Detecting Anomalies in VoIP traffic using Principal Components Analysis

Francesco Sbolci

Table of contents

1. Summary

1.1 Anomaly Detection using Principal Components Analysis

The idea of using a method based on Principal Components Analysis to detect anomalies in network's traffic was first introduced by A. Lakina, M. Crovella and C. Diot in an article published in 2004 called "Diagnosing Network-Wide Traffic Anomalies" [1].
They proposed a general method to diagnose traffic anomalies, using PCA to effectively separate the high-dimensional space occupied by a set of network traffic measurements into disjoint subspaces corresponding to normal and anomalous network conditions.
This algorithm was tested in subsequent works, taking into consideration different characteristics of IP traffic over a network (such as byte counts, packet counts, IP-flow counts, etc...) [2].
The proposal of using entropy as a summarization tool inside the algorithm led to significant advances in terms or possibility of analyzing massive data sources [3]; but this type of AD method still lacked the possibility of recognizing the users responsible of the anomalies detected.
This last step was obtained using random aggregations of the IP flows, by means of sketches [4], leading to better performances in the detection of anomalies and to the possibility of identifying the responsible IP flows.
This version of the algorithm has been implemented by C. Callegari and L. Gazzarini, in Universitá di Pisa, in an AD software, described in [5], for analyzing IP traffic traces and detecting anomalies in them.
Our work consisted in adapting this software (designed for working with IP traffic traces) for using it with VoIP Call Data Records, in order to test its applicability as an Anomaly Detection system for voice traffic.
We then used our modified version of the software to scan a real VoIP traffic trace, obtained by a telephonic operator, in order to analyze the software's performances in a real environment situation. We used two different types of analysis on the same traffic trace, in order to understand software's features and limits, other than its possibility of application in AD problematics.
As we discovered that the software's performances are heavily dependent on the input parameters used in the analysis, we concluded with several tests performed using artificially created anomalies, in order to understand the relationships between each input parameter's value and the software's capability of detecting different types of anomalies.
The different analysis performed, in the ending, led us to some considerations upon the possibility of applying this PCA's based software as an Anomaly Detector in VoIP environments.

At the best of our knowledge this is the first time a technique based on Principal Components Analysis is used to detect anomalous users in VoIP traffic; in more detail our contribution consisted in:

- Creating a version of an AD software based on PCA that could be used on VoIP traffic traces
- Testing the software's performances on a real traffic trace, obtained by a telephonic operator
- From the first tests, analyzing the appropriate parameters' values that permitted us to obtain results that could be useful for detecting anomalous users in a VoIP environment

- Observing the types of users detected using the software on this trace and classify them, according to their behavior during the whole duration of the trace
- Analyzing how the parameters' choice impact the type of detections obtained from the analysis and testing which are the best choices for detecting each type of anomalous users
- Proposing a new kind of application of the software that avoids the biggest limitation of the first type of analysis (that we will see that is the impossibility of detecting more than one anomalous user per time-bin)
- Testing the software's performances with this new type of analysis, observing also how this different type of application impacts the results' dependence from the input parameters
- Comparing the software's ability of detecting anomalous users with another type of AD software that works on the same type of trace (VoIP SEAL)
- Modifying the trace in order to obtain, from the real trace, a version cleaned from all the detectable anomalies, in order to add in that trace artificial anomalies
- Testing the software's performances in detecting different types of artificial anomalies
- Analyzing in more detail the software's sensibility from the input parameters, when used for detecting artificially created anomalies
- Comparing results and observations obtained from these different types of analysis to derive a global analysis of the characteristics of an Anomaly Detector based on Principal Components Analysis, its values and its lacks when applying it on a VoIP trace

The structure of our work is the following:

1. We'll start describing the state of the art in AD techniques
2. We will continue analyzing the PCA theory, describing the structure of the algorithm used in our software, his features and the type of data it needs to be used as an Anomaly Detection system for VoIP traffic.
3. Then, after shortly describing the type of trace we used to test our software, we will introduce the first type of analysis performed, the single round analysis, pointing out the results obtained and their dependence from the parameters' values.
4. In the following section we will focus on a different type of analysis, the multiple round analysis, that we introduced to test the software's performances, removing its biggest limitation (the impossibility of detecting more than one user per time-bin); we will describe the results obtained, comparing them with the ones obtained with the single round analysis, check their dependence from the parameters and compare the performances with the ones obtained using another type of AD software (VoIP SEAL) on the same trace.
5. We will then consider the results and observations obtained testing our software using artificial anomalies added on a “cleaned” version of our original trace (in which we removed all the anomalous users detectable with our software), comparing the software's performances in detecting different types of anomalies and analyzing in detail their dependence from the parameters' values.
6. At last we will describe our conclusions, derived using all the observations obtained with different types of analysis, about the applicability of a software based on PCA as an Anomaly Detector in a VoIP environment.

2. State of the art

2.1 Introduction

As advances in networking technology help to connect the distant corners of the globe and as the Internet continues to expand its influence as a medium for communications and commerce, the threat from spammers, attackers and criminal enterprises has also grown accordingly [6]. It is the prevalence of such threats that has made intrusion detection systems join ranks with firewalls as one of the fundamental technologies for network security.
An intrusion detection system gathers and analyzes information from various areas within a computer or a network to identify possible security breaches. In other words, intrusion detection is the act of detecting actions that attempt to compromise the confidentiality, integrity or availability of a system/network.
Traditionally, intrusion detection systems have been classified as a signature detection system, an anomaly detection system or a hybrid/compound detection system. A signature detection system identifies patterns of traffic or application data presumed to be malicious while anomaly detection systems compare activities against a “normal” baseline. On the other hand, a hybrid intrusion detection system combines the techniques of the two approaches. Both signature detection and anomaly detection systems have their share of advantages and drawbacks. The primary advantage of signature detection is that known attacks can be detected fairly reliably with a low false positive rate. The major drawback of the signature detection approach is that such systems typically require a signature to be defined for all of the possible attacks that an attacker may launch against a network. Anomaly detection systems have two major advantages over signature based intrusion detection systems. The first advantage that differentiates anomaly detection systems from signature detection systems is their ability to detect unknown attacks as well as “zero day” attacks. This advantage is because of the ability of anomaly detection systems to model the normal operation of a system/network and detect deviations from them. A second advantage of anomaly detection systems is that the aforementioned profiles of normal activity are customized for every system, application and/or network, and therefore making it very difficult for an attacker to know with certainty what activities it can carry out without getting detected. However, the anomaly detection approach has its share of drawbacks as well. For example, the intrinsic complexity of the system, the high percentage of false alarms and the associated difficulty of determining which specific event triggered those alarms are some of the many technical challenges that need to be addressed before anomaly detection systems can be widely adopted.

2.2 Intrusion detection

An intrusion detection system is a software tool used to detect unauthorized access to a computer system or network. An intrusion detection system is capable of detecting all types of malicious network traffic and computer usage. This includes network attacks against vulnerable services, data driven attacks on
applications, host-based attacks -such as privilege escalation, unauthorized logins and access to sensitive files- and malware. An intrusion detection system is a dynamic monitoring entity that complements the static monitoring abilities of a firewall. An intrusion detection system monitors traffic in a network in promiscuous mode, very much like a network sniffer. The network packets that are collected are analyzed for rule violations by a pattern recognition algorithm. When rule violations are detected, the intrusion detection system alerts the administrator.
One of the earliest work that proposed intrusion detection by identifying abnormal behavior can be attributed to Anderson [7]. In his report, Anderson presents a threat model that classifies threats as external penetrations, internal penetrations, and misfeasance, and uses this classification to develop a security monitoring surveillance system based on detecting anomalies in user behavior. External penetrations are defined as intrusions that are carried out by unauthorized computer system users; internal penetrations are those that are carried out by authorized users who are not authorized for the data that is compromised; and misfeasance is defined as the misuse of authorized access both to the system and to its data.
In a seminar paper, Denning [8] puts forth the idea that intrusions to computers and networks could be detected by assuming that users of a computer/network would behave in a manner that enables automatic profiling. In other words, a model of the behavior of the entity being monitored could be constructed by an intrusion detection system, and subsequent behavior of the entity could be verified against the entity's model. In this model, behavior that deviates sufficiently from the norm is considered anomalous. In the paper, Denning mentioned several models that are based on statistics, Markov chains, time-series etc.
In a much cited survey on intrusion detection systems, Axelsson [9] put forth a generalized model of a typical intrusion detection system. According to Axelsson, the generic architectural model of an intrusion detection system contains the following modules:

- Audit data collection: This module is used in the data collection phase. The data collected in this phase is analyzed by the intrusion detection algorithm to find traces of suspicious activity. The source of the data can be host/network activity logs, command-based logs, application-based logs, etc.
- Audit data storage: Typical intrusion detection systems store the audit data either indefinitely or for a sufficiently long time for later reference. The volume of data is often exceedingly large. Hence, the problem of audit data reduction is a major research issue in the design of intrusion detection systems.
- Analysis and detection: The processing block is the heart of an intrusion detection system. It is here that the algorithms to detect suspicious activities are implemented. Algorithms for the analysis and detection of intrusions have been traditionally classified into three broad categories: signature (or misuse) detection, anomaly detection and hybrid (or compound)

detection.

- Configuration data: The configuration data is the most sensitive part of an intrusion detection system. It contains information that is pertinent to the operation of the intrusion detection system itself such as information on how and when to collect audit data, how to respond to intrusions, etc.
- Reference data: The reference data storage module stores information about known intrusion signatures (in the case of signature detection) or profiles of normal behavior (in the case of anomaly detection). In the latter case, the profiles are updated when new knowledge about system behavior is available.
- Active/processing data: The processing element must frequently store intermediate results such as information about partially fulfilled intrusion signatures.
- Alarm: This part of the system handles all output from the intrusion detection system. The output may be either an automated response to an intrusion or a suspicious activity alert for a system security officer.

