the first objective in this chapter is to explore whether it is possible to mimic the results obtained with the gold standard objectives (label driven learning), with the original clustering objectives.

The second objective of this chapter is to establish the significance of a prior attribute normalization on the learning of the system. Network traffic under a flow based representation utilizes attributes representing different properties, such as time related features and packet length related features. As a result, significant variation in attribute ranges is commonly observed. Such diversity in attribute ranges is typically considered to have a negative impact on machine learning algorithms in general, resulting in the wide spread use of a prior attribute standardization/normalization to achieve a common variance or dynamic range across all attributes. To this end, many approaches [41, 21, 20, 22, 44] in the literature apply a logarithmic transformation to the data. Thus, the effects of applying a logarithmic transformation to the data set to increase the homogeneity between the attributes are analyzed. This is believed [41] to lead to better classification results.

7.2 Gold Standard Model

The gold standard model is established by defining a second set of objectives, in which *Fwithin* and *Fbetween* are replaced with Detection Rate (*DR*) and False Positive Rate (*FPR*) as clustering objectives, but keeping the *Fcomplexity* and *Fclusters* objectives. By doing so, the learning of the model is guided towards achieving better classification results, rather than towards generating high quality clusters. However, this second approach has the disadvantage of (i) being computationally much more expensive, as the calculation of *DR* and *FPR* requires a test run over the entire training data for each individual evaluation, and (ii) being label dependent, requiring a labelled training data that is expensive to generate. Thus, the interest in this chapter is to identify under what conditions (if any) the purely cluster style objectives are able to approach the performance of the explicitly label driven approach for traffic identification. Such an analysis will consider both classification performance and attribute support.

7.3 Logarithmic Transformation

It is important to observe that not all 38 features from Table 5.1, chapter 5, belong to the same type. Instead, there is a mix of time related features, with packet length related features, and others, resulting in significant range differences between their values. These differences can account for up to seven orders of magnitude between their average values. Presumably, differences of this order could bias the design of clusters towards some of the features, not because of class discriminating characteristics, but because of their range in values. I.e., clustering is a data description process, thus will be biased to modeling the most frequent/dominant properties in the data. Whether this is an appropriate bias for traffic discrimination is unknown. To test the impact of a logarithmic transformation to reduce the effect of these range differences, additional experiments are conducted where a log transformation was applied to each attribute. In the literature, it is believed that the logged data should observe much lower inter attribute variation, thus less bias towards any feature in particular, potentially resulting in different features being identified as appropriate support for clustering.

7.4 Feature Selection Results

A total of four sets of experiments were conducted. The first set consisted of running the MOGA with the original clustering objectives, without applying a logarithmic transformation. The second set consisted of running MOGA with the original objectives, but after applying a logarithmic transformation to the data. The third and fourth sets of experiments consisted of running MOGA with the gold standard objectives, without applying logarithmic transformation in the third set, and after applying logarithmic transformation in the fourth set. Each set of experiments consisted of 25 independent runs, from which the non-dominated individuals were selected. Then those non-dominated individuals were combined, and a subset that was considered to be the best individuals was taken. In the case of the runs with the original objectives, the best individuals were those with the highest intra and inter-cluster distances, and in the case of the gold standard model, the best individuals were those with *DR* above 90% and *FPR* under 0.6%.

Figures 7.1 to 7.4 show the features selected by the best individuals in each set of experiments. The vertical axis contains the 38 available features from Table 5.1, chapter 5, and the horizontal axis represents the percentage of best individuals that employed each feature. Figure 7.1 and Figure 7.2 demonstrate that MOGA with the original objectives selects a subset of the features identified by the gold standard model without logarithmic transformation. Both time and packet length related features are selected by the best individuals. It appears that the standard deviation forward inter arrival time, *std_fiat*, and the minimum backward inter arrival time, *min_biat*, were selected by almost all of the individuals in both sets of experiments. Also, it can be seen from these figures that with the exception of the features between *duration*, and *std_idle*, the remaining 26 features were selected with a very similar frequency. On the other hand, once the logarithmic transformation is included, it is observed that the features selected with the original objectives significantly differ from the features selected by the gold standard model; compare Figures 7.3 and 7.4. With the exception of the standard backward inter arrival time, *std_biat*, the other selected features seem to be the opposite as the ones selected by the gold standard model.

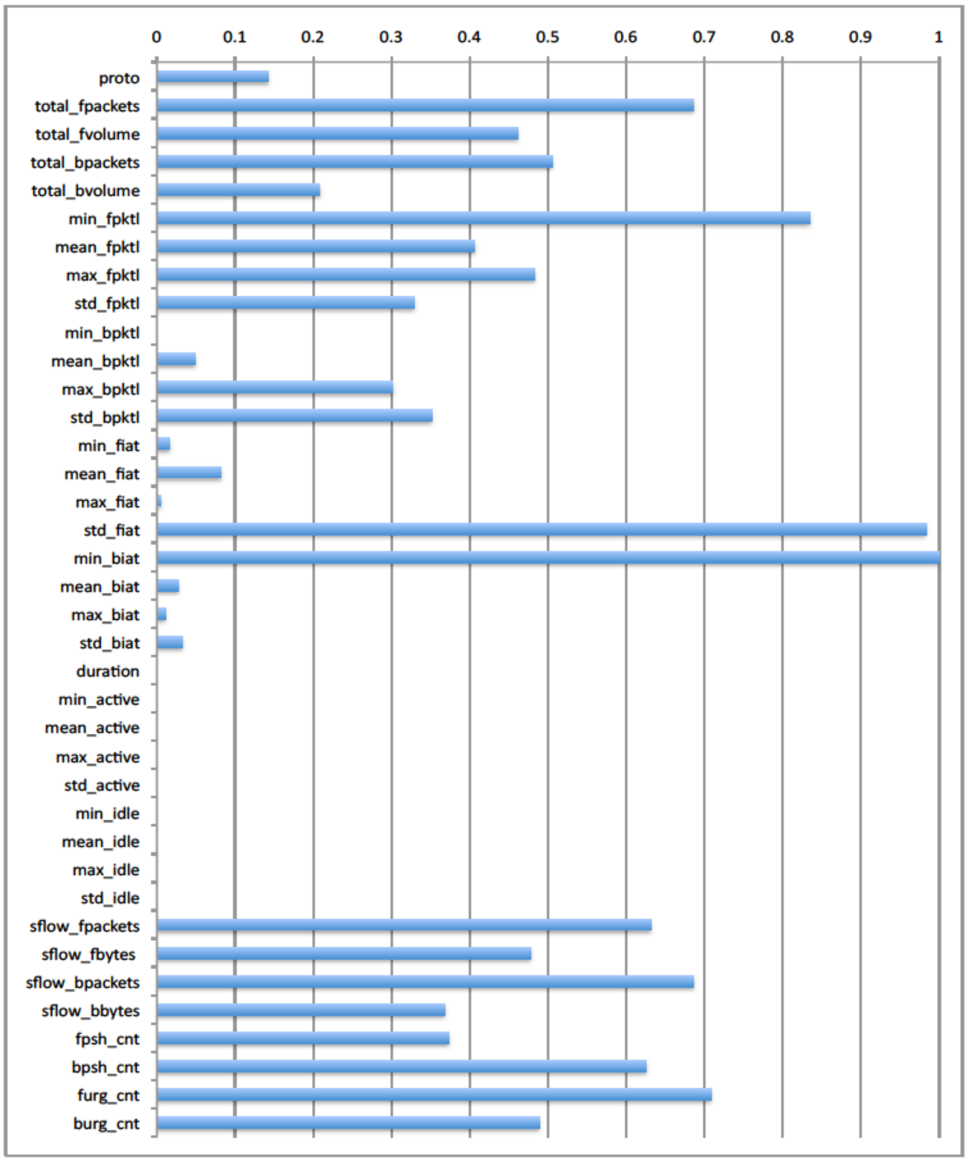

Figure 7.1: Features selected with original objectives without log. standardization.

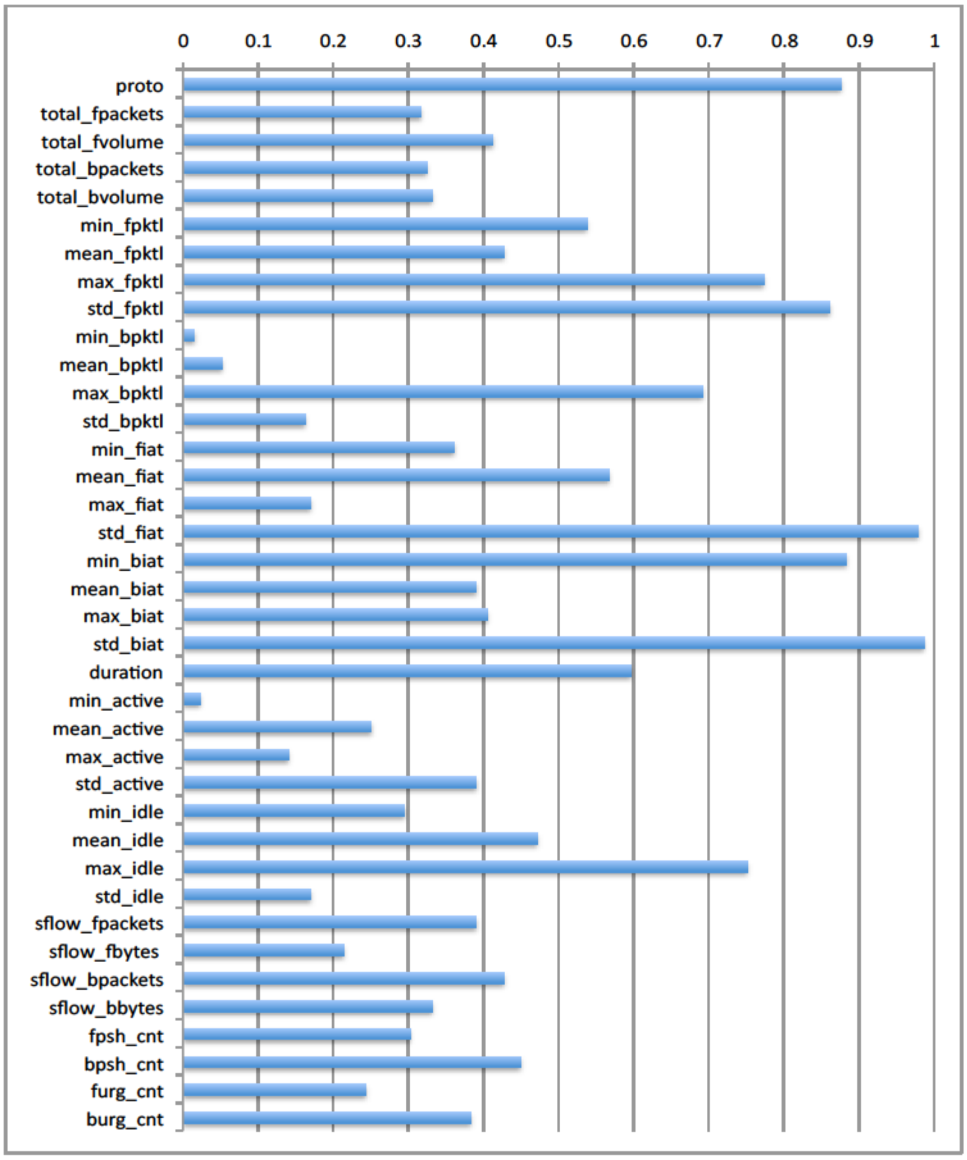

Figure 7.2: Features selected with gold standard without log. standardization.