As mentioned above, algorithms for the analysis and detection of intrusions/attacks are traditionally classified into the following three broad categories:

- *Signature or misuse detection* is a technique for intrusion detection that relies on a predefined set of attack signatures. By looking for specific patterns, the signature detection-based intrusion detection systems match incoming packets and/or command sequences to the signatures of known attacks. In other words, decisions are made based on the knowledge acquired from the model of the intrusive process and the observed trace that it has left in the system. Legal or illegal behavior can be defined and compared with observed behavior. Such a system tries to collect evidence of intrusive activity irrespective of the normal behavior of the system. One of the chief benefits of using signature detection is that known attacks can be detected reliably with a low false positive rate. The existence of specific attack sequences ensures that it is easy for the system administrator to determine exactly which attacks the system is currently experiencing. If the audit data in the log files do not contain the attack signature, no alarm is raised. Another benefit is that the signature detection system begins protecting the computer/network immediately upon installation. One of the biggest problems with signature detection systems is maintaining state information of signatures in which an intrusive activity spans multiple discrete events -that is, the complete attack signature spans multiple packets. Another drawback is that the signature detection system must have a signature defined for all of the possible attacks that an attacker may launch. This requires frequent signature updates to keep the signature database up-to-date.
- An *anomaly detection system* first creates a baseline profile of the normal system, network, or program activity. Thereafter, any activity that deviates from the baseline is treated as a possible intrusion. Anomaly detection systems offer several benefits. First, they have the capability to detect insider attacks. For instance, if a user or someone using a stolen account starts performing actions that are outside the normal user-profile, an anomaly detection system generates an alarm. Second, because the system is based on customized profiles, it is very difficult for an attacker to know with certainty what activity it can carry out without setting off

an alarm. Third, an anomaly detection system has the ability to detect previously unknown attacks. This is due to the fact that a profile of intrusive activity is not based on specific signatures representing known intrusive activity. An intrusive activity generates an alarm because it deviates from normal activity, not because someone configured the system to look for a specific attack signature. Anomaly detection systems, however, also suffer from several drawbacks. The first obvious drawback is that the system must go through a training period in which appropriate user profiles are created by defining "normal" traffic profiles. Moreover, creating a normal traffic profile is a challenging task. The creation of an inappropriate normal traffic profile can lead to poor performance. Maintenance of the profiles can also be time-consuming. Since, anomaly detection systems are looking for anomalous events rather than attacks, they are prone to be affected by time consuming false alarms. False alarms are classified as either being false positive or false negative. A false positive occurs when an IDS reports as an intrusion an event that is in fact legitimate network activity. A side affect of false positives, is that an attack or malicious activity on the network/system could go undetected because of all the previous false positives. This failure to detect an attack is termed as a false negative in the intrusion detection jargon. A key element of modern anomaly detection systems is the alert correlation module. However, the high percentage false alarms that are typically generated in anomaly detection systems make it very difficult to associate specific alarms with the events that triggered them. Lastly, a pitfall of anomaly detection systems is that a malicious user can train an anomaly detection system gradually to accept malicious behavior as normal.

- A *hybrid* or *compound detection system* combines both approaches. In essence, a hybrid detection system is a signature inspired intrusion detection system that makes a decision using a "hybrid model" that is based on both the normal behavior of the system and the intrusive behavior of the intruders.

2.3 Premise of anomaly detection

An anomaly detection approach usually consists of two phases: a training phase and a testing phase. In the former, the normal traffic profile is defined; in the latter, the learned profile is applied to new data. The central premise of anomaly detection is that intrusive activity is a subset of anomalous activity [10]. If we consider an intruder, who has no idea of the legitimate user's activity patterns, intruding into a host system, there is a strong probability that the intruder's activity will be detected as anomalous. In the ideal case, the set of anomalous activities will be the same as the set of intrusive activities. In such a case, flagging all anomalous activities as intrusive activities results in no false positives and no false negatives. However, intrusive activity does not always coincide with anomalous activity. Kumar and Stafford [10] suggested that there are four possibilities, each with a non-zero probability:

- Intrusive but not anomalous: These are false negatives. An intrusion detection system fails to detect this type of activity as the activity is not anomalous. These are called false negatives because the intrusion detection system falsely reports the absence of intrusions.
- Not intrusive but anomalous: These are false positives. In other words, the activity is not intrusive, but because it is anomalous, an intrusion detection system reports it as intrusive. These are called false positives because an intrusion detection system falsely reports intrusions.
- Not intrusive and not anomalous: These are true negatives; the activity is not intrusive and is not reported as intrusive.
- Intrusive and anomalous: These are true positives; the activity is intrusive and is reported as such.

When false negatives need to be minimized, thresholds that define an anomaly are set low. This results in many false positives and reduces the efficacy of automated mechanisms for intrusion detection. It creates additional burdens for the security administrator as well, who must investigate each incident and discard false positive instances.
In the following subsection, we review a number of different architectures and methods that have been proposed for anomaly detection. These include statistical anomaly detection, data-mining based methods, and machine learning based techniques.

2.4 Statistical Anomaly Detection

In statistical methods for anomaly detection, the system observes the activity of subjects and generates profiles to represent their behavior. The profile typically includes such measures as activity intensity measure, audit record distribution measure, categorical measures (the distribution of an activity over categories) and ordinal measure (such as CPU usage). Typically, two profiles are maintained for each subject: the current profile and the stored profile. As the system/network events (viz. audit log records, incoming packets etc.) are processed, the intrusion detection system updates the current profile and periodically calculates an anomaly score (indicating the degree of irregularity for the specific event) by comparing the current profile with the stored profile using a function of abnormality of all measures within the profile. If the anomaly score is higher than a certain threshold, the intrusion detection system

generates an alert.

Statistical approaches to anomaly detection have a number of advantages. Firstly, these systems, like most anomaly detection systems, do not require prior knowledge of security flaws and/or the attacks themselves. As a result, such systems have the capability of detecting “zero day” or the very latest attacks. In addition, statistical approaches can provide accurate notification of malicious activities that typically occur over extended periods of time and are good indicators of impending denial-of-service attacks. A very common example of such an activity is a portscan. Typically, the distribution of portscans is highly anomalous in comparison to the usual traffic distribution. This is particularly true when a packet has unusual features (e.g., a crafted packet). With this in mind, even portscans that are distributed over a lengthy time frame will be recorded because they will be inherently anomalous.

However, statistical anomaly detection schemes also have drawbacks. Skilled attackers can train a statistical anomaly detection to accept abnormal behavior as normal. It can also be difficult to determine thresholds that balance the likelihood of false positives with the likelihood of false negatives. In addition, statistical methods need accurate statistical distributions, but, not all behaviors can be modeled using purely statistical methods. In fact, a majority of the proposed statistical anomaly detection techniques require the assumption of a quasi-stationary process, which cannot be assumed for most data processed by anomaly detection systems.

Haystack [11] is one of the earliest examples of a statistical anomaly-based intrusion detection system. It used both user and group-based anomaly detection strategies, and modeled system parameters as independent, Gaussian random variables. Haystack defined a range of values that were considered normal for each feature. If during a session, a feature fell outside the normal range, the score for the subject was raised. Assuming the features were independent, the probability distribution of the scores was calculated. An alarm was raised if the score was too large. Haystack also maintained a database of user groups and individual profiles. If a user had not previously been detected, a new user profile with minimal capabilities was created using restrictions based on the user’s group membership. It was designed to detect six types of intrusions: attempted break-ins by unauthorized users, masquerade attacks, penetration of the security control system, leakage, DoS attacks and malicious use. One drawback of Haystack was that it was designed to work offline. The attempt to use statistical analyses for real-time intrusion ystems failed, since doing so required high-performance systems. Secondly, because of its dependence on maintaining profiles, a common problem for system administrators was the determination of what attributes were good indicators of intrusive activity.

One of the earliest intrusion detection systems was developed at the Stanford Research Institute (SRI) in the early 1980’s and was called the Intrusion Detection Expert System (IDES) [12, 13]. IDES was a system that continuously monitored user behavior and detected suspicious events as they occurred. In IDES, intrusions could be flagged by detecting departures from established normal behavior patterns for individual users. As the analysis methodologies developed for IDES matured, scientists at SRI developed an improved version of IDES called the Next-Generation Intrusion Detection Expert System (NIDES) [14,15, 16]. NIDES was one of the few intrusion detection systems of its generation that could operate in real time for continuous monitoring of user activity or could run in a batch mode for periodic analysis of the audit data. However, the primary mode of operation of NIDES was to run in real-time. Unlike IDES, which is an anomaly detection system, NIDES is a hybrid system that has an upgraded statistical analysis engine. In both IDES and NIDES, a profile of normal behavior based on a selected set of variables is maintained by the statistical analysis unit. This enables the system to compare the

current activity of the user/system/network with the expected values of the audited intrusion detection variables stored in the profile and then flag an anomaly if the audited activity is sufficiently far from the expected behavior. Each variable in the stored profile reflects the extent to which a particular type of behavior is similar to the profile built for it under "normal conditions". The way that this is computed is by associating each measure/variable to a corresponding random variable. The frequency distribution is built and updated over time, as more audit records are analyzed. It is computed as an exponential weighted sum with a half- life of 30 days. This implies that the half-life value makes audit records that were gathered 30 days in the past to contribute with half as much weight as recent records; those gathered 60 days in the past contribute one–quarter as much weight, and so on. The frequency distribution is kept in the form of a histogram with probabilities associated with each one of the possible ranges that the variable can take. The cumulative frequency distribution is then built by using the ordered set of bin probabilities. Using this frequency distribution, and the value of the corresponding measure for the current audit record, it is possible to compute a value that reflects how far away from the "normal" value of the measure the current value is. The actual computation in NIDES [16] renders a value that is correlated with how abnormal this measure is. Combining the values obtained for each measure and taking into consideration the correlation between measures, the unit computes an index of how far the current audit record is from the normal state. Records beyond a threshold are flagged as possible intrusions.

However the techniques used in [12-16] have several drawbacks. Firstly, the techniques are sensitive to the normality assumption. If data on a measure are not normally distributed, the techniques would yield a high false alarm rate. Secondly, the techniques are predominantly univariate in that a statistical norm profile is built for only one measure of the activities in a system. However, intrusions often affect multiple measures of activities collectively.

SPADE (Statistical Packet Anomaly Detection Engine) [17] is a statistical anomaly detection system that is available as a plug-in for SNORT [18], and is can be used for automatic detection of stealthy port scans. SPADE was one of the first papers that proposed using the concept of an anomaly score to detect port scans, instead of using the traditional approach of looking at p attempts over q seconds. In [17], the authors used a simple frequency based approach, to calculate the "anomaly score" of a packet. The fewer times a given packet was seen, the higher was its anomaly score. In other words, the authors define an anomaly score as the degree of strangeness based on recent past activity. Once the anomaly score crossed a threshold, the packets were forwarded to a correlation engine that was designed to detect port scans. However, the one major drawback for SPADE is that it has a very high false alarm rate. This is due to the fact that SPADE classifies all unseen packets as attacks regardless of whether they are actually intrusions or not.

Anomalies resulting from intrusions may cause deviations on multiple measures in a collective manner rather than through separate manifestations on individual measures. To overcome the latter problem, Ye et al. [19] presented a technique that used the Hotellings T^2 test to analyze the audit trails of activities in an information system and detect host based intrusions. The assumption is that host based intrusions leave trails in the audit data. The advantage of using the Hotellings T^2 test is that it aids in the detection of both counter relationship anomalies as well as mean-shift anomalies. In another paper, Kruegel et al. [20] show that it is possible to find the description of a system that computes a payload byte distribution and combines this information with extracted packet header features. In this approach, the resultant ASCII characters are sorted by frequency and then aggregated into six groups. However, this approach

leads to a very coarse classification of the payload.
A problem that many network/system administrators face is the problem of defining, on a global scale, what network/system/user activity can be termed as "normal". Maxion et al. [21] characterized the normal behavior in a network by using different templates that were derived by taking the standard deviations of Ethernet load and packet count at various periods in time. An observation was declared anomalous if it exceeded the upper bound of a predefined threshold. However, Maxion et al. did not consider the non- stationary nature of network traffic which would have resulted in minor deviations in network traffic to go unnoticed.
More recently, analytical studies on anomaly detection systems were conducted. Lee and Xiang [22] used several information-theoretic measures, such as entropy and information gain, to evaluate the quality of anomaly detection methods, determine system parameters, and build models. These metrics help one to understand the fundamental properties of audit data.