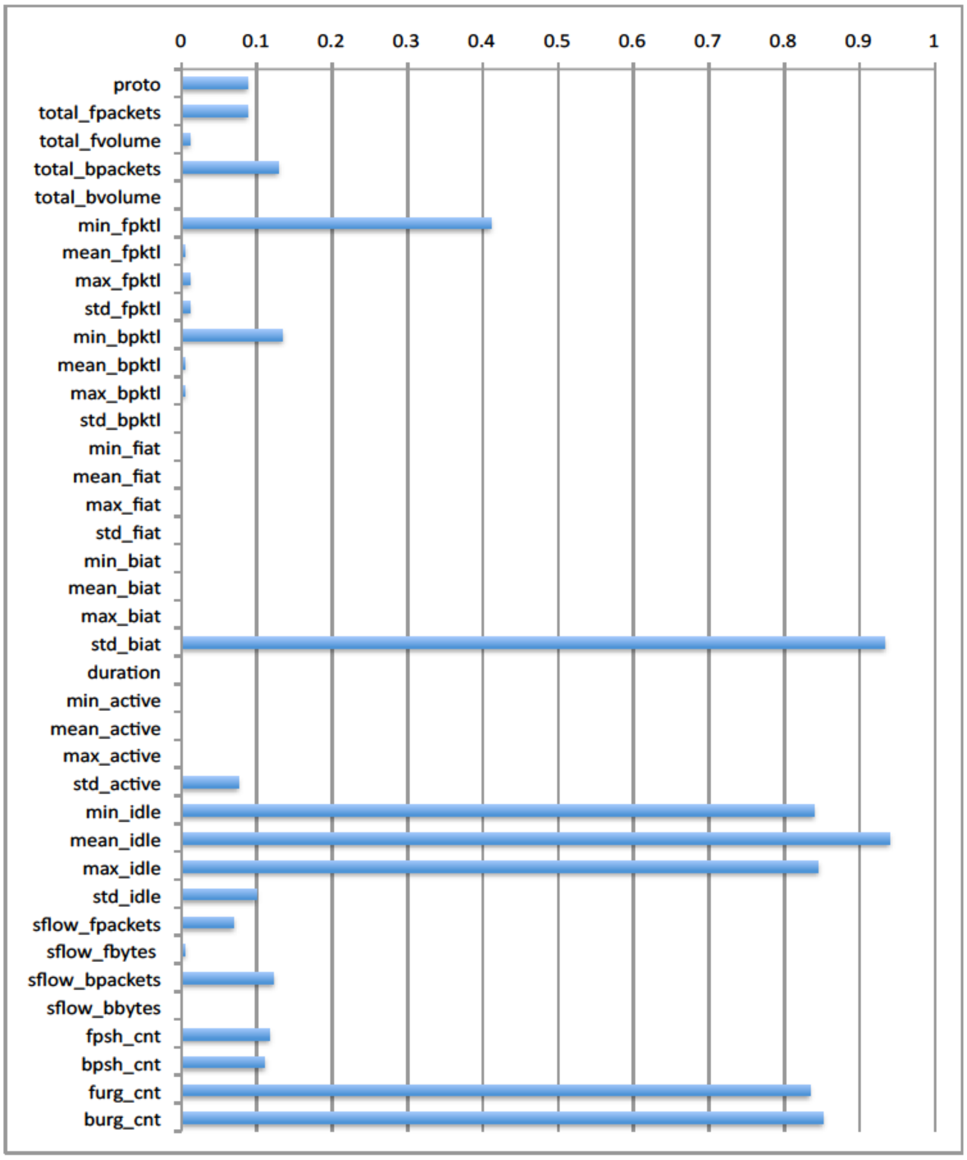

Figure 7.3: Features selected with original objectives with log. standardization.

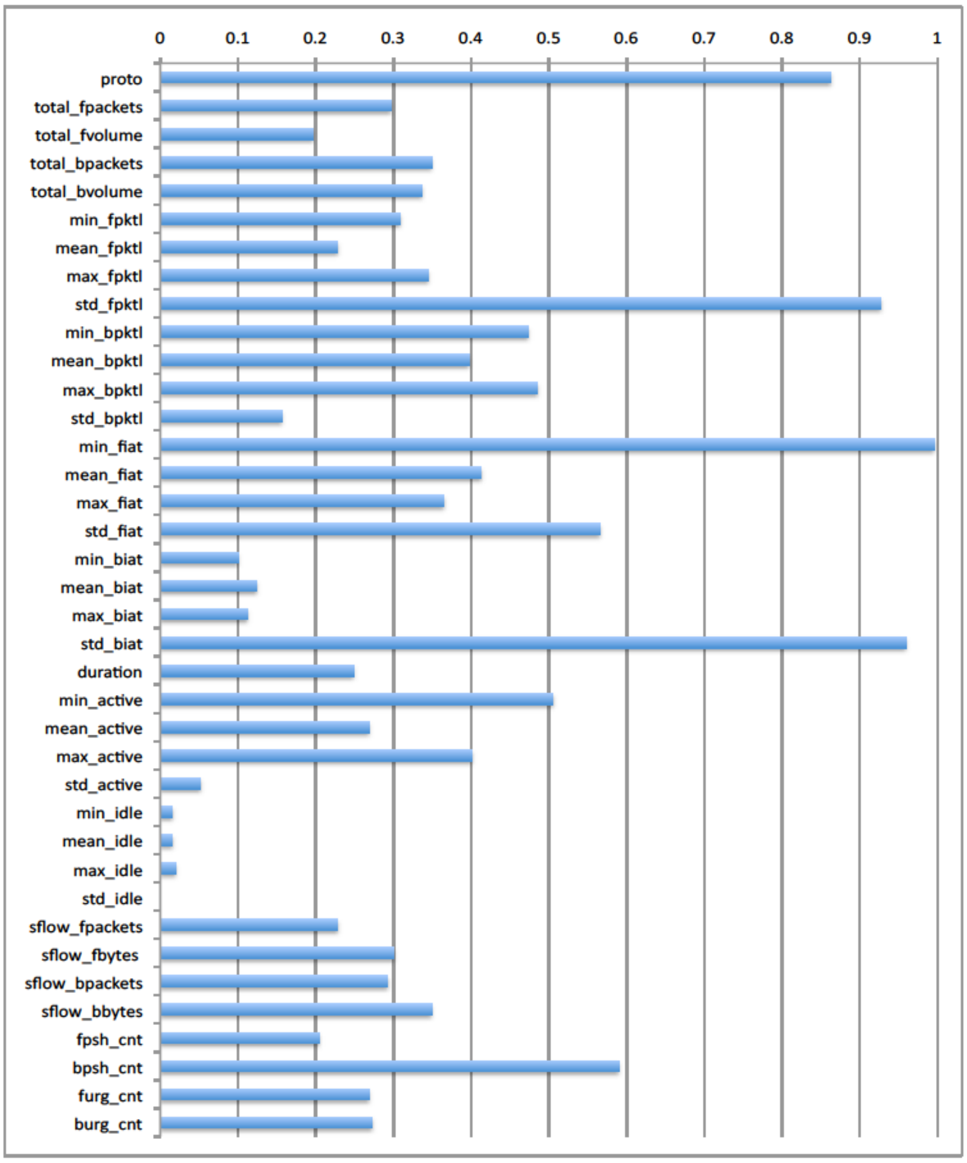

Figure 7.4: Features selected with gold standard with log. standardization.

It is interesting to note how without the log transformation, the model trained with the clustering objectives never selects the features between *duration* and *std_idle*, Figure 7.1. To investigate this phenomenon, the values of the attributes in the flows of the training data were plotted. Figure 7.5 shows the maximum, minimum, and median values, as well as first and third quartiles. As it can be observed, the maximum values of the attributes between *duration* and *std_idle* are much larger than the rest of the attributes, whose values appear to be minimal in comparison. It is the existence of these larger values what prevents MOGA from favoring any individual that makes use of these attributes. Such large values would have larger variances, which would ultimately lead to lower values in the clustering objectives *Fwithin* and *Fbetween* (section 4.3, chapter 4).

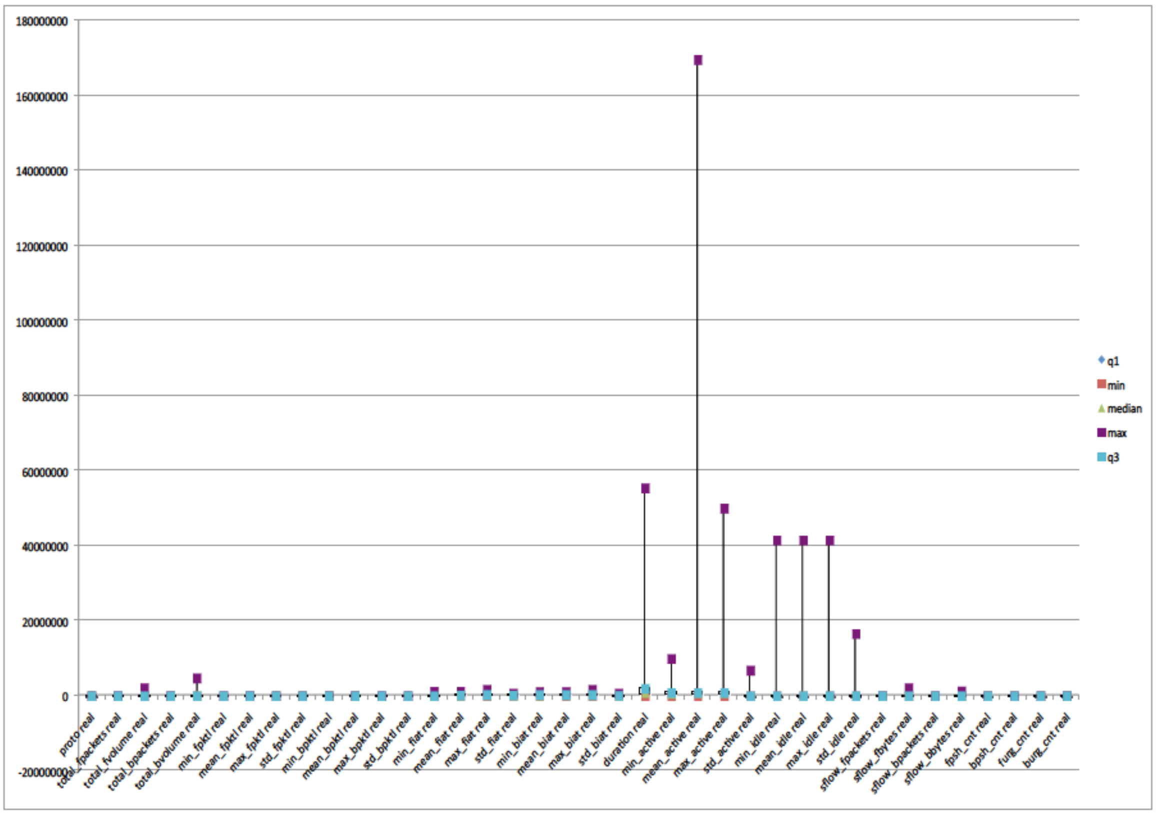

Figure 7.5: Box Plot of Features Values in Training Data

On the other hand, when the attributes in the training data after the log transformation are plotted, it can be observed a more even distribution across the features, Figure 7.6. Still, having a more homogeneous representation after the logarithmic transformation did not lead the MOGA to mimic the behavior of the gold standard model.