2.5 Machine learning based anomaly detection

Machine learning can be defined as the ability of a program and/or a system to learn and improve their performance on a certain task or group of tasks over time. Machine learning aims to answer many of the same questions as statistics or data mining. However, unlike statistical approaches which tend to focus on understanding the process that generated the data, machine learning techniques focus on building a system that improves its performance based on previous results. In other words systems that are based on the machine learning paradigm have the ability to change their execution strategy on the basis of newly acquired information.

System call based sequence analysis
One of the widely used machine learning techniques for anomaly detection involves learning the behavior of a program and recognizing significant deviations from the normal. In a seminal paper, Forrest et al. [23] established an analogy between the human immune system and intrusion detection. They did this by proposing a methodology that involved analyzing a program's system call sequences to build a normal profile. In their paper, they analyzed several UNIX based programs like sendmail, lpr etc. and showed that correlations in fixed length sequences of system calls could be used to build a normal profile of a program. Therefore, programs that show sequences that deviated from the normal sequence profile could then be considered to be victims of an attack. The system they developed was only used off-line using previously collected data and used a quite simple table-lookup algorithm to learn the profiles of programs. Their work was extended by Hofmeyr et al. [24], where they collected a database of normal behavior for each program of interest. Once a stable database is constructed for a given program in a particular environment, the database was then used to monitor the program's behavior. The sequences of system calls formed the set of normal patterns for the database, and sequences not found in the database indicated anomalies.
Another machine learning technique that has been frequently used in the domain of machine learning is the sliding window method. The sliding window method, a sequential learning methodology, converts the sequential learning problem into the classical learning problem. It constructs a window classifier h_w that maps an input window of width w into an individual output value y. Specifically, let $d = (w - 1)/2$ be the "half-width" of the window. Then h_w predicts $y_{i,t}$ using the window $< x_{i,t-d}, x_{i,t-d+1}, ..., x_{i,t}, ...,$

$x_{i,t+d-1}$, $x_{i,t+d}$ >. The window classifier h_w is trained by converting each sequential training example (x_i , y_i) into windows and then applying a standard machine learning algorithm. A new sequence x is classified by converting it to windows, applying h_w to predict each y_t and then concatenating the y_t's to form the predicted sequence y. The obvious advantage of this sliding window method is that it permits any classical supervised learning algorithm to be applied. While, the sliding window method gives adequate performance in many applications; it does not take advantage of correlations between nearby y_t values. If there are correlations among the y_t values that are independent of the x_t values, then these are not captured. The sliding window method has been successfully used in a number of machine learning based anomaly detection techniques [25-27]. Warrender et al. [27] proposed a method that utilized sliding windows to create a database of normal sequences for testing against test instances. Eskin et al., in [26], improved the traditional sliding window method by proposing a modeling methodology that uses dynamic length of a sliding window dependent on the context of the system-call sequence.

However, system call based approaches for host based intrusion detection system suffer from two drawbacks. Firstly, the computational overhead that is involved in monitoring every system call is very high. This high overhead leads to a performance degradation of the monitored system. The second problem is that system calls themselves are irregular by nature. This irregularity leads to high false positive rate as it becomes difficult to differentiate between normal and anomalous system calls.

Bayesian networks

A Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest. When used in conjunction with statistical techniques, Bayesian networks have several advantages for data analysis [28]. Firstly, because Bayesian networks encode the interdependencies between variables, they can handle situations where data is missing. Secondly, Bayesian networks have the ability to represent causal relationships. Therefore, they can be used to predict the consequences of an action. Lastly, because Bayesian networks have both causal and probabilistic relationships, they can be used to model problems where there is a need to combine prior knowledge with data. Several researchers have adapted ideas from Bayesian statistics to create models for anomaly detection [29-31] . Valdes et al. [30] developed an anomaly detection system that employed naïve Bayesian networks to perform intrusion detection on traffic bursts. Their model, which is a part of EMERALD [32], has the capability to potentially detect distributed attacks in which each individual attack session is not suspicious enough to generate an alert. However, this scheme also has a few disadvantages. First, as pointed out in [29], the classification capability of naïve Bayesian networks is identical to a threshold based system that computes the sum of the outputs obtained from the child nodes. Secondly, because the child nodes do not interact between themselves and their output only influences the probability of the root node, incorporating additional information becomes difficult as the variables that contain the information cannot directly interact with the child nodes.

Another area, within the domain of anomaly detection, where Bayesian techniques have been frequently used is the classification and suppression of false alarms. Kruegel et al. [29] proposed a multi-sensor fusion approach where the outputs of different IDS sensors were aggregated to produce a single alarm. This approach is based on the assumption that any anomaly detection technique cannot classify a set of events as an intrusion with sufficient confidence. Although using Bayesian networks for intrusion detection or intruder behavior prediction can be effective in certain applications, their limitations should be considered in the actual implementation. Since the accuracy of this method is dependent on

certain assumptions that are typically based on the behavioral model of the target system, deviating from those assumptions will decrease its accuracy. Selecting an inaccurate model will lead to an inaccurate detection system. Therefore, selecting an accurate model is the first step towards solving the problem. Unfortunately selecting an accurate behavioral model is not an easy task as typical systems and/or networks are complex.

Principal components analysis

Typical datasets for intrusion detection are typically very large and multidimensional. With the growth of high speed networks and distributed network based data intensive applications storing, processing, transmitting, visualizing and understanding the data is becoming more complex and expensive. To tackle the problem of high dimensional datasets, researchers have developed a dimensionality reduction technique known as Principal Component Analysis (PCA) [33-35]. In mathematical terms, PCA is a technique where n correlated random variables are transformed into d ≤ n uncorrelated variables. The uncorrelated variables are linear combinations of the original variables and can be used to express the data in a reduced form. Typically, the first principal component of the transformation is the linear combination of the original variables with the largest variance. In other words, the first principal component is the projection on the direction in which the variance of the projection is maximized. The second principal component is the linear combination of the original variables with the second largest variance and orthogonal to the first principal component, and so on. In many data sets, the first several principal components contribute most of the variance in the original data set, so that the rest can be disregarded with minimal loss of the variance for dimension reduction of the dataset.

PCA has been widely used in the domain of image compression, pattern recognition and intrusion detection. Shyu et al. [36] proposed an anomaly detection scheme, where PCA was used as an outlier detection scheme and was applied to reduce the dimensionality of the audit data and arrive at a classifier that is a function of the principal components. They measured the Mahalanobis distance of each observation from the center of the data for anomaly detection. The Mahalanobis distance is computed based on the sum of squares of the standardized principal component scores. Shyu et al. evaluated their method over the KDD CUP99 data and have demonstrated that it exhibits better detection rate than other well known outlier based anomaly detection algorithms such as the Local Outlier Factor "LOF" approach, the Nearest Neighbor approach and the k^{th} Nearest Neighbor approach. Other notable techniques that employ the principal component analysis methodology include the work done by Wang et al. [35], Bouzida et al. [37] and Wang et al. [38].

Markov models

Markov chains, have also been employed extensively for anomaly detection. Ye et al. [39], present an anomaly detection technique that is based on Markov chains. In their paper, system call event sequences from the recent past were studied by opening an observation window of size N. The type of audit events, E_{t-N+1} ,..., E_t in the window at time t was examined and the sequence of states X_{t-N+1} ,..., X_t obtained. Subsequently, the probability that the sequence of states X_{t-N+1} ,..., X_t is normal was obtained. The larger the probability, the more likely the sequence of states results from normal activities. A sequence of states from attack activities is presumed to receive a low probability of support from the Markov chain model of the normal profile.

A hidden Markov model, another popular Markov technique, is a statistical model where the system

being modeled is assumed to be a Markov process with unknown parameters. The challenge is to determine the hidden parameters from the observable parameters. Unlike a regular Markov model, where the state transition probabilities are the only parameters and the state of the system is directly observable, in a hidden Markov model, the only visible elements are the variables of the system that are influenced by the state of the system, and the state of the system itself is hidden. A hidden Markov model's states represent some unobservable condition of the system being modeled. In each state, there is a certain probability of producing any of the observable system outputs and a separate probability indicating the likely next states. By having different output probability distributions in each of the states, and allowing the system to change states over time, the model is capable of representing non-stationary sequences.

To estimate the parameters of a hidden Markov model for modeling normal system behavior, sequences of normal events collected from normal system operation are used as training data. An expectation-maximization (EM) algorithm is used to estimate the parameters. Once a hidden Markov model has been trained, when confronted with test data, probability measures can be used as thresholds for anomaly detection. In order to use hidden Markov models for anomaly detection, three key problems need to be addressed. The first problem, also known as the evaluation problem, is to determine given a sequence of observations, what is the probability that the observed sequence was generated by the model. The second is the learning problem which involves building from the audit data a model, or a set of models, that correctly describes the observed behavior. Given a hidden Markov model and the associated observations, the third problem, also known as the decoding problem, involves determining the most likely set of hidden states that have led to those observations.

Warrender et al. [27] compare the performance of four methods viz., simple enumeration of observed sequences, comparison of relative frequencies of different sequences, a rule induction technique, and hidden Markov models at representing normal behavior accurately and recognizing intrusions in system call datasets. The authors show that while hidden Markov models outperform the other three methods, the higher performance comes at a greater computational cost. In the proposed model, the authors use an hidden Markov model with fully connected states, i.e., transitions were allowed from any state to any other state. Therefore, a process that issues S system calls will have S states. This implies that we will roughly have $2S^2$ values in the state transition matrix. In a computer system/network, a process typically issues a very large number of system calls. Modeling all of the processes in a computer system/network would therefore be computationally infeasible.

In another paper, Yeung et al. [40] describe the use of hidden Markov models for anomaly detection based on profiling system call sequences and shell command sequences. On training, their model computes the sample likelihood of an observed sequence using the forward or backward algorithm. A threshold on the probability, based on the minimum likelihood among all training sequences, was used to discriminate between normal and anomalous behavior. One major problem with this approach is that it lacks generalization and/or support for users who are not uniquely identified by the system under consideration.