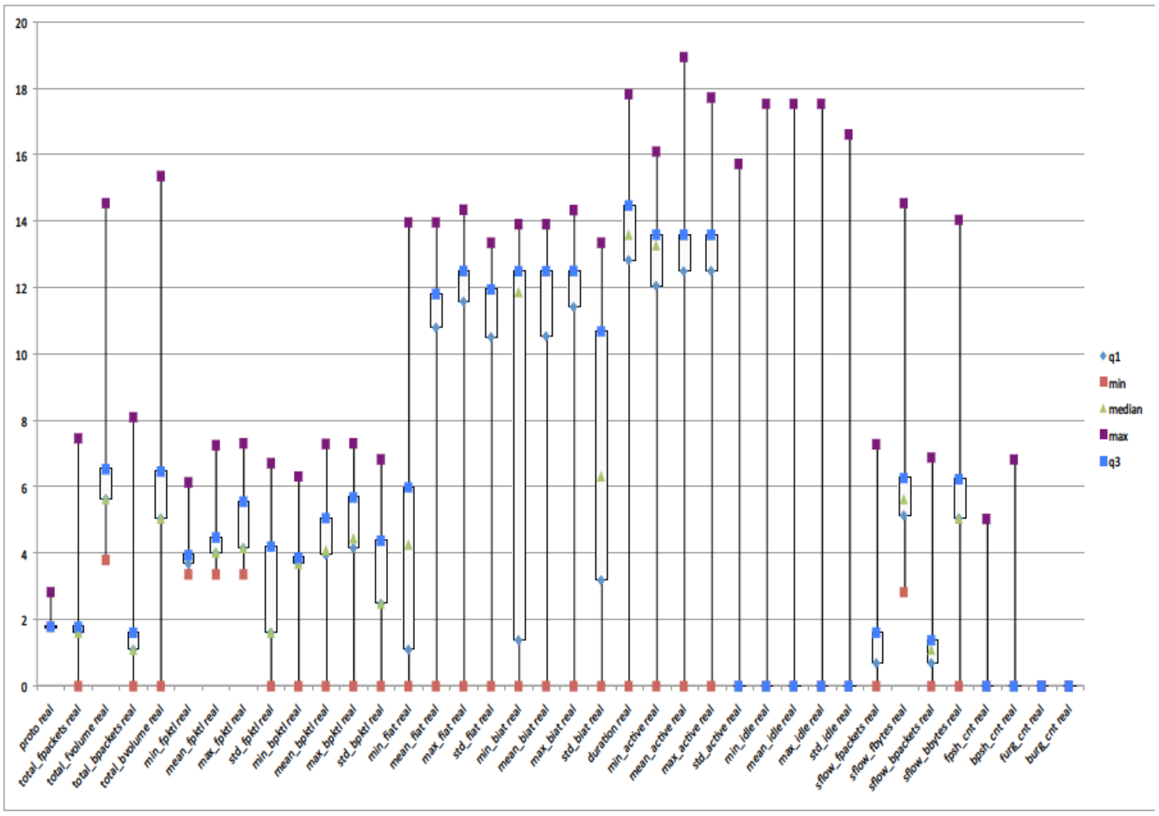

Figure 7.6: Box Plot of Features Values in Training Data After Log Transformation

7.5 Performance Results

To investigate the effects of feature selection in the classification performance of the model, a post training evaluation is conducted, as described in chapter 4. The best non-dominated individuals per set of experiments are tested in the training data, in order to identify the ones with the best classifications rates. The individual with the best classification performance in post training becomes the final solution for that set of experiments, and it is then tested in the entire data set (chapter 5).

Figures 7.7 to 7.10 show the plot of the best individuals performances during the post training phase. The vertical axis represents the *DR* and the horizontal axis represents the *FPR*. The final solution per experiment is marked with a darker square on each plot. From Figure 7.7, it can be observed that with the original objectives and without the logarithmic standardization, the final solution achieves a *DR* of 93.5% and a *FPR* of 0.25% in post training. That same individual achieves a *DR* of 90% and a *FPR* of 0.4% when tested on the entire data set (Table 7.1). In comparison, the gold standard model achieves a *DR* of 93.9% and a *FPR* of 0.22% in post training (Figure 7.8), and a *DR* of 90% and a *FPR* of 0.8% when tested on the entire data set. It can be concluded that the original objectives are capable of closely mimicking the performance of the gold standard model. This does not come as a surprise, as it was already observed that the original model generally selects a similar set of features as the gold standard model.

On the other hand, Figures 7.9 and 7.10 show that the results achieved in post training when applying the logarithmic standardization, differ between the original objectives and the gold standard objectives. With the original objectives, the final solution achieves a *DR* of 95.4% and a *FPR* of 1.1% in post training (Figure 7.9). Notice that the rest of the individuals do not appear on the plot for being out of scale with much larger *FPR*. That same individual achieves a *DR* of 91.4% and a *FPR* of 17% when tested in the entire data set, which is a prohibitory high *FPR*. The gold standard model, Figure 7.10, achieves a *DR* of 94.4% and a *FPR* of 0.16% in post training, and a *DR* of 91% and a *FPR* of 0.2% when tested in the entire data set.

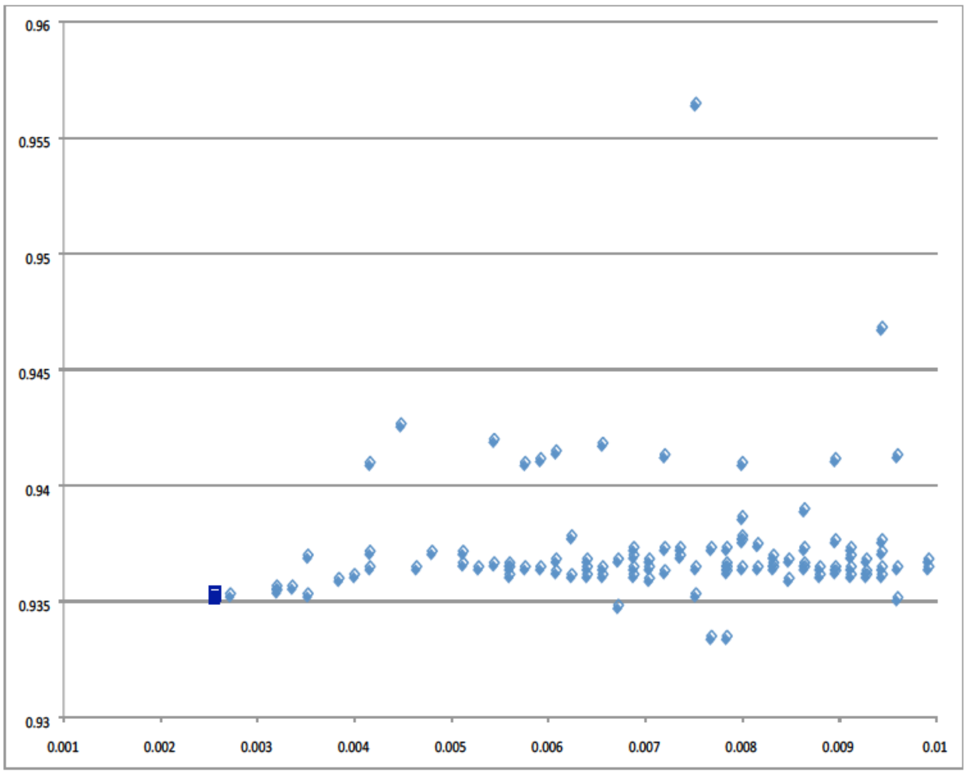

Figure 7.7: Post training with original objectives without logarithmic standardization. *DR* (y-axis) over 93 to 96% range; *FPR* (x-axis) over 0.1 to 1% range.

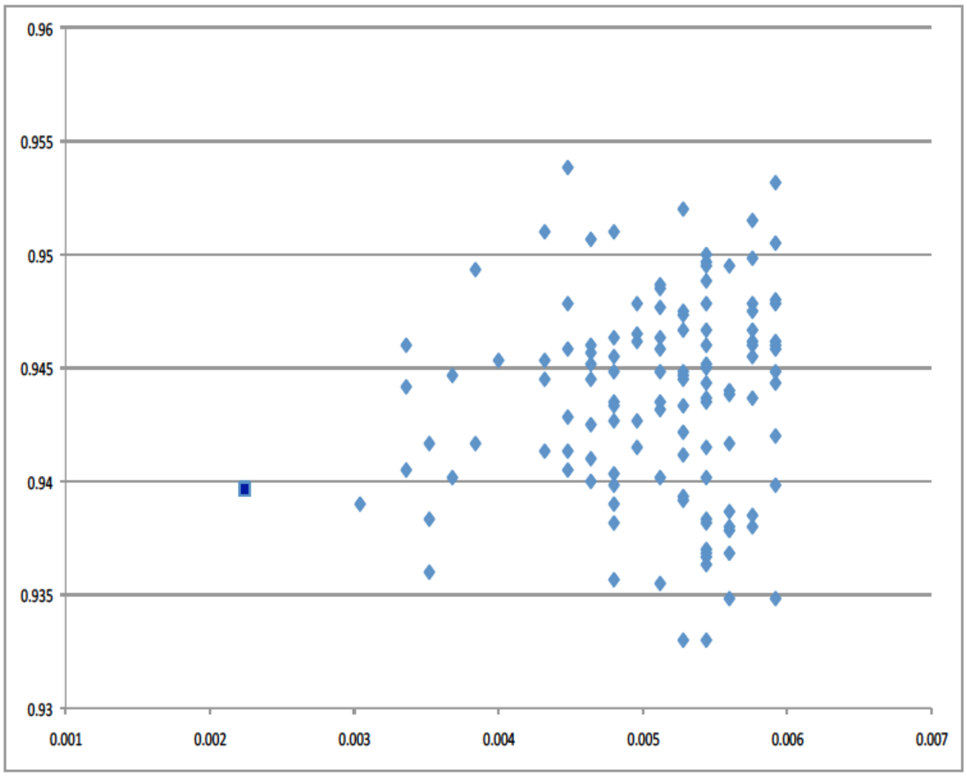

Figure 7.8: Post training with gold standard without logarithmic standardization. *DR* (y-axis) over 93 to 96% range; *FPR* (x-axis) over 0.1 to 0.7% range.