Mahoney et al. in [41-43] presented several methods that address the problem of detecting anomalies in the usage of network protocols by inspecting packet headers. The common denominator of all of them is the systematic application of learning techniques to automatically obtain profiles of normal behavior for protocols at different layers. Mahoney et al. experimented with anomaly detection over the DARPA network data [44] by range matching network packet header fields. PHAD (Packet Header Anomaly

Detector) [41], LERAD (LEarning Rules for Anomaly Detection) [42] and ALAD (Application Layer Anomaly Detector) [43] use time-based models in which the probability of an event depends on the time since it last occurred. For each attribute, they collect a set of allowed values and flag novel values as anomalous. PHAD, ALAD, and LERAD differ in the attributes that they monitor. PHAD monitors 33 attributes from the Ethernet, IP and transport layer packet headers. ALAD models incoming server TCP requests: source and destination IP addresses and ports, opening and closing TCP flags, and the list of commands (the first word on each line) in the application payload. Depending on the attribute, it builds separate models for each target host, port number (service), or host/port combination. LERAD also models TCP connections. Even though, the data set is multivariate network traffic data containing fields extracted out of the packet headers, the authors break down the multivariate problem into a set of univariate problems and sum the weighted results from range matching along each dimension. While the advantage of this approach is that it makes the technique more computationally efficient and effective at detecting network intrusions, breaking multivariate data into univariate data has significant drawbacks especially at detecting attacks. For example, in a typical SYN flood attack an indicator of the attack, is having more SYN requests than usual, but observing a lower than normal ACK rate. Because higher SYN rate or lower ACK rate alone can both happen in normal usage (when the network is busy or idle), it is the combination of higher SYN rate and lower ACK rate that signals the attack.

The one major drawback of many of the machine learning techniques, like the system call based sequence analysis approach and the hidden Markov model approach mentioned above, is that they are resource expensive. For example, an anomaly detection technique that is based on the Markov chain model is computationally expensive because it uses parametric estimation techniques based on the Bayes' algorithm for learning the normal profile of the host/network under consideration. If we consider the large amount of audit data and the relatively high frequency of events that occur in computers and networks of today, such a technique for anomaly detection is not scalable for real time operation.

2.6 Data mining based anomaly detection

To eliminate the manual and ad-hoc elements from the process of building an intrusion detection system, researchers are increasingly looking at using data mining techniques for anomaly detection [45-47]. Grossman [48] defines data mining as being "concerned with uncovering patterns, associations, changes, anomalies, and statistically significant structures and events in data". Simply put data mining is the ability to take data as input, and pull from it patterns or deviations which may not be seen easily to the naked eye. Another term sometimes used is knowledge discovery. Data mining can help improve the process of intrusion detection by adding a level of focus to anomaly detection. By identifying bounds for valid network activity, data mining will aid an analyst in his/her ability to distinguish attack activity from common everyday traffic on the network.

Classification-based intrusion detection

An intrusion detection system that classifies audit data as normal or anomalous based on a set of rules, patterns or other affiliated techniques can be broadly defined as a classification-based intrusion detection system. The classification process typically involves the following steps:

1. Identify class attributes and classes from training data;
2. Identify attributes for classification;
3. Learn a model using the training data;
4. Use the learned model to classify the unknown data samples.

A variety of classification techniques have been proposed in the literature. These include inductive rule generation techniques, fuzzy logic, genetic algorithms and neural networks-based techniques.

Inductive rule generation algorithms typically involve the application of a set of association rules and frequent episode patterns to classify the audit data. In this context, if a rule states that "if event X occurs, then event Y is likely to occur", then events X and Y can be described as sets of (variable, value)-pairs where the aim is to find the sets X and Y such that X "implies" Y. In the domain of classification, we fix Y and attempt to find sets of X which are good predictors for the right classification. While supervised classification typically only derives rules relating to a single attribute, general rule induction techniques, which are typically unsupervised in nature, derive rules relating to any or all the attributes. The advantage of using rules is that they tend to be simple and intuitive, unstructured and less rigid. As the drawbacks they are difficult to maintain, and in some cases, are inadequate to represent many types of information. A number of inductive rule generation algorithms have been proposed in literature. Some of them first construct a decision tree and then extract a set of classification rules from the decision tree. Other algorithms (for eg. RIPPER [25], C4.5 [49]) directly induce rules from the data by employing a divide- and-conquer approach. A post learning stage involving either discarding (C4.5 [49]) or pruning (RIPPER [25]) some of the learnt rules is carried out to increase the classifier accuracy. RIPPER has been successfully used in a number of data mining based anomaly detection algorithms to classify incoming audit data and detect intrusions. One of the primary advantages of using RIPPER is that the generated rules are easy to use and verify. Lee et al. [45, 46, 50] used RIPPER to characterize sequences occurring in normal data by a smaller set of rules that capture the common elements in those sequences. During monitoring, sequences violating those rules are treated as anomalies.

Fuzzy logic techniques have been in use in the area of computer and network security since the late 1990's [51]. Fuzzy logic has been used for intrusion detection for two primary reasons [52]. Firstly, several quantitative parameters that are used in the context of intrusion detection, e.g., CPU usage time, connection interval etc., can potentially be viewed as fuzzy variables. Secondly, as stated by Bridges et al. [52], the concept of security itself is fuzzy. In other words, the concept of fuzziness helps to smooth out the abrupt separation of normal behavior from abnormal behavior. That is, a given data point falling outside/inside a defined "normal interval", will be considered anomalous/normal to the same degree regardless of its distance from/within the interval. Dickerson et al., [53] developed the Fuzzy Intrusion Recognition Engine (FIRE) using fuzzy sets and fuzzy rules. FIRE uses simple data mining techniques to process the network input data and generate fuzzy sets for every observed feature. The fuzzy sets are then used to define fuzzy rules to detect individual attacks. FIRE does not establish any sort of model representing the current state of the system, but instead relies on attack specific rules for detection.

Instead, FIRE creates and applies fuzzy logic rules to the audit data to classify it as normal or anomalous. Dickerson et al. found that the approach is particularly effective against port scans and probes. The primary disadvantage to this approach is the labor intensive rule generation process.

Genetic algorithms, a search technique used to find approximate solutions to optimization and search

problems, have also been extensively employed in the domain of intrusion detection to differentiate normal network traffic from anomalous connections. The major advantage of genetic algorithms is their flexibility and robustness as a global search method. In addition, a genetic algorithm search converges to a solution from multiple directions and is based on probabilistic rules instead of deterministic ones. In the domain of network intrusion detection, genetic algorithms have been used in a number of ways. Some approaches [54, 55] have used genetic algorithms directly to derive classification rules, while others [52, 56] use genetic algorithms to select appropriate features or determine optimal parameters of related functions, while different data mining techniques are then used to acquire the rules. The earliest attempt to apply genetic algorithms to the problem of intrusion detection was done by Crosbie et al. [57] in 1995, when they applied multiple agent technology to detect network based anomalies. While the advantage of the approach was that it used numerous agents to monitor a variety of network based parameters, lack of intra-agent communication and a lengthy training process were some issues that were not addressed.

Neural network based intrusion detection systems have traditionally been host based systems that focus on detecting deviations in program behavior as a sign of an anomaly. In the neural network approach to intrusion detection, the neural network learns to predict the behavior of the various users and daemons in the system. The main advantage of neural networks is their tolerance to imprecise data and uncertain information and their ability to infer solutions from data without having prior knowledge of the regularities in the data. This in combination with their ability to generalize from learned data has shown made them an appropriate approach to intrusion detection. However, the neural network based solutions have several drawbacks. First they may fail to find a satisfactory solution either because of lack of sufficient data or because there is no learnable function. Secondly, neural networks can be slow and expensive to train. The lack of speed is partly because of the need to collect and analyze the training data and partly because the neural network has to manipulate the weights of the individual neurons to arrive at the correct solution. There are a few different groups advocating various approaches to using neural networks for intrusion detection. Ghosh et al. [58-60] used the feed-forward back propagation and the Elman recurrent network [61] for classifying system-call sequences. Their experimental results with the 1998 and 1999 DARPA intrusion detection evaluation dataset verified that the application of Elman networks in the domain of program-based intrusion detection provided superior results as compared to using the standard multilayer perceptron based neural network. However, training the Elman network was expensive and the number of neural networks required was large. In another paper, Ramadas et al. [62] present the Anomalous Network-Traffic Detection with Self Organizing Maps (ANDSOM). ANDSOM is the anomaly detection module for the network based intrusion detection system, called INBOUNDS, being developed at Ohio University. The ANDSOM module creates a two dimensional Self Organizing Map or SOM for each network service that is being monitored. In the paper, the authors test the proposed methodology using the DNS and HTTP services. Neurons are trained with normal network traffic during the training phase to capture characteristic patterns. When real time data is fed to the trained neurons, then an anomaly is detected if the distance of the incoming traffic is more than a preset threshold.

Anomaly detection schemes also involve other data mining techniques such as support vector machines (SVM) and other types of neural network models [63, 64]. Because data mining techniques are data driven and do not depend on previously observed patterns of network/system activity, some of these techniques have been very successful at detecting new kinds of attacks. However, these techniques often

have a very high false positive rate. For example, as pointed out in [65], the approach adopted by Sung et al. [66] that use a SVM technique to realize an intrusion detection system for class-specific detection is flawed because they totally ignore the relationships and dependencies between the features.

Clustering and outlier detection

Clustering is a technique for finding patterns in unlabeled data with many dimensions (the number of dimensions is equivalent to the number of attributes). Clustering has attracted interest from researchers in the context of intrusion detection [67-69] . The main advantage that clustering provides is the ability to learn from and detect intrusions in the audit data, while not requiring the system administrator to provide explicit descriptions of various attack classes/types. As a result, the amount of training data that needs to be provided to the anomaly detection system is also reduced. Clustering and outlier detection are closely related. From the viewpoint of a clustering algorithm, outliers are objects not located in the clusters of a data set, and in the context of anomaly detection, they may represent intrusions/attacks. The statistics community has studied the concept of outliers quite extensively [70]. In these studies, data points are modeled using a stochastic distribution and points are determined to be outliers based on their relationship with this model. However, with increasing dimensionality, it becomes increasingly difficult to accurately estimate the multidimensional distributions of the data points [71]. Recent outlier detection algorithms [68, 72, 73] are based on the full dimensional distances between the points as well as the densities of local neighborhoods.

There exist at least two approaches to clustering based anomaly detection. In the first approach, the anomaly detection model is trained using unlabeled data that consists of both normal as well as attack traffic. In the second approach, the model is trained using only normal data and a profile of normal activity is created. The idea behind the first approach is that anomalous or attack data forms a small percentage of the total data. If this assumption holds, anomalies and attacks can be detected based on cluster sizes-large clusters correspond to normal data, and the rest of the data points, which are outliers, correspond to attacks.