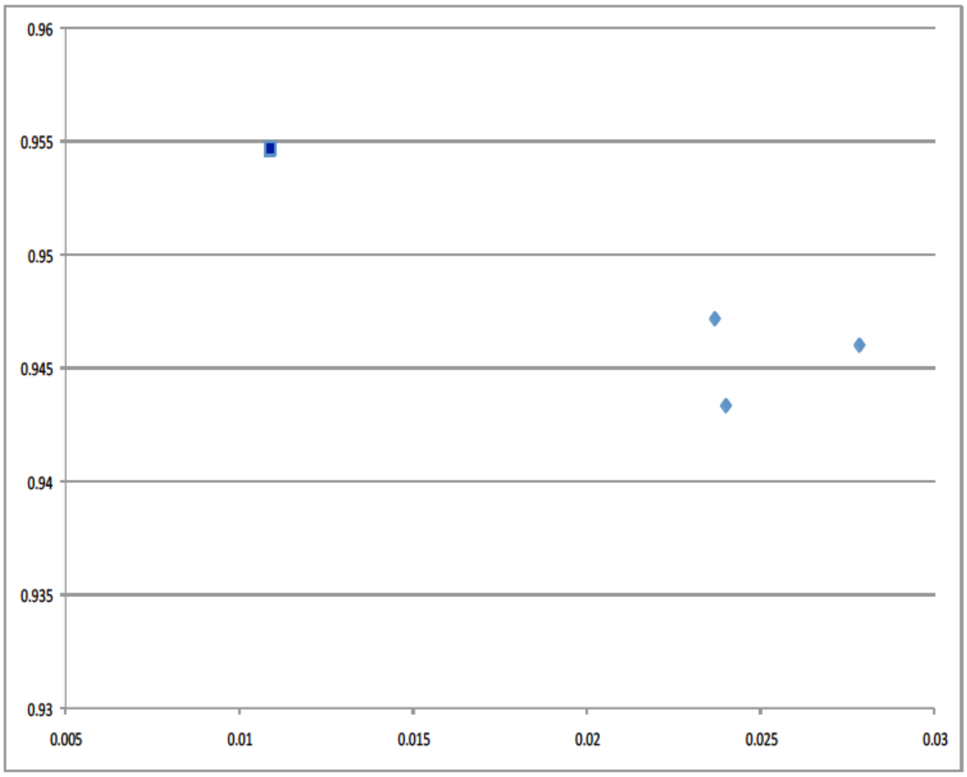

Figure 7.9: Post training with original objectives with logarithmic standardization. *DR* (y-axis) over 93 to 96% range; *FPR* (x-axis) over 0.5 to 3% range.

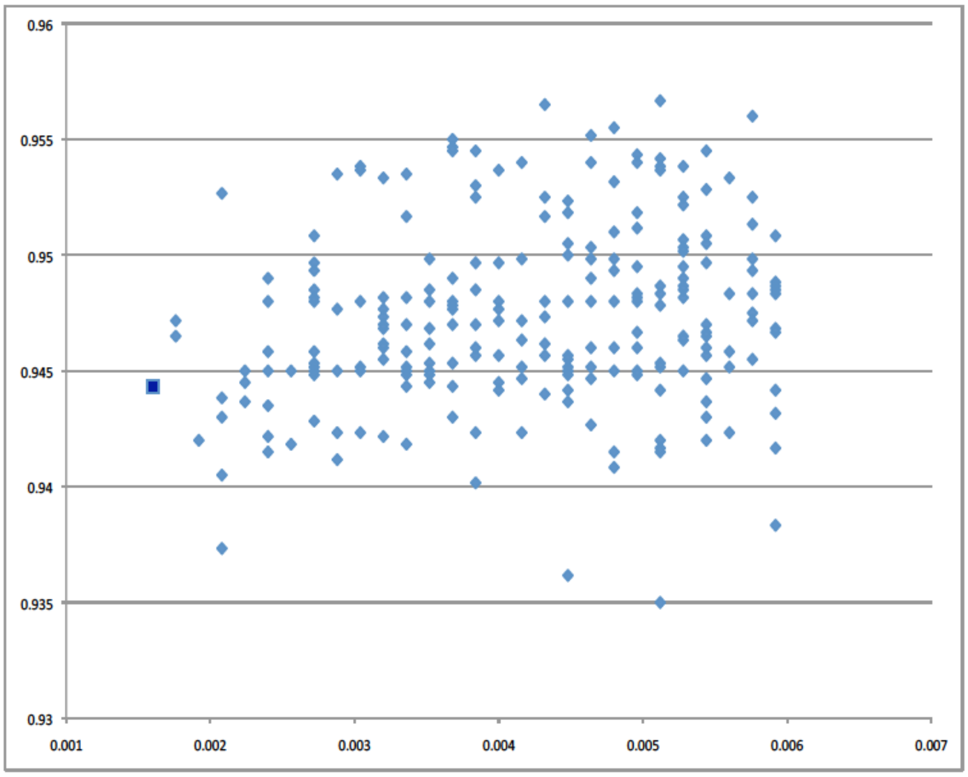

Figure 7.10: Post training with gold standard with logarithmic standardization. *DR* (y-axis) over 93 to 96% range; *FPR* (x-axis) over 0.1 to 0.7% range.

Thus, it can be concluded that because the original objectives under a logged data do not select the same features as the gold standard, the performance of the MOGA with the original clustering objectives is inferior to that of the gold standard model. All the test results are summarized in Table 7.1.

Table 7.1: MOGA vs Gold Standard Test Results

Experiment	DR	FPR
Original Objectives	90.0%	0.4%
Gold Standard	90.0%	0.8%
Original Objectives logged	91.4%	17.0%
Gold Standard logged	91.0%	0.2%

7.6 Conclusions

With regards to the first goal, it is observed that the original cluster style objectives can quite closely mimic the behavior of the gold standard classifier style objectives in the non-logged experiments. In that case, it is observed that not only both sets of experiments selected a similar set of features, but also that the results from testing both models' best individuals on the entire data set show very similar performances. The gold standard model achieves a *DR* of 90% with a *FPR* of 0.8% when tested on the entire data set, and the original objectives also achieve a *DR* 90% with a *FPR* of 0.4%.

With regards to the second goal, on the other hand, it is observed that after applying a log transformation to the attributes, the original objectives do not mimic the behavior of the gold standard model. Both sets of experiments do not select similar features. Moreover, the classification performance achieved with the original objectives is considerably lower than the logged gold standard model. The gold standard model achieved a *DR* of 91% with a *FPR* of 0.2% when tested on the entire data set, whereas the original model achieved a *DR* of 91.4% with a *FPR* of 17%, which is prohibitory high.

As for a feature selection analysis, it is observed that the best results are achieved consistently with a mix of time and packet length related features. In particular, the features *std_fiat* and *min_biat* were employed by almost all of the best individuals in the non-logged experiments. Also, it is observed that the features between *duration* and *std_idle* have significantly higher maximum values, which prevents the MOGA from selecting them in the non logged experiments. There is no reason to believe that these features could not potentially contribute to a better class discrimination. Thus, it would be interesting to search for a different normalization function in the future. One that enables the inclusion of these features, while at the same time mimicking the behavior of the gold standard model.

Chapter 8

MOGA vs Other Unsupervised Learning Techniques

8.1 Introduction

In this chapter the performance of the model proposed in this thesis, a MOGA applied to encrypted traffic identification, is benchmarked against other unsupervised learning techniques existing in the literature. Like in the previous chapters, Secure Shell (SSH) is chosen as an example encrypted application. In this case, four unsupervised learning techniques are benchmarked: basic K-Means, semi-supervised K-Means, DBSCAN, and EM, and the results are compared with the proposed MOGA used for the dual identification of appropriate (flow) feature subspace and clustering of traffic types.

8.2 Unsupervised Learning Algorithms

The three clustering algorithms selected for this work are K-Means, DBSCAN, and EM. The selection of these algorithms is based in part on the work of Erman *et al.* in [20], in which the authors made a similar comparison. Furthermore, the proposed model is also benchmarked against a semi-supervised method proposed in [22]. The following subsections give a brief explanation of these algorithms, more details can be found in [20] and [22]. For the implementations of these algorithms, the K-Mean, DBSCAN, and EM provided by Weka [42] is employed, and the criteria of similarity was the Euclidean distance (section 4.4, chapter 4).

8.2.1 Basic K-Means

K-Means clustering algorithm is an unsupervised learning technique that aims to partition n observations into K clusters, in which each observation belongs to the cluster with the nearest mean. Erman *et al.* observed in their experiments that one of the main advantages of the K-Means algorithm over other clustering algorithms was that the resulting clusters tended to be mainly of a single application type [20].

For this experiment, several values of K (20, 40, 60, 80, 100, 200, 300, 400) were tested, and the best result was selected to be compared with the other models. A more detailed explanation of the algorithm can be found in chapter 2.

8.2.2 Semi-supervised K-Means

The semi-supervised approach proposed in [22] was followed, in which high detection rates are achieved by labeling the resulting clusters employing only a small fraction of the training data. Thus, after running a K-Means clustering algorithm, the clusters were labelled post training, using only the labels of 5% of the flows, 613 out of 12250. This model was also trained with larger data sets consisting of 32000 flows, with 80, 800, and 8000 of them labeled. Following the work in [22], the K number of clusters was set to 400. In all of these experiments the total number of SSH flows was 6000, so that there was a base point of comparison with the other methods presented. In addition, the effectiveness of the weighted sampling approach proposed in [22] was also evaluated. A 3602 flows training data was generated, out of which only 180 flows were labeled. For this approach 50% of the flows were selected from below, and 50% of the flows from above the 95th percentile of the flow transfer size of the original training data (12250 flows).

8.2.3 DBSCAN

DBSCAN [24] is a density based algorithm, so it regards "clusters as dense areas of objects that are separated by less dense areas" [20]. The main advantage of this algorithms is that unlike K-Means, it is not limited to find clusters of spherical shape. Instead, dense area clusters can follow any arbitrary shape [20]. The DBSCAN algorithm takes two input parameters, epsilon (*eps*) and the number of minimum points (*MinPts*). *MinPts* is the minimum required points to form a core object, and *eps* is the distance between two objects to be considered "eps-neighbors". DBSCAN does not take as an input the number of clusters to generate, it finds the optimum number of clusters based on the *MinPts* and *eps*. Also, unlike K-Means and EM, if an object is not part of an existing cluster, it is considered noise [20]. A more detailed explanation of the algorithm can be found in chapter 2.

8.2.4 EM

The Expectation Maximization algorithm works with the probabilities of each instance belonging to each cluster [42]. The algorithm has two phases, an expectation phase and a maximization phase. The parameters used by the algorithm that "govern the distinct probability distribution of each cluster" [20] are estimated during the expectation phase, and are continually re-estimated during the maximization phase [20]. A more detailed explanation of the algorithm can be found in chapter 2.

8.3 Methodology

To perform this comparison, the same features used in [22] were employed for the basic K-Means, semi-supervised K-Means, DBSCAN, and EM algorithms, Table 8.1. For the MOGA, on the other hand, all the 38 original features were employed for it to select the most appropriate ones during the training phase, Table 5.1, chapter 5. In order to reproduce the features employed in [22] the *netai_flowstats.c* module of NetMate was modified, originally used to obtained the aforementioned 38 features employed by MOGA. The reason the exact same features as in [22] were employed was to accurately reproduce the methodology described in that paper.

Table 8.1: Flow Features employed in [22]

Total Number of Packets= fpackets+bpackets
Total Caller to Calle Payload Bytes= fvolume-fhlen
Total Bytes= fvolume+bvolume
Total Caller to Callee Header Bytes= fhlen
Total Header (Transport + Network Layer)= fhlen+bhlen
Number of Callee to Caller Packets= bpacket
Average Packet Size= (fpktl + bpktl)/(fpackets + bpackets)
Total Callee to Caller Payload Bytes= bvolume-bhlen
Number of Caller to Callee Packets= fpacket
Total Callee to Caller Header Bytes= bhlen
Total Caller to Callee Bytes= fvolume

8.4 Experiments and Results

To compare the performance of these algorithms we measured the obtained results in terms of Detection Rate (*DR*) and False Positive Rate (*FPR)* (section 4.3, chapter 4). In this case, *DR* will reflect the number of SSH flows correctly classified, and *FPR* will reflect the number of non-SSH flows incorrectly classified as SSH. Given that the encrypted traffic only forms a few percent of the traffic in real life, false positive rates are a very important performance indicator on such heavily unbalanced data sets. As it becomes obvious, a high *DR* and a low *FPR* would be the desired outcomes. For all the conducted experiments the same training and test data was used, with the exception of the semi-supervised K-Means approach, in which additional training data samples were also employed. Both the training and test data sets are described in chapter 5.

8.4.1 K-Means Results

The basic K-Means algorithm was run with values of K from 20 to 400 (Table 8.2). The best combination of *DR* and *FPR* was obtained with $K = 100$, which achieved a *DR* of 98% and a *FPR* of 11%.

Table 8.2: K-Means Results

Number of clusters	*DR*	*FPR*
20	90%	13%
40	94%	11%
60	97%	11%
80	97%	11%
100	98%	11%
200	98%	15%
300	98%	15%
400	98%	15%

8.4.2 Semi-supervised K-Means Results

For the semi-supervised model proposed in [22], the first experiment was conducted with the same training data (12250 flows), with $K = 400$ (as done in [22]), but only considering the labels of 5% (613) of the flows when labeling the clusters. The *DR* for that experiment was 90.1% and the *FPR* was 0%. Then, the training data was expanded to 32,000 flows, considering the labels of 80, 800, and 8000 of the flows (as done in [22]) when labeling the clusters (Table 8.3). The highest *DR*, 98.9%, was achieved with 8000 labelled flows. However, the *FPR* was 19%, which is much higher than the other results. Thus, the best combination of *DR* and *FPR* was achieved with only 800 of the flows labelled, which gives a *DR* of 92% with a 0% *FPR*. Also, in evaluating the effectiveness of the weighted sampling approach, a *DR* of 90% and *FPR* of 0% was obtained, which is the same the results obtained without the weighted sampling method. However, with the weighted sample approach the training data consisted of only 3602 flows, with 180 of them labeled. Thus, from these results it can be observed that the weighted sample approach achieves good results with a very small training data.

Table 8.3: Semi-Supervised Results

Number of clusters	# labelled flows	*DR*	*FPR*
400	80	90.8%	0%
400	800	92%	0%
400	8000	98.9%	19%

8.4.3 DBSCAN Results

The results obtained with DBSCAN are displayed in Figure 8.1. The y-axis represents the detection rate and the x-axis represents the *eps*. Several experiments were conducted to assess the performance of the algorithm with different values for *MinPts* (3, 6, 9, 12) and for *eps* (0.01, 0.02, 0.04, 0.06, 0.08), the selection of both these ranges of parameters followed the work presented in [22]. The best results were achieved with *MinPts*= 3 and *eps*= 0.02, with a *DR* of 47.4% and *FPR* of 47%.

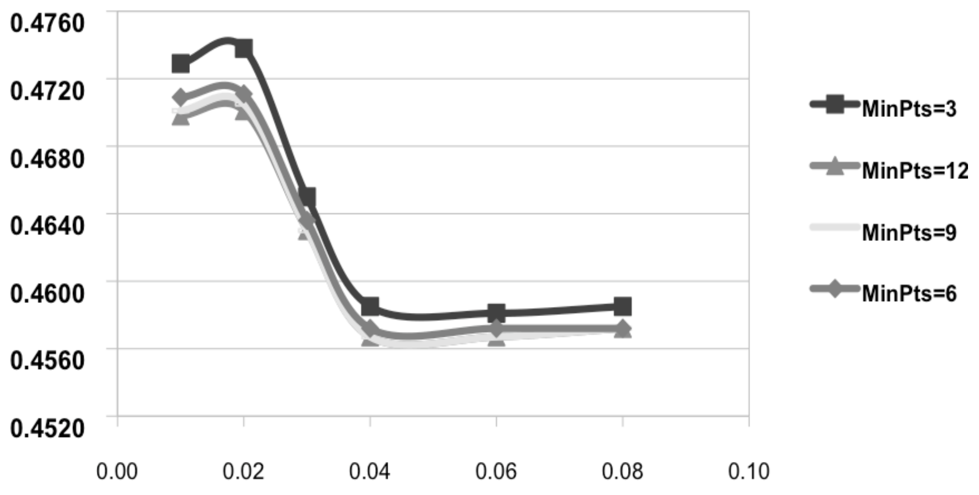

Figure 8.1: DBSCAN Results

8.4.4 EM Results

For the EM algorithm several experiments were conducted with the number of clusters between 100 and 400 (Table 8.4). The selection of these values was in part following the work presented in [22], and in part to have a common point of comparison with the values employed in the K-Means experiments. The best results were achieved with the number of clusters being 400, which gives a high *DR*, 96.4%, while at the same time keeping the *FPR* relatively low, 5%.

Table 8.4: EM Results

Number of clusters	*DR*	*FPR*
100	93.8%	7.9%
200	95.8%	7.6%
300	97.5%	10.6%
400	96.4%	5%

8.4.5 MOGA Results

The MOGA was run in two sets of experiments, of 25 runs each. In the first set of experiments, the length of the individuals in the population was 60, which allowed the individuals to cluster with a minimum K of 2, and a maximum K of 23. In the second set of experiments, the length of the individuals was 100, allowing the K number of clusters to be up to 63. The first set of experiments generated a total of 1173 non-dominated individuals and the second set of experiments generated a total of 869 non-dominated individuals. Out of those individuals, during the post-training phase the ones that had the lowest *FPR* (under 1%) and the highest *DR* were identified. Those individuals were considered to be the final solutions. Figures 8.2 and 8.3 display the plot of the candidate solutions in the post-training phase for both sets of experiments. The x-axis represents the *FPR* and the y-axis represents the *DR*. The best individual in the first set of experiments, which is represented by a larger black square instead of a gray diamond in Figure 8.2, achieved a *DR* of 94.9% and a *FPR* of 0.9% in the post training phase. That same individual achieved a *DR* of 90.9% and a *FPR* of 1.5% in the test data (Table 8.5). This final solution employed 22 out of the 38 available features, and clustered the data into 10 clusters. The best individual in the second set of experiments, which is represented by a larger black square instead of a gray diamond in Figure 8.3, achieved a *DR* of 95.8% and a *FPR* of 0.8% in the post training phase. That same individual achieved a *DR* of 93.5% and a *FPR* rate of 0.7% in the test data (Table 8.5). This final solution employed 18 out of the 38 available features, and clustered the data into 36 clusters. Thus, these solutions not only achieved superior results in terms of *DR* and *FPR*, but also considerably decreased the number of employed features, and clustered the data in a relatively low number of clusters, both very desirable outcomes.

Table 8.5: MOGA Test Results

Number of clusters	*DR*	*FPR*
10	90.9%	1.5%
36	93.5%	0.7%

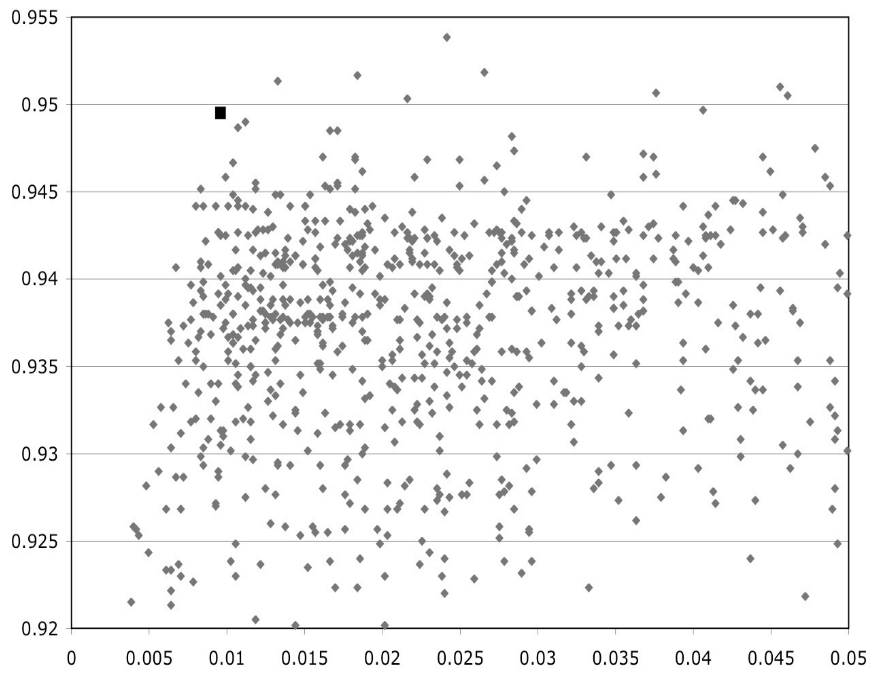

Figure 8.2: Non-dominated individuals length 60.

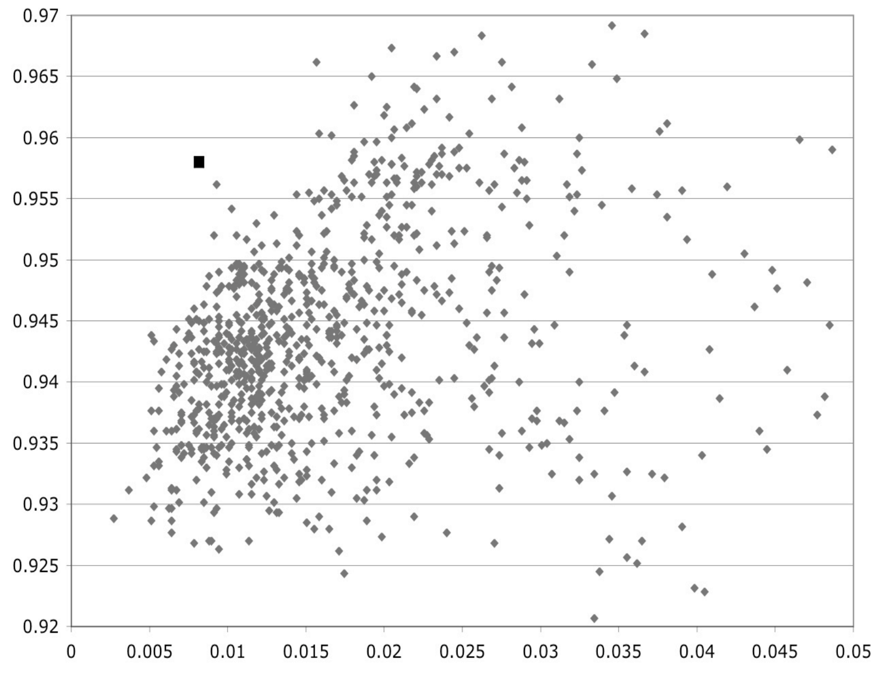

Figure 8.3: Non-dominated individuals length 100.