The distances between points play an important role in clustering. The most popular distance metric is the Euclidean distance. Since each feature contributes equally to the calculation of the Euclidean distance, this distance is undesirable in many applications. This is especially true when features have very different variability or different features are measured on different scales. The effect of the features that have large scales of measurement or high variability would dominate others that have smaller scales or less variability. A distance-based approach which incorporates this measure of variability is the Mahalanobis distance [74, 75] based outlier detection scheme. In this scheme, the threshold is computed according to the most distant points from the mean of the “normal” data, and it is set to be a user defined percentage, say m% , of the total number of points. In this scheme, all data points in the audit data that have distances to the mean (calculated during the training phase) greater than the threshold are detected as outliers. The Mahalanobis distance utilizes group means and variances for each variable, and the correlations and covariances between measures.

The notion of distance-based outliers was recently introduced in the study of databases [68]. According to this notion, a point, P, in a multidimensional data set is an outlier if there are less than p points from the data in the -neighborhood of P, where p is a user-specified constant. In [68], Ramaswamy et al. described an approach that is based on computing the Euclidean distance of the kth nearest neighbor from a point O. In other words, the k-nearest neighbor algorithm classifies points by assigning them to

the class that appears most frequently amongst the k nearest neighbors. Therefore, for a given point O, d k (O) denotes the Euclidean distance from the point O to its kth nearest neighbor and can be considered as the "degree of outlierness" of O. If one is interested in the top n outliers, this approach defines an outlier as follows: Given values for k and n , a point O is an outlier, if the distance to its k th nearest neighbor is smaller than the corresponding value for no more than (n − 1) other points. In other words, the top n outliers with the maximum D_k (O) values are considered as outliers. While the advantage of the k-nearest neighbors approach is that it is robust to noisy data, the approach suffers from the drawback that it is very difficult to choose an optimal value for k in practice. Several papers [76, 77] have proposed using the k- nearest neighbor outlier detection algorithms for the purpose of anomaly detection.

The Minnesota Intrusion Detection System (MINDS) [78] is another network-based anomaly detection approach that utilizes data mining techniques. The MINDS anomaly detection module assigns a degree of outlierness to each data point, which is called the local outlier factor (LOF) [72]. The LOF takes into consideration the density of the neighborhood around the observation point to determine its outlierness. In this scheme, outliers are objects that tend to have high LOF values. The advantage of the LOF algorithm is its ability to detect all forms of outliers, including those that cannot be detected by the distance-based algorithms.

Association rule discovery

Association rules [79, 80] are one of many data mining techniques that describe events that tend to occur together. The concept of association rules can be understood as follows: Given a database D of transactions where each transaction T ∈D denotes a set of items in the database, an association rule is an implication of the form X => Y , where X ⊂ D , Y ⊂ D and X ∩ Y = ∅ . The rule X => Y holds in the transaction set D with confidence c if c% of transactions in X also contain Y . Two important concepts when dealing with association rules are rule confidence and rule support. The probability of rule confidence is defined as the conditional probability P (Y ⊆ T | X ⊆ T) The rule X => Y has support s in the transaction database D if s% of transactions in D contain X ∪ Y . Association rules have been successfully used to mine audit data to find normal patterns for anomaly detection [46, 50, 81]. They are particularly important in the domain of anomaly detection because association rules can be used to construct a summary of anomalous connections detected by the intrusion detection system. There is evidence that suggests program executions and user activities exhibit frequent correlations among system features. These consistent behaviors can be captured in association rules.

Lee et al. [46, 50] proposed an association rule-based data mining approach for anomaly detection where raw data was converted into ASCII network packet information, which in turn was converted into connection-level information. These connection level records contained connection features like service, duration etc. Association rules were then applied to this data to create models to detect intrusions. In another paper, Barbará et al. describe ADAM (Audit Data Analysis and Mining), a real-time anomaly detection system that uses a module to classify the suspicious events into false alarms or real attacks. ADAM was one out of seven systems tested in the 1999 DARPA evaluation [82]. It uses data mining to build a customizable profile of rules of normal behavior and then classifies attacks (by name) or declares false alarms. To discover attacks in TCPdump audit trail, ADAM uses a combination of association rules, mining and classification. During the training phase, ADAM builds a database of "normal" frequent itemsets using attack free data. Then it runs a sliding window online algorithm that

finds frequent item sets in the last D connections and compares them with those stored in the normal item set repository. With the remaining item sets that have deemed suspicious, ADAM uses a classifier which has previously been trained to classify the suspicious connections as a known attack, unknown attack, or a false alarm. Association rules are used to gather necessary knowledge about the nature of the audit data. If the item set's support surpasses a threshold, then that item set is reported as suspicious. The system annotates suspicious item sets with a vector of parameters. Since the system knows where the attacks are in the training set, the corresponding suspicious item set along with their feature vectors are used to train a classifier. The trained classifier will be able to, given a suspicious item set and a vector of features, classify the item set as a known attack (and label it with the name of attack), an unknown attack, or a false alarm.

2.7 Hybrid systems

It has been suggested in the literature [14, 15, 32, 83, 84] that the monitoring capability of current intrusion detection systems can be improved by taking a hybrid approach that consists of both anomaly as well as signature detection strategies. In such a hybrid system, the anomaly detection technique aids in the detection of new or unknown attacks while the signature detection technique detects known attacks. The signature detection technique will also be able to detect attacks launched by a patient attacker who attempts to change the behavior patterns with the objective of retraining the anomaly detection module so that it will accept attack behavior as normal. Tombini et al. [83] used an approach wherein the anomaly detection technique is used to produce a list of suspicious items. The classifier module which uses a signature detection technique then classified the suspicious items into false alarms, attacks, and unknown attacks. This approach works on the premise that the anomaly detection component would have a high detection rate, since missed intrusions cannot be detected by the follow-up signature detection component. In addition, it also assumed that the signature detection component will be able to identify false alarms. While the hybrid system can still miss certain types of attacks, its reduced false alarm rate increases the likelihood of examining most of the alerts.

EMERALD [32] was developed in the late 1990's at SRI. It is an extension of the seminal work done in [8, 13-15]. EMERALD is a hierarchical intrusion detection system that monitors systems at a variety of levels viz. individual host machines, domains and enterprises to form an analysis hierarchy. EMERALD uses a subscription-based communication scheme both within and between monitors. However, inter monitor subscription methodology is hierarchical and therefore limits the access to events and/or results from the layer immediately below. The system has a built-in feedback system that enables the higher layers to request more information about particular anomalies from the lower layers. To achieve a high rate of detection, the architects of EMERALD employed an ensemble of techniques like statistical analysis engines and expert systems. The single most defining feature of EMERALD is its ability to analyze system-wide, domain-wide and enterprise-wide attacks like Internet worms, DDoS attacks etc. at the top level.

In another paper [84], Zhang et al. [85] employed the random forests algorithm in the signature detection module to detect known intrusions. Thereafter, the outlier detection provided by the random forests algorithm is utilized to detect unknown intrusions. Approaches that use signature detection and anomaly detection in parallel have also been proposed. In such systems, two sets of reports of possible intrusive activity are produced and a correlation component analyzes both sets to detect intrusions. An

example of such a system is NIDES [15, 16].
Lee et al. [45, 86] extended the work done by them in [50] and proposed a hybrid detection scheme that utilized the Common Intrusion Detection Framework (CIDF) to automatically get audit data, build models, and distribute signatures for novel attacks to ensure that the time required to detect them is reduced. The advantage of using CIDF was that it enabled different intrusion detection and response components to interoperate and share the information and resources in a distributed environment. Although it is true that combining multiple intrusion detection technologies into a single system can theoretically produce a much stronger intrusion detection system, the resulting hybrid systems are not always better. Different intrusion detection technologies examine system and/or network traffic and look for intrusive activity in different ways. Therefore, the major challenge to building an operational hybrid intrusion detection system is getting these different technologies to interoperate effectively and efficiently.

3. An Anomaly Detector based on PCA

3.1 The algorithm

Purpose of this software is to analyze a network traffic trace (in our specific case this will be Voice over IP traffic collected by a telephonic operator), collected over a certain amount of time (measured in time-bins of desired duration) and detect in which of these time-bins the behavior of a certain user seems to be anomalous, trying then to identify which is the user responsible for the anomaly detected in that time-bin.
To obtain this results the algorithm uses several stages of computation, that we are going to resume in the following chapters.

3.2 1st phase: Traffic Aggregation

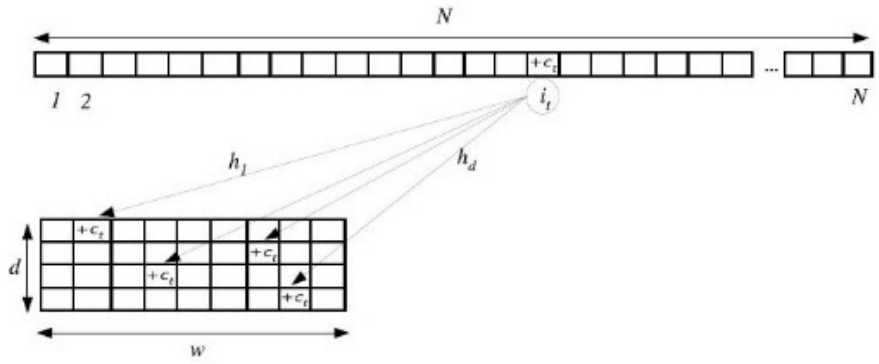
The data used as an input is a list of the traffic (in our specific case the number of calls attempted) originated from VoIP users in our specific network, identified by their URIs, in the selected time bin (we used time-bins of 1 hour). The amount of users in a net can be very high (in the trace we used the number of users was in the order of millions), so, in order to obtain a more efficient computation, the traffic needs to be aggregated; this is performed by means of sketches functions.
Sketches are a family of data structures that use the same underlying hashing scheme for summarizing data. They differ in how they update hash buckets and use hashed data to derive estimates. Among the different sketches, the one with the best time and space bounds is the so called count-min sketch [87], which is basically the one used in this work.
In more detail, the sketch data structure is a two-dimensional d×w array T[l][j], where each row l (l = 1, . . . , d) is associated to a given hash function h_l . In our work, we have chosen to use functions belonging to the 4-universal hash family[1] [88]. These functions give an output in the interval (1, . . . , w) and these outputs are associated to the columns of the array. As an example, the element T[l][j] is associated to the output value j of the hash function l.
The input data is viewed as a stream that arrives sequentially, item by item. Each item consists of an hash key, $i_t \in (1, \ldots, N)$, and a weight, c_t . When new data arrive, the sketch is updated as follows:

$$T[l][h_l(i_t)] \leftarrow T[l][h_l(i_t)] + c_t$$

The update procedure is realized for all the different hash functions as shown in Figure 1.

1 A class of hash functions H : (1, . . . , N) → (1, . . . , w) is a k-universal hash if for any distinct x_0 , . . ., $x_{k-1} \in (1, \ldots, N)$ and any possibly v_0 , . . ., $v_{k-1} \in (1, \ldots, w)$: $\Pr_{h \in H} = \{h(x_i) = v_i; \forall i \in (1, \ldots, k)\} = \frac{1}{w^k}$

Figure 1



To be noted that, given the use of hash functions, it is possible to have some collisions in the sketch table. In this work we have taken advantage of this fact, indeed having collisions allows us to randomly aggregate the traffic flows, namely all the IP flow (URIs) that collide in the same bucket will be considered as an aggregate. To be noted that in this way each IP flow (URI) will be part of d (in the software used this is equal to 8) distinct random aggregates, each of which will be analyzed to check if it presents any anomaly. This means that, in practice, any flow will be checked more than once, thus, it will be easier to detect an anomalous flow. Indeed, an anomalous flow could be masked in a given traffic aggregate, while being detectable in another one.
To resume, we can say that with this process we have a family of hash-functions (in the software used this is composed by 8 of them), each of which divide the traffic in a certain number of aggregates (in our case 64), because of the collisions that happen in the sketch table. We obtain a randomized aggregation of the traffic, different for each hash-function used: each IP flow (URI for VoIP traffic) will be part of several (8 in the specific case) distinct random aggregates, each of which will be analyzed to check if it presents any anomaly.
The fact of using different functions for aggregating the traffic will be the key feature in the mechanism that lead to the identification of the single IP flows (or URI) responsible for the anomalies detected, as we will explain later.
The output of this block is given by 4N distinct files, each file corresponding to a specific time-bin and a specific aggregate.

3.3 2nd phase: Time series construction

The assumption on which this anomaly detection technique is based is that an anomaly will induce changes in the normal distribution of some features in the network traffic; in the original software the creators took into consideration the number of bytes sent by each IP address, while in our case we chose to consider the number of attempted calls per user. Based on this observation, we have examined the distributions of these particular traffic features as a means to detect and to classify network anomalies.
The feature distribution has been estimated with the empirical histogram. Thus, in each time-bin for each aggregate we have evaluated the histogram as follows: $X^t = \{n_i^t, i = 1, \ldots, N\}$

where n^t_i is the number of bytes transmitted by the *i-th* IP address (calls attempted by the *i-th* URI) in the time-bin *t*.
Unfortunately, the histogram is a high dimensional object, quite difficult to handle in low time and computational resources (we should memorize a histogram, with dimension proportional to the number of users in the data trace, for each time-bin, traffic aggregate and type of aggregation). For this reason the algorithm tries to concentrate all the information taken by the histogram in a single value, able to hold the most useful information, that, in our case, is the trend of the distribution: this purpose is obtained computing the entropy of the distribution. The entropy provides a computational efficient metric for estimating the degree of dispersion or concentration of a distribution, capturing in a single value the distributional changes in traffic features, and observing the time series of entropy exposes unusual traffic behavior.
Given the empirical histogram, X^t, we can evaluate the entropy value as follows:

$$H^t = -\sum_{i=1}^{N} \frac{n_i^t}{S} \log_2 \frac{n_i^t}{S} \quad \text{with} \quad S = \sum_{i=1}^{N} n_i^t$$

This module will output a matrix for each type of aggregation in which, for all the aggregates, the values of the entropy evaluated in each time-bin are reported.
This matrix have the following structure:

$$Y = \begin{pmatrix} y_{11} & \cdots & y_{1N} \\ \vdots & & \vdots \\ y_{T1} & \cdots & y_{TN} \end{pmatrix}$$

where N is the number of aggregates and T the number of time-bins.

3.4 3rd phase: Principal Components computation

The Principal Components Analysis (PCA) is a linear transformation that maps a coordinate space onto a new coordinate system whose axes, called Principal Components (PCs), have the property to point in the directions of maximum variance of data.
In more detail, the first PC captures the greatest degree of data variance in a single direction, the second one captures the greatest degree of variance of data in the remaining orthogonal directions, and so on.
Thus, the PCs are ordered by the amount of data variance they capture. Typically, the first PCs contribute most of the variance in the original data set, so that we can describe them with only these PCs, neglecting the others, with minimal loss of variance.
In mathematical terms, to calculate the PCs is equivalent to compute the eigenvectors. Thus, given the matrix of data $B = \{B_{i,j}\}$, with $1 < i < m$ and $1 < j < t$ (a dataset of m simples captured in t time-bins), each PC, v_i , is the $i-th$ eigenvector computed from the spectral decomposition of B^TB, that is:

$$B^T B v_i = \lambda_i v_i \qquad i = 1, \ldots, m$$

where λ_i is the eigenvalue corresponding to the eigenvector v_i .
In practice , the first PC, v_1 , is computed as follows:

$$v_1 = \underset{\|v\|=1}{\operatorname{argmax}} \|Bv\|$$

Proceeding recursively, once the first k-1 PCs have been determined, the k-th PC can be evaluate as follows:

$$v_k = \underset{\|v\|=1}{\operatorname{argmax}} \left\| (B - \sum_{i=1}^{k-1} B v_i v_i^T) v \right\|$$

It corresponds to the direction of maximum variance in the residual data (i.e., the difference between the original data and the data mapped onto the first k-1 PCs).
The algorithm computes the Principal Components (using the previous equations) on the time-series of entropy values obtained in the previous step. As we said, there is a set of PCs (called *dominant* PCs) that contributes most of the variance in the original data set. The idea is to select the dominant PCs and describe the normal behavior only using these ones. It is worth highlighting that the number of dominant PCs is a very important parameter, and needs to be properly tuned when using PCA as a traffic anomaly detector, as it has a heavy impact on the software's performances in detecting anomalies.
As a result, we separate the PCs into two sets, dominant and negligible PCs, that will be then used to distinguish between normal and anomalous variations in traffic.
We will call P the set of dominant PCs (r is the number of dominant PCs selected): $P = (v_1, \dots, v_r)$.
These first 3 steps are the most time-consuming part of the algorithm, but fortunately they only need to be computed one time on a single trace, making easier to perform several analysis on a same trace (for example to estimate the best parameters to use), because the next steps (the detection and identification phases) are much faster.

3.5 4th phase: Detection phase

This step is performed by separating the high-dimensional space of traffic measurements in two subspace: the *normal* subspace ($\hat{S}$), spanned by the dominant PCs (which number is an important execution parameter), which capture the normal variations in the traffic, and the *anomalous* subspace (S), spanned by the other PCs, which capture the anomalous variations.
The normal and anomalous components of data are obtained by projecting the aggregate traffic, in a single time-bin, onto these two subspaces.
Thus, the original data, in the time-bin *t*, Y_t is decomposed into two parts as follows:

$$Y_t = \hat{Y}_t + \tilde{Y}_t$$

where $\hat{Y}t$ and $\tilde{Y}t$ are the projection onto $\hat{S}$ and S, respectively , and can be evaluate as follows:

$$\hat{Y}_t = PP^T Y_t \qquad\qquad \tilde{Y}_t = (1 - PP^T) Y_t$$

The presence of anomalous traffic in the network, in the time-bin taken in account, will provoke a large change in the anomalous components ($\tilde{Y}_t$). Thus, an efficient method to detect traffic anomalies is to compare the squared prediction error of t $\tilde{Y}_t$ ($||\tilde{Y}_t||^2$) with a threshold (ξ), givens as input to the algorithm (we'll see that the value of the threshold is a very important parameter); if it exceeds the threshold the traffic is considered anomalous, and that particular time-bin (t) is marked as anomalous.
In the first phase we used *d* different hash functions to aggregate the traffic (8 in our specific case); so,

the previously described analysis returns *d* different responses.
Thus, a voting analysis is performed. We evaluate the number of produced alarms and we decide if the time-bin is anomalous or not according to the following rule:

- Anomalous if number of alarms > d/2 – 1
- Normal otherwise

3.6 5th phase: Identification phase

After detecting the anomalous time-bins the system tries to perform an identification phase, to recognize the flows responsible of the anomalies. Up to the previous step we were only able to determinate in which time-bins an anomaly was present, but not the specific network event that caused that anomaly.
In fact it is worth noticing that an anomalous time-bin may contain multiple anomalous events, and that a single anomalous event can span over multiple time-bins. The algorithm is only able to identify a single responsible for an anomaly detected in a single time-bin.
In more detail, to identify the nature of an anomaly, at first, we search the specific traffic aggregation in which the anomaly has occurred. Since we consider a time-bin anomalous if the squared prediction error of the *anomalous* traffic in that time-bin exceeds the threshold, the identification method consists of searching the particular aggregate that if removed from the aforementioned statistics, would bring it under threshold.
This way we identify a set of candidate IP flows (VoIP users, in our case) responsible for the detected anomaly. Since we used *d* different hashing functions for aggregating the traffic, we have *d* different sets of candidate responsible flows for the anomaly detected. Given that, the candidate responsible flow can be found by simply intersecting all of these aggregates.
In this step we can define another important tunable parameter for the software: the number of hashing functions that we want to signal an aggregate in which the candidate anomalous user is present, to consider the identification phase successful. The simple intersection of 8 aggregates, for example, can lead to the identification of an user that is only part of 3 or 4 aggregates (only with 3 or 4 hashing functions we detect a candidate aggregate in which that user is present). Keeping the number of required functions low could lead to more identifications, but also to more false-alarms (users with a normal behavior identified as anomalous), while using a high number of required hashing functions could lead to a low number of users identified (situations in which the software detect a high number of time-bins as anomalous, but only for a few of them a responsible user is identified).
It's important to note that a mechanism such this can only identify a single responsible user for each anomaly, so we can't detect more than one anomalous user per time-bin (if two different users have an anomalous behavior in the same time-bin we can only identify one of them).

3.7 Software's specifications

The software uses as an input text files in CSV format: in our case this file is a list of the number of attempted calls, for each VoIP user (identified by his URI), in a time-bin of one hour.
This is an extract from an example input file (the format is: Hour,User,Calls) :

1,URIuser1,1
2,URIuser2,2
3,URIuser3,3
3,URIuser2,1

The software execution is made of two steps:

1. The software computes the first three phases of the algorithm, creating the matrices needed for the following step
2. The software uses the matrices previously created to perform the final two phases (detection and identification)

The first step is only required the first time we are using a certain trace, and only needs, as input, the file formatted as in the previous example.
The second step (much faster than the previous one) requires two other parameters to perform the final part of the algorithm:

- The number of Principal Components that we want to be included in the normal subspace (an integer)
- The value of the threshold that we want to use in the detection phase (a float)

We can choose an exact number of PCs and an exact threshold, or a range of PCs and a range of thresholds (scanned with a chosen step) for which we want to perform the analysis.
As we said before another tunable parameter is the number of hashing functions required to consider the identification method successful, but it must be chosen by modifying the code; it's not directly selectable as an input to the software.
Running the software (first the first step, then the second step as many times as needed) produces a series of text output files, one for each number of PCs and value of the threshold chosen for the analysis. These files contain a list of the time-bins considered anomalous by the fourth phase of the algorithm, each of which is followed by the candidate responsible user (identified by his URI) for the

anomaly in that time-bin (only if the identification phase is successful for that particular time-bin) with the number of hashing functions that signal, as a candidate responsible, an aggregate which that specific user is part of.
This is an extract from an example output file:

Anomaly detected with sketches in bin 10
Anomaly detected with sketches in bin 11
Anomaly detected with sketches in bin 14
Anomaly detected with sketches in bin 19
...
timebin: 10
timebin: 11
timebin: 14
num hash:8, responsible:URIuser1
timebin: 19
num hash:7, responsible:URIuser3
...