8.4.6 Discussion

Table 8.6 summarizes the results from the conducted experiments. It contains the best results from each set of experiments per algorithm. Thus, for the K-Means algorithm, a value of $K = 100$ was employed, and for the semi-supervised K-Means a value of $K = 400$ was employed. It should be noted that in the case of the semi-supervised K-Means the training data was about three times larger than with the other algorithms (32,000 flows). For the DBSCAN algorithm, a value of $eps = 0.02$ and a value of $MinPts = 3$ were employed. For the EM algorithm, a value of $K = 400$ was employed. For the MOGA, the two individuals that were selected as the best individuals in each of the two conducted sets of experiments were employed. These results show that MOGA (100 ind) offers a very good combination of *DR* and *FPR*, when compared to the other methods presented here.

Table 8.6: MOGA vs Other Methods Summarized

Algorithm	# clusters	***DR***	***FPR***
K-Means	100	98%	11%
K-Means (semi-sup.)	400	92%	0%
DBSCAN (*mP.=3*)	36	47.4%	47%
EM	400	96.4%	5%
MOGA (ind 60)	10	90.9%	1.5%
MOGA (ind 100)	36	93.5%	0.7%

8.4.7 Time Analysis

One clear advantage of being able to obtain a high *DR* and a low *FPR* with a low number of features and a low number of clusters is the time involved in both, building the models, and testing the data. The building time and the test time were measured for all the models described here (Table 8.7). Specifically, the algorithms with the parameters that produced the best results in the previously described experiments were measured. These results show a considerable advantage of the MOGA over all the other algorithms, except DBSCAN. However, the *DR* obtained with DBSCAN was much lower. These experiments were conducted on a standard PC with an Intel(R) Core(TM) 2 Duo CPU T6400@ 2.00 GHz, with 4 GB of memory.

Table 8.7: MOGA vs Other Methods Time Analysis

Algorithm	# clusters	**Build time**	**Test time**
K-Means	100	00:02:19	00:49:29
K-Means (semi-sup.)	400	00:22:32	03:07:55
DBSCAN (*mP.=3*)	36	00:04:30	00:05:43
EM	400	00:37:01	03:54:15
MOGA (ind 60)	10	00:00:11	00:18:09
MOGA (ind 100)	36	00:01:55	00:35:10

8.5 Conclusions

This chapter compared the performance of basic K-Means, semi-supervised K-Means, DBSCAN, EM, and MOGA, to identify encrypted traffic, specifically SSH. Results show that the MOGA obtained better performance in terms of combined *DR*, *FPR*, and computational cost (measured in CPU time). The MOGA also clustered the data with a very small value of K, which is very desirable for a potential implementation of an encrypted traffic detection system.

In this case, MOGA's best individual achieved a detection rate of 93.5% and a false positive rate of 0.7%, whereas it employed only 18 out of the 38 available features and clustered the data in only 36 clusters. The fact that MOGA was able to cluster the data with a much smaller number of clusters, provided a noticeable advantage over the other presented algorithms in terms of the amount of time needed to both building the models and testing/running them on a real life data set. With regards to the semi-supervised model proposed by Erman *et al.* in [22], it can be observed that the weighting sampling method could achieve good results with a very small training data. For future work, it could be interesting to employ MOGA under a semi-supervised context with a weighted sampling method.

Chapter 9

Hierarchical MOGA

9.1 Introduction

Up until this point all the conducted experiments have been based on a flat clustering approach, i.e., data has been clustered at one level of clusters. This chapter describes an adaptation of the MOGA described in chapter 4, to implement a two level hierarchical clustering approach. The purpose of the second layer of clusters is to further partition the data after the original MOGA has separated the data into a first layer of clusters. The motivation behind this analysis is to observe the possible gains, if any, obtained by further increasing cluster purity.

9.2 Methodology

Hierarchical clusters can be either agglomerative or divisive [17]. Hierarchical clusters are agglomerative when the hierarchy is built bottom-up, that is, existing clusters are grouped to form larger clusters based on some criteria of similarity. Divisive hierarchical clusters, on the other hand, are built from the top, dividing the data into more specialized clusters. In this approach, the clusters obtained with the original MOGA proposed in chapter 4, will be divided into a second layer. Thus, this will be a divisive approach, Fig. 9.1.

The first layer of clusters is built by letting the MOGA partition the data as originally explained. Then, based on a *purity threshold*, the resulting clusters are evaluated to identify the ones that need to be further partitioned into smaller clusters. If the purity of a cluster, defined by the number of flows of the majority class in the cluster, divided by the total number of flows in the cluster, is less than the *purity threshold*, then the flows in that cluster need to be reclustered. Reclustering is done by feeding back into the MOGA the flows originally assigned to the partition being reclustered. The MOGA repeats the optimization process, searching for an optimum number of clusters and appropriate subset of features to partition the fed flows. Fig. 9.1 displays a two layered clustering approach, where the number of clusters and employed features on the second layer will vary depending on the data being reclustered.

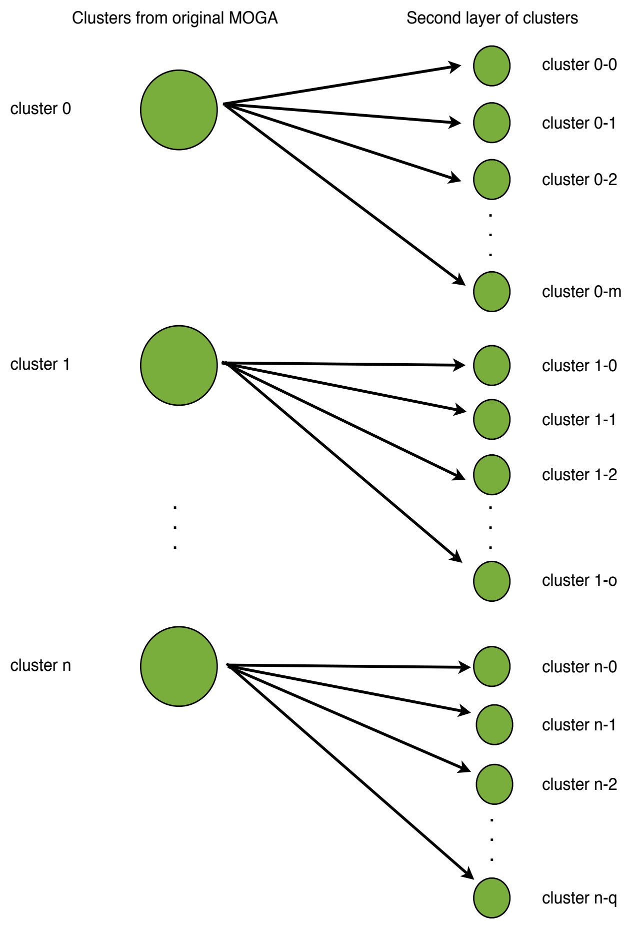

Figure 9.1: Hierarchical Clusters

Aside from the *purity threshold*, there is a second parameter that needs to be estimated to run the hierarchical MOGA. Unlike the original MOGA, where the post-training selection was done once (to select the best individual out of a set of non-dominated candidate solutions), in the hierarchical MOGA this selection needs to be done once after the initial first layer clustering, and then every time a cluster is reclustered. Thus, the post training needs to be automated. This was done by setting the model to always select, out of the set of non-dominated solutions, the individual with the highest detection rate, and whose false positive rate was less than a *false positive threshold*. Both of these parameters, the *purity threshold*, and the *false positive threshold*, were estimated empirically by running the MOGA with several values and observing the parameters that generated the best results. For the *false positive threshold*, good results were obtained with the *false positive threshold* set to 0.05%. For the *purity threshold*, on the other hand, good results were obtained with the *purity threshold* set to 90%.

Once the hierarchical model is trained, the testing process is conducted very similarly as performed with the original MOGA, chapter 4. The only difference is that the labeling of the clusters will now consider a third option. A cluster in the first layer can be labelled as "SSH", "non-SSH", or "Phase 2". If a cluster is labelled as "SSH" or "non-SSH", then flows in the test data that are assigned to any of those clusters are classified like in the original MOGA, with the label of the cluster. If, on the other hand, a flow is assigned to a cluster, c, that has a label "Phase 2", then that flow is redirected to the second layer, where it is assigned to any of the clusters that resulted from repartitioning that data of cluster c. The labels of the clusters in the second layer are (as in the original MOGA), either "SSH", or "non-SSH".

9.3 Results

Twenty five runs of the hierarchical version of MOGA were conducted. Unlike the original MOGA, where the output was a set of non-dominated individuals, in these experiments the result is a unique final solution (individual). The resulting individuals from each of the twenty five conducted runs were plotted in terms of *DR* and *FPR* obtained during the validation phase (test run on the same data used for training), Fig. 9.2. Like in the previous experiments, the idea is to identify the solution that leads to a higher *DR* and a lower *FPR*. The five individuals that achieved the lowest *FPR* were selected as candidate solutions. The selected individuals have a number assigned in Fig. 9.2, which corresponds to their *FPR* order. The post training results of the selected individuals, ordered by their assigned number, plus the number of clusters, and the values in their objective values are displayed on Table 9.1.

The selected individuals are then tested on the entire UCIS data set. Test results are displayed on Table 9.2. Both the training and test data sets are described in

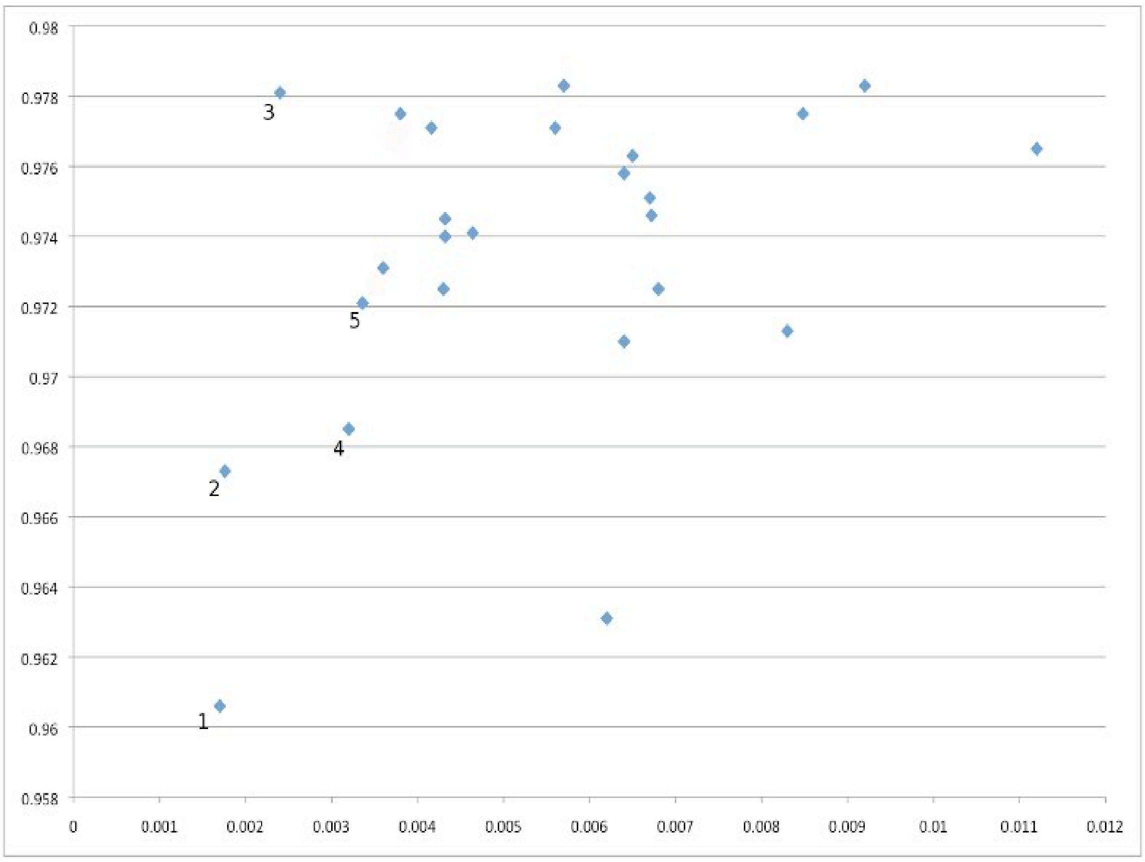

Figure 9.2: Hierarchical MOGA Experiments Post-Training Results

Table 9.1: Hierarchical MOGA Post-Training Results and Objective Values

Ind.	***DR***	***FPR***	***K***	***Fwithin***	***Fbetween***	***Fclusters***	***Fcomplex.***
1	96.1%	0.18%	44	5.12E-06	307.92	0.4815	0.6578
2	96.7%	0.18%	36	1.68E-04	606.51	0.5802	0.6315
3	97.8%	0.24%	43	7.84E-06	379.27	0.4938	0.5
4	96.9%	0.32%	36	1.41E-04	501.89	0.5802	0.6842
5	97.2%	0.33%	41	1.55E-04	615.77	0.5185	0.7368

chapter 5. To asses the effects of increasing cluster purity in the classification performance of the system, these individuals were also tested by building only the first layer of clusters, like in the original MOGA. The columns "DR Hierarchical" and "FPR Hierarchical" represents the results of the hierarchical MOGA, and the columns "DR Flat" and "FPR Flat" represent the results when testing these individuals under the original MOGA. As it can be observed, all the selected individuals obtained gains of about 2% to 5% increase in *DR*, without significant increases in *FPR*. The best

results were obtained with the third individual, which achieved a *DR* of 97.8% and a *FPR* of 0.24% in post training, and a *DR* of 96.3% and a *FPR* of 1.3% when tested on the entire data set. Thus, it can be concluded that increasing cluster purity by means of a hierarchical implementation of the proposed model, does indeed lead to better classification performances.

Table 9.2: Hierarchical MOGA Test Results on UCIS trace

Ind.	***DR* Hierarchical**	***FPR* Hierarchical**	***DR* Flat**	***FPR* Flat**
1	93%	1.3%	91.1%	1.4%
2	94.5%	0.9%	90.5%	0.4%
3	96.3%	1.3%	93.3%	0.6%
4	94.5%	0.8%	90.7%	0.4%
5	95.4%	1.1%	90.5%	0.4%

9.4 Robustness

In the case of hierarchical MOGA, the robustness of the proposed model was also assessed by testing the best individuals on three additional data sets. The best individuals were tested on the AMP trace, on the MAWI [34] trace, and on the DARPA [15] trace. Tables 9.3, 9.4, and 9.5 display the results obtained on the respective data sets. To benchmark the performance of the original MOGA versus the hierarchical MOGA, like in the previous experiments, the best individuals were also tested by building only the first layer of clusters. The columns “DR Hierarchical” and “FPR Hierarchical” represents the results of the hierarchical MOGA, and the columns “DR Flat” and “FPR Flat” represent the results when testing these individuals without employing the second layer of clusters.

In the case of the DARPA trace, Table 9.3, it can be observed that substantial gains were obtained in terms of *DR* with the hierarchical MOGA. The first individual achieved a *DR* of 89.4% with the hierarchical MOGA, compared to the 0% obtained with the original MOGA. The remaining individuals also observed increases in *DR*, but at a smaller scale. In the case of the AMP trace, Table 9.4, there were also gains in terms of *DR*, but for some of the individuals these were quite modest. The fourth individual observed the highest gains, achieving a *DR* of 32% with the hierarchical MOGA, compared to 0.1% obtained with the original MOGA. In the case of the

MAWI trace, Table 9.5, the best individual was the fourth individual as well. The fourth individual achieved a *DR* of 14% with the hierarchical MOGA, compared to the 2.1% obtained with the original MOGA.

Table 9.3: Hierarchical MOGA Test Results on DARPA trace

Ind.	***DR* Hierarchical**	***FPR* Hierarchical**	***DR* Flat**	***FPR* Flat**
1	89.4%	1.7%	0%	0%
2	33.8%	0.8%	0%	0%
3	27.9%	0.5%	27.9%	0.4%
4	10.3%	0.3%	0%	0%
5	0%	0.1%	0%	0%

Table 9.4: Hierarchical MOGA Test Results on AMP trace

Ind.	***DR* Hierarchical**	***FPR* Hierarchical**	***DR* Flat**	***FPR* Flat**
1	1.8%	0.1%	0.6%	0.9%
2	0.4%	1.3%	0.1%	0.2%
3	1.5%	1%	0.7%	0.2%
4	32%	1.1%	0.1%	0.2%
5	9.5%	1.4%	0.4%	0.1%

Table 9.5: Hierarchical MOGA Test Results on MAWI trace

Ind.	***DR* Hierarchical**	***FPR* Hierarchical**	***DR* Flat**	***FPR* Flat**
1	2.2%	0.7%	2.4%	0.7%
2	1.9%	1.0%	1.9%	0.1%
3	2.2%	0.5%	2.2%	0.2%
4	14%	0.7%	2.1%	0%
5	3.2%	1%	3%	0%

From these results it can be observed that some individuals observed significant gains with the hierarchical MOGA in terms of robustness. However, with the exception of the first individual in the DARPA trace, the classification results were still quite low. Thus, it would be interesting to identify which factors contribute to certain individuals achieving better results than others in terms of robustness. For these experiments in particular, the first four individuals achieved the strongest results. Table 9.1 displayed the objective values of the best individuals, as well as the number of employed clusters. By inspecting this table, it can be observed that the first individual, which was the best individual in the DARPA trace, employed a slightly higher number of clusters, 44, but with no particular higher values in the other objectives. It is important to note, however, that the first individual did not achieve good results on the AMP and MAWI traces. The second and fourth individuals have strong values in *Fwithin* and *Fbetween*, and the third individual has a slightly strong value in *Fbetween*. Thus, these preliminary observations do not seem to provide much information regarding which factor contributed to achieving more robust results. However, the fact that significant gains were observed in some of the individuals should encourage more research in the area, as it seems feasible to achieve robust solutions with the proposed model.

The ideal final individual should be able to classify traffic in all the tested traces, which did not happen in this case. The individuals that observed gains in some of the traces were not the same ones that perform well on the other traces. There are several things that could be attempted on this regard. The existing training data could be modified in terms of both, size and composition. It could be the case that the SSH behaviors captured in the training data do not represent the behaviors in the other traces. Also, given that considerable gains in performance were observed by means of a second layer of clusters, it would be interesting to observe the effects of implementing additional layers of clusters. This, combined with larger training data sets could potentially lead to better results. Finally, it could be intuitive to think that the use of time related features might have a detrimental effect on the robustness of the system, as packet arrival times could vary from network to network. While this could be a valid argument, it should be noted that in [5] and [6] the authors achieved good results in terms of robustness employing both packet length and packet inter-arrival time related features with a supervised machine learning approach. While the model proposed in this work is an unsupervised machine learning technique, the progress achieved by the authors of that work could indicate that robustness can be achieved with a mix of time and packet length related features.

9.5 Conclusions

This chapter described the implementation of a hierarchical version of the proposed MOGA. The hierarchical version increased the resulting cluster purity by further partitioning the clusters generated with the original MOGA. The resulting two level hierarchical clusters lead to an increase in performance in terms of *DR* and *FPR* with respect to the results obtained with the original MOGA. The best individual achieved a *DR* of 96.3% and a *FPR* 1.3% when tested on the entire UCIS data set. This chapter also assessed the robustness of the proposed hierarchical model by testing the best individuals in three additional data sets, DARPA, AMP, and MAWI. While some of the tested individuals observed a notorious increase in classification performances on the different data sets, the obtained *DR* were still low. However, the fact that significant gains were observed with the hierarchical model in terms of robustness encourages further research in the area.

Chapter 10

Conclusions

This work has investigated the application of a Multi-Objective Genetic Algorithm (MOGA) to the problem of encrypted traffic identification, in particular, SSH. The proposed model was employed for the dual problem of feature selection and cluster count optimization under a flow based representation. The proposed MOGA employed an unsupervised machine learning technique, K-Means, as the clustering algorithm. The main motivation behind the use of an unsupervised learning technique for traffic classification was the limitations of both of the traditionally employed techniques, namely port number and payload inspection, when faced with encrypted traffic. The use of unsupervised techniques, i.e., purely cluster/data driven approaches to separate network traffic into classes, has already been demonstrated by many of the authors reviewed in chapter 3. The main contribution of this work to the field of network classification comes, however, from further increasing the performance of those methods by incorporating a genetic algorithm that can explore the solution space, searching for an optimum combination between an appropriate subset of features and the right number of clusters, as well as specifically focusing on encrypted traffic identification.

The first hypothesis of this work was whether the MOGA was capable of selecting an appropriate subset of features to partition the data into encrypted versus non encrypted traffic. Chapter 7 addressed this hypothesis by benchmarking the performance of the MOGA against a gold standard feature selection model for traffic classification. The proposed MOGA selected a subset of the features selected by the gold standard model, which indicates that the clustering objectives guide, to a certain extend, the learning of the system towards the ultimate goal of traffic classification. Furthermore, because the MOGA selected similar features with the gold standard model, the classification performance of the proposed model was equivalent to that of the gold standard. Chapter 7 also evaluated the effects of applying a logarithmic normalization to the data prior to running the MOGA. With the normalized data, it could be observed that the MOGA did not select similar features as the gold standard model, consequently, not being able to achieve an equivalent performance.

The second hypothesis of this work was whether it was possible to obtain gains in both computational performance and classification rates with the proposed model. In chapter 8, the performance of the proposed MOGA was benchmarked against the basic K-Means, but without the MOGA optimization. Furthermore, the performance of the proposed model was also benchmarked against other clustering techniques existing in the literature, such as semi-supervised K-Means, DBSCAN, and EM. Results showed that MOGA not only outperformed the other models in terms of classification performance, but also in terms of the required time to build and to test the model.