As we can see from the example not every detected time-bin is associated with a candidate responsible user.
In the Anomaly Detection process on VoIP traffic we are not interested just in finding anomalous time-bins (that in our case correspond to one hour of calls): the nature of the traffic is such that in almost every our of the day we can consider an anomaly to be present, so in the following we'll only consider as successfully detected the anomalies for which the identification method lead to a candidate responsible user (in the previous example just time-bins 14 and 19).

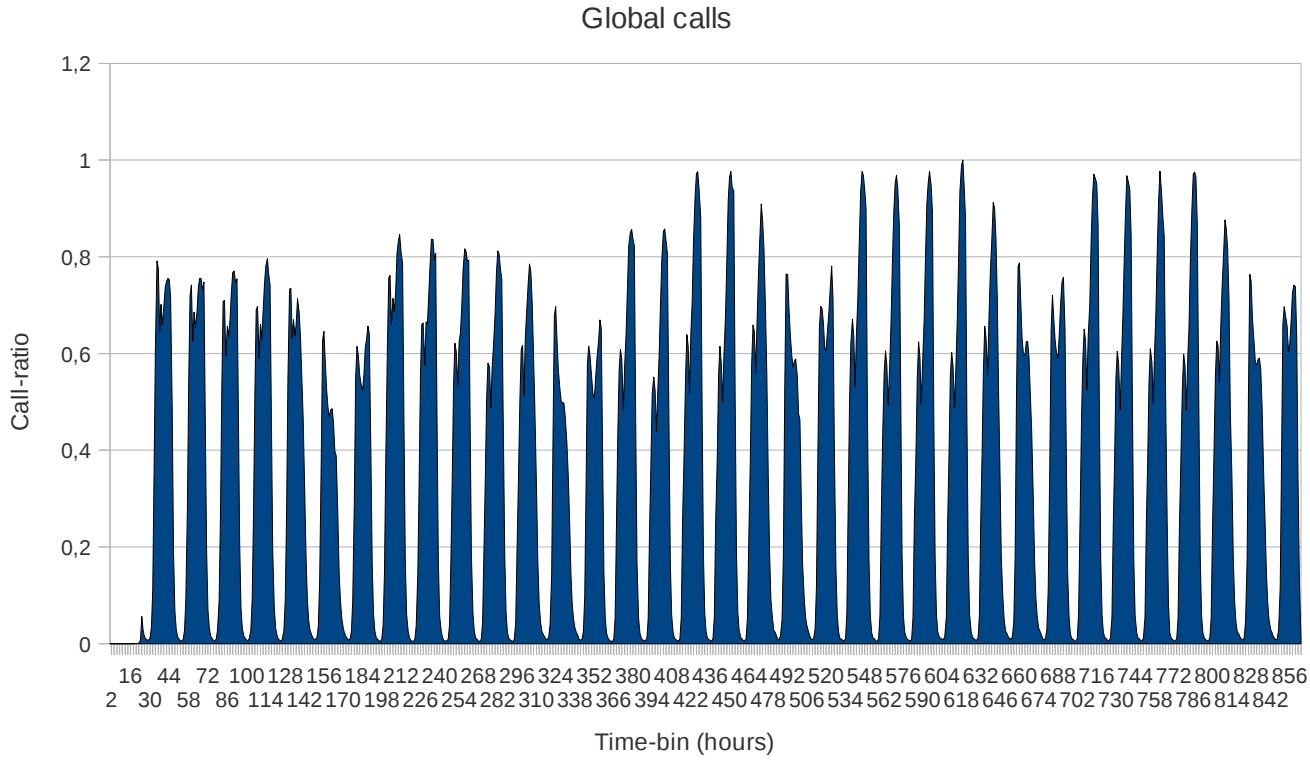
4. Single round analysis

4.1 Trace used

The first type of analysis we decided to perform is a standard analysis, in which we launched the software to detect anomalies upon a trace that records the number of calls per user per hour (we used time-bins of 1 hour) among a period of more than 800 hours, equal to a little more than one month of voice traffic.
The observation of the plot of total calls (obtained summing the number of calls for all the users, time-bin per time-bin) per hour in the trace, normalized by its maximum value (Figure 2) leads us to some considerations:

1. The first 19 hours of the trace are useless in the analysis, because there are no calls recorded. We'll see that the software automatically discards these time-bins, adapting the results to a different time reference, in which time-bin zero corresponds to the 19th hour in the original trace (the first with some calls recorded)
2. Observing the trend of global calls per hour we can recognize the typical behavior of voice traffic, with periodicity among the day (peaks in the working hours and a small amount of attempted calls during the night hours) and among the week (peaks of calls on Saturdays and Sundays are smaller than their corresponding ones in working days). We expect the software to adapt to this type of behavior and signal as anomalous users the ones who don't follow this trends.

Figure 2



To better understand which is the typical behavior of a single user we can focus on two graphs: the one that plots the average number of calls per hour among all users (the trend is the same as in figure 1 but the number of calls is scaled by the total number of users in the trace), in figure 3, and the one that plots the average number of calls per hour among only the active users (in each time-bin we consider only the users that attempt at least one call and we mediate among them to calculate the average number of calls in that hour), in figure 4. From these we can observe that the average number of calls attempted by a single user in one hour is quite low: even considering only the active users the average in a single hour never goes over three calls, while the global average (among the whole trace) is less than two calls (1.57 calls). For this reason even users that follow the normal behavior but with a quantity of calls per hour (in time-bins corresponding to peaks in the trace) more than ten times the average (over 20-30 calls in our case) can be considered as anomalous.

Figure 3

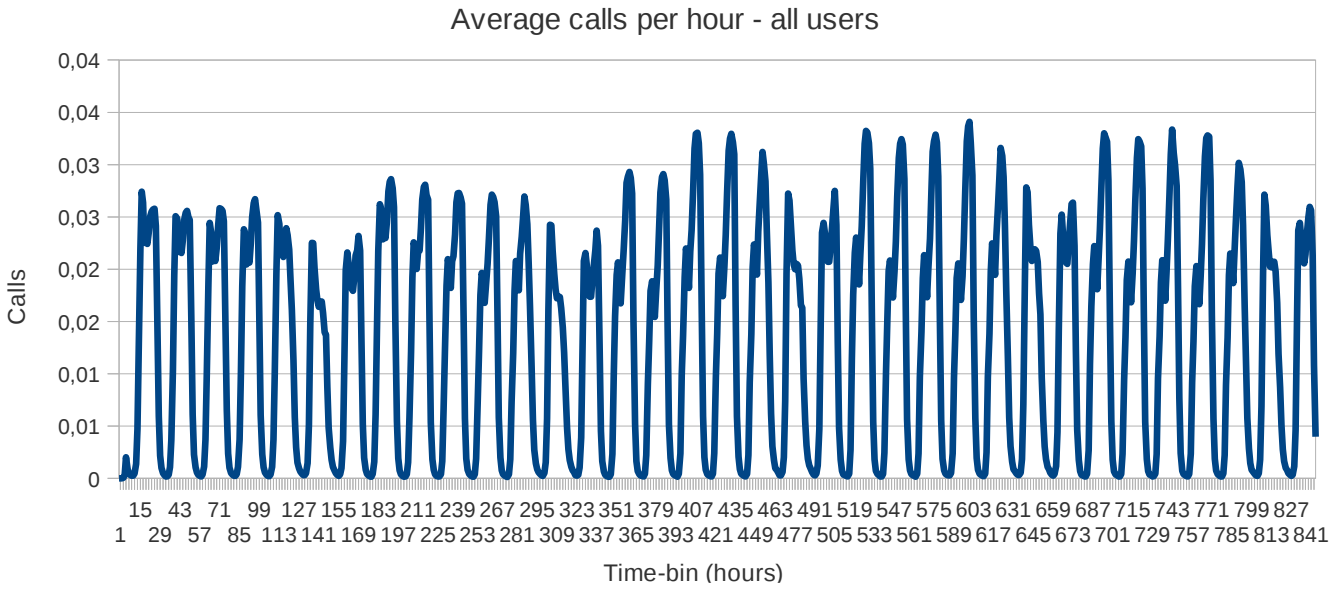
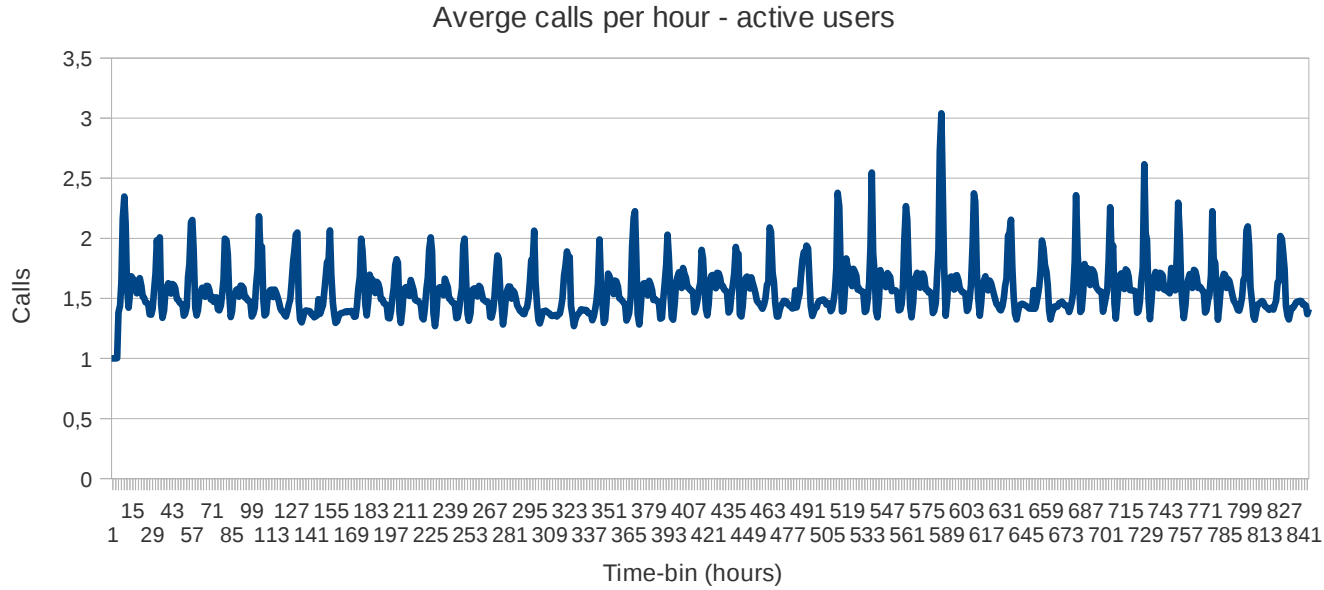