The last hypothesis that was investigated had to do with the potential gains in classification performances by increasing cluster purity. In chapter 9, a hierarchical version of MOGA was implemented, which is based on a divisive hierarchical approach, aimed at increasing the purity of the cluster obtained with the original MOGA. The performance of the hierarchical model was then benchmarked against the original MOGA. Results show that the hierarchical version of MOGA, as expected, had a better classification performance.

In summary, the proposed method showed promising results in terms of classification performance, and also in terms of computational costs. The latter attributed to the use of less clusters and less features, the former attributed to the higher inter and intra-cluster distances. It is important to note that the savings in computational cost move unsupervised approaches one step closer to online traffic classification. Another interesting contribution was the benchmarking of the proposed model against a gold standard model for feature selection. Results gave a good indication that this purely data driven approach can somewhat mimic the behavior of a label driven model. This, it is believed, should leave the door open to further attempts in this area.

With regards to the future work in terms of the data representation, the experiments conducted in chapter 7 investigated the effects of a prior logarithmic transformation to the data. From these experiments it could be observed that under a normalized data representation, it was not possible to mimic the behavior of the gold standard model. Moreover, the results were considerably inferior when compared to the results obtained without applying the logarithmic normalization. Still, it could be observed from the non-normalized experiments that several features, specifically features with higher variances in their values, were constantly ignored by the MOGA. There is no reason to believe those features could not potentially contribute to a better class discrimination. Thus, it would be interesting to search for an appropriate normalization function that allows the proper inclusion of all the available features, while at the same time mimicking the behavior of the gold standard model.

With regards to the training data, the sampling of the trace to generate a training data was based on the work of [5]. This training data was kept constant throughout the development of this work, so that it would be possible to compare the performances of the different attempts. Still, more work could be done, investigating potentially better training samples. In particular, training data sets that include the applications that are more commonly misclassified, specially in the robustness experiments. Also with regards to the training data, it would be interesting to incorporate the work presented in [22], where the authors achieved good classification performances with a smaller training data. Such training data is generated with a percentage of the flows from above the 95th percentile of the traffic size. This approach was implemented in chapter 8, leading to very good results. A smaller training data set would more than likely decrease the building times for the MOGA, which would be very desirable.

With regards to the methodology employed for this work, several variations could potentially lead to better results, from larger initial populations, to a larger number of epochs. Furthermore, the genetic algorithm itself could be replaced with a different generational algorithm, or with a variation of the proposed MOGA. In particular, it would be interesting to introduce some kind of elitist mechanism to further bias the evolution of the objectives. The proposed MOGA assigns fitness based on a roulette wheel that is biased towards selecting non-dominated individuals. However, arguably there is still room for further guidance of the evolution of the individuals without necessarily falling into a hill climbing approach. With regards to the employed objective values, in particular the *Fwithin* and *Fbetween*, these objectives could be further analyzed to explore alternative objective functions that lead to a data description that more closely resembles existing classes in the data. Finally, with regards of the unsupervised algorithm optimized by the MOGA, a basic K-Means algorithm, this model could have also employed other unsupervised algorithms, such as semi-supervised K-Means, DBSCAN, or EM.

With regards to the robustness of the proposed model evaluated in chapter 9, it could be observed that the hierarchical MOGA considerably increased the performance of the system when tested on additional data sets. However, for the majority of the experiments the obtained classification results were still low. More work is needed in this regard, which could consider larger and different training data sets, more layers for the hierarchical approach, and/or different objective values.

Finally, in this work SSH was chosen as an example of encrypted traffic identification. This same approach could also be used with other types of encrypted traffic, such as SSL, or Skype. Moreover, it is also of interest to identify applications within an encrypted tunnel, or to identify malicious behaviors hidden in the tunnel.

Bibliography

[1] Ethem Alpaydin. *Introduction to Machine Learning.* MIT Press, 2004.

[2] R Alshammari. Automatically Classifying Encrypted Network Traffic: A Case Study of SSH. Master's thesis, Dalhousie University, Halifax, Nova Scotia, Canada, 2008.

[3] R. Alshammari and A.N. Zincir-Heywood. A flow based approach for ssh traffic detection. In *Systems, Man and Cybernetics, IEEE International Conference on*, pages 296–301, Oct. 2007.

[4] R. Alshammari and A.N. Zincir-Heywood. Investigating two different approaches for encrypted traffic classification. In *Privacy, Security and Trust, 2008. PST '08. Sixth Annual Conference on*, pages 156–166, Oct. 2008.

[5] R. Alshammari and A.N. Zincir-Heywood. Generalization of signatures for ssh encrypted traffic identification. In *IEEE Symposium Series on Computational Intelligence on Cyber Security*, 2009.

[6] R. Alshammari and A.N. Zincir-Heywood. Machine learning based encrypted traffic classification: identifying SSH and skype. In *Proceedings of the Second IEEE international conference on Computational intelligence for security and defense applications*, pages 289–296. IEEE Press, 2009.

[7] C. Bacquet, K. Gumus, D. Tizer, A.N. Zincir-Heywood, and M.I. Heywood. A comparison of unsupervised learning techniques for encrypted traffic identification. *Journal of Information Assurance and Security*, 5:464–472, 2010.

[8] C. Bacquet, A.N. Zincir-Heywood, and M.I. Heywood. An Investigation of Multi-objective Genetic Algorithms for Encrypted Traffic Identification. In *Computational Intelligence in Security for Information Systems: Cisis' 09, 2nd International Workshop Burgos, Spain*, pages 93–100. Springer, Sep. 2009.

[9] C. Bacquet, A.N. Zincir-Heywood, and M.I. Heywood. An analysis of clustering objectives for feature selection applied to encrypted trafc identication. In *Proceedings of IEEE World Congress on Computational Intelligence, Barcelona, Spain*, July. 2010.

[10] D.J. Barrett, R.E. Silverman, and R.G. Byrnes. *SSH, the secure shell: the definitive guide.* O'Reilly Media, Inc., 2005.

[11] Laurent Bernaille and Renata Teixeira. Early recognition of encrypted applications. In *Proceedings of the Eighth Passive and Active Measurement Conference (PAM07)*, 2007.

[12] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. Traffic classification on the fly. *SIGCOMM Comput. Commun. Rev.*, 36(2):23–26, 2006.

[13] Laurent Bernaille, Renata Teixeira, and Kave Salamatian. Early application identification. In *CoNEXT '06: Proceedings of the 2006 ACM CoNEXT conference*, pages 1–12, New York, NY, USA, 2006. ACM.

[14] P.S. Bradley, U. Fayyad, and C. Reina. Scaling EM (expectation-maximization) clustering to large databases. *Microsoft Research Report, MSR-TR-98-35*, 1998.

[15] DARPA. http://www.ll.mit.edu/ist/ideval/docs/1999/schedule.html.

[16] A.P. Dempster, N.M. Laird, D.B. Rubin, et al. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[17] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* Wiley, New York, 2. edition, 2001.

[18] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli. Detection of encrypted tunnels across network boundaries. In *Proceedings of the 43rd IEEE International Conference on Communications (ICC 2008), Beijing, China*, 2008.

[19] J.G. Dy and C.E. Brodley. Feature selection for unsupervised learning. *The Journal of Machine Learning Research*, 5:889, 2004.

[20] J. Erman, M. Arlitt, and A. Mahanti. Traffic classification using clustering algorithms. In *Proceedings of the SIGCOMM workshop on Mining network data*, pages 281–286. ACM, 2006.

[21] J. Erman, A. Mahanti, and M. Arlitt. Internet Traffic Identification using Machine Learning. In *Global Telecommunications Conference. GLOBECOM '06.*, pages 1–6. IEEE, 2006.

[22] J. Erman, A. Mahanti, M. Arlitt, I. Cohen, and C. Williamson. Offline/realtime traffic classification using semi-supervised learning. *Perform. Eval.*, 64(9-12):1194–1213, 2007.

[23] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson. Identifying and discriminating between web and peer-to-peer traffic in the network core. In *Proceedings of the 16th international conference on World Wide Web*, pages 883–892. ACM, 2007.

[24] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. KDD*, volume 96, pages 226–231, 1996.

[25] J.H. Holland. Genetic Algorithms. *Scientific American*, 267(1):66–72, 1992.

[26] IANA. http://www.iana.org/assignments/port-numbers.

[27] IETF. http://www.ietf.org/.

[28] GPS Junior, JEB Maia, R. Holanda, and JN De Sousa. P2P Traffic Identification using Cluster Analysis. In *Global Information Infrastructure Symposium. GIIS 2007. First International*, pages 128–133, 2007.

[29] T. Karagiannis, A. Broido, N. Brownlee, KC Claffy, and M. Faloutsos. Is p2p dying or just hiding?[p2p traffic measurement]. In *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04*, volume 3.

[30] YeongSeog Kim, W. Nick Street, and Filippo Menczer. Feature selection in unsupervised learning via evolutionary search. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 365–369, New York, NY, USA, 2000. ACM.

[31] Rajeev Kumar and Peter Rockett. Improved sampling of the pareto-front in multiobjective genetic optimizations by steady-state evolution: a pareto converging genetic algorithm. *Evol. Comput.*, 10(3):283–314, 2002.

[32] J.B. MacQueen et al. Some methods for classification and analysis of multivariate observations, 1966.

[33] G. Maiolini, G. Molina, A. Baiocchi, and A. Rizzi. On the fly Application Flows Identification by exploiting K-Means based classifiers. *Journal of Information Assurance and Security*, 4:142–150, 2009.

[34] MAWI. http://tracer.csl.sony.co.jp/mawi/.

[35] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow clustering using machine learning techniques. *Lecture Notes in Computer Science*, 3015:205–214, 2004.

[36] A.R. McIntyre. *Novelty detection+ coevolution= automatic problem decomposition: A framework for scalable Genetic Programming classifiers.* PhD thesis, Dalhousie University, Halifax, Canada, 2008.

[37] M. Mitchell. *An introduction to genetic algorithms.* MIT Press Cambridge, MA, USA, 1996.

[38] NetMate. http://www.ip-measurement.org/tools/netmate.

[39] T.T.T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *Communications Surveys Tutorials, IEEE*, 10(4):56 –76, quarter 2008.

[40] PacketShaper. http://www.packeteer.com/products/packetshaper.

[41] V. Paxson. Empirically derived analytic models of wide-area TCP connections. *IEEE/ACM Transactions on Networking (TON)*, 2(4):316–336, 1994.

[42] WEKA. http://www.cs.waikato.ac.nz/ml/weka/.

[43] Caihong Yang, Fei Wang, and Benxiong Huang. Internet traffic classification using dbscan. *Information Engineering, International Conference on*, 2:163–166, 2009.

[44] Liu Yingqiu, Li Wei, and Li Yunchun. Network traffic classification using k-means clustering. In *Computer and Computational Sciences, 2007. IMSCCS 2007. Second International Multi-Symposiums on*, pages 360–365, Aug. 2007.

[45] S. Zander, T. Nguyen, and G. Armitage. Automated traffic classification and application identification using machine learning. In *Conference on Local Computer Networks*, volume 30, pages 250–257. IEEE, 2005.