Figure 4



4.2 Characteristics of the analysis

Using only the trace, in CDR text file format (as described previously), we were able to launch the first step of the software execution (construction of matrices needed for detection and identification of anomalies). For the second step, the detection and identification phase, we needed to choose some important parameters: the number of hashing functions required to consider the identification method successful, the number of Principal Components to be included in the *normal* subspace and the value of the threshold to be used in the detection phase.
For starting we left the number of hashing functions as set by the creators of the software, that is 4 on 8, bat later we modified the value to 5, in order to avoid many of the false alarms (as we will better explain in the following).
Looking to previous analysis performed on IP traffic with this type of Anomaly Detection method (for example in [1]), we observed that the number of PCs needed to perform an effective analysis is usually quite low (around 5), so we decided to perform the analysis varying the number of Principal Components from 1 to 10.
The threshold was the hardest parameter to set, because the software performances are heavily dependent on this value, and the appropriate value for obtaining an effective detection of anomalies is deeply related to the type of trace used. Trying the software on an IP traffic trace and on VoIP traces of different duration (a few days, one week and finally one month, the trace described before) we discovered that the appropriate threshold's values changed a lot from one trace to the other. A the end we decided to perform the analysis on the one-month trace using thresholds from 0.000001 (the minimum value permitted by the software) to 0.0001 (we observed that thresholds over this value led to no detections in that trace).
In the following chapters we will see in detail the impact that these parameters have on the results of the analysis.

4.3 Number of hash functions

The number of hash functions that we require to consider a candidate responsible user correctly recognized, has, of course, a major impact on the performances of the identification phase. While the previous phases (until the detection of anomalous time-bins) are not sensible to this parameter, the number of time-bins for which the software is able to produce a candidate responsible user is strongly dependent on the number of hash functions required.
The total of hash functions used by the software is eight, so in the first trial we chose to set this parameter equal to four: an anomalous user was considered as a candidate suspect if at least four on eight hash functions pointed out as a candidate aggregate a list of users in which that specific user was present.
To analyze the performances of the software with this setting we collected the list of users pointed out as a candidate responsible for an anomaly in one time-bin at least one time, with a wide range analysis that used a fixed number of PCs (varying from one to ten) and a range of thresholds from 0.000001 to 0.0001. The resulting lists were different changing the number of Principal Components used (we'll

discuss this in a following chapter), but a manual check of the detected users' behavior led to the observation that on a number of detected users varying from 160 to 260 (dependent from the PCs number) an average of 80% of them were false-alarms. An idea to filter a good number of false alarms could be to select only the users that, on 100 thresholds used, were identified more than a chosen number of times, but it would not be an effective solution, because it could only work for an analysis that uses a range of thresholds and not for a single threshold analysis.
So we decided to perform the same test with a number of required hash functions equal or superior to 5 on 8. With this choice we had, of course, a lower number of users recognized by the identification mechanism, but analyzing the list of users pointed out at least one time on 100 thresholds (varying the PCs number from 1 to 10), we observed that in a set of approximately 30 users (with small variations with different number of PCs) only an average of 10% of them were false alarms.
In the following analysis so we decided to keep this parameter set to 5.

4.4 Threshold's value

The value of the threshold used in the detection phase is a critical parameter for using the software for anomaly detection. We can pass it as an input to the software, in a form of float value, with a maximum precision of 10^-6 (the minimum value accepted, beside zero, is 0.000001). To analyze as deeply as possible the software performances we just had to choose the range of thresholds on which testing the system. After some initial trials we chose to use a range of thresholds from the minimum value accepted by the software (0.000001) to a maximum value of 0.0001, scanned with a step of 0.000001; this way we performed the analysis on a set of 100 thresholds. This choice was acceptable for the very specific trace we took in account (thresholds over 0.0001 led to no detections at all), but not in general for other traces (with traces that covered only some days of traffic the range of thresholds in which the algorithm led to detections was wider). We didn't analyze in detail this phenomenon, but this could be an issue if trying to apply this software as an Anomaly Detector on traces with different types of data and lengths.
Launching the analysis on the range of thresholds previously described (from 0.000001 to 0.0001) led to the following consideration about the results:

1. High values of the thresholds lead to no detection at all: no time bin was detected as anomalous and, of course, no user was identified as anomalous (the identification phase works only on the time-bins signaled as anomalous in the detection phase)
2. Starting from the maximum value of the threshold and gradually lowering it the software starts to detect anomalous time-bins, but no responsible user is found (the identification phase is not successful). As we said we are not interested in this type of “detections” so we are not analyzing this phenomenon in detail, but we can observe that these time-bins (the “first” ones detected, so we can consider them the most-anomalous for the algorithm) are the ones with really low quantities of global calls; on these type of anomalous hours the identification phase is most likely to fail, because there is not really one responsible user for an hour with a low value of general calls.
3. Continuing to lower the threshold leads to the first “real” detections, in which the identification phase produces results and the software starts to point out anomalous users. We can consider

these users as the ones with the “most anomalous” behavior in the trace. We'll discuss in a following chapter how the number of Principal Components used impacts the type of users detected.

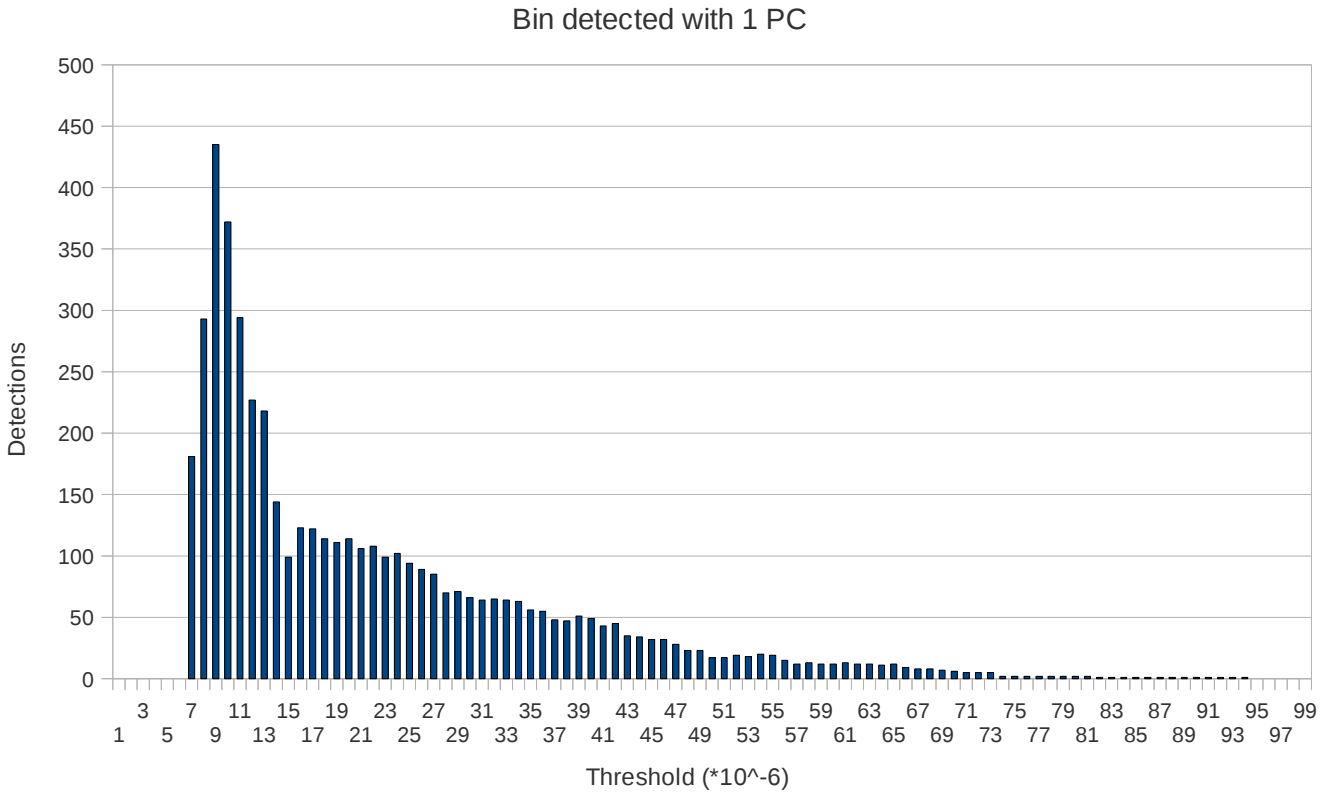
4. Lower values of the threshold lead to bigger sets of anomalous users identified, as we could expect. In general the “most anomalous” users (the first detected, with higher thresholds) continue to be detected, and other users start to be detected. But observing the results we can see that there are often exceptions for this rule: some anomalous users are detected only among a range of thresholds, and using values of the thresholds out of this range (even lower) we are not able to detect these users. Because of this phenomenon the performances of the software are heavily dependent on the value of the threshold: using a single threshold (even a very low one) can't guarantee the detection of all anomalous users in the trace, and is always suggested using a range of thresholds.
5. Really low values of the threshold lead to marking all the time-bins as anomalous, but not to the identification of any anomalous user. This phenomenon is related to the identification procedure: using very low threshold's values lead to the fact that there isn't any aggregate that, removed from the computation of the squared prediction error, brings this statistic under the threshold (since its value is too low), so the identification method is unsuccessful.

Analyzing in more detail the type of anomalous users identified by the software, we can also say that most of the false alarms come out with low values of the threshold. This phenomenon is quite predictable, because a low threshold's value lead to a higher number of time-bins detected as anomalous, and since the identification method tries to point out a responsible user for each anomalous time-bin there could be two types of errors in the execution:

- A time-bin with no anomalies is detected as anomalous, so the identification phase (if successful) can only point out a false alarm. This type of error is most likely to happen with low values of the threshold.
- An anomalous time-bin is correctly detected but the identification phase fails to identify the correct responsible user. This type of error can happen with any value of the thresholds, but as a lower value of the thresholds leads to a higher number of anomalous time-bins detected the probability of finding this type of error is higher with low values (even if the error ratio in the identification phase is the same).

If we wanted to select a hypothetical “best” threshold for launching the analysis, we should select the one that leads to the bigger set of anomalous users identified or the one in which we have the bigger number of time-bins in which a responsible anomalous user is identified; these thresholds could be different (for example there are cases in which a high number of anomalous time-bins are detected but in the majority of them the responsible user identified is part of a small set, while in other cases a small number of anomalous time-bins are detected but for each of them a different responsible user is pointed out by the software).
We can observe how these “best” threshold's values are dependent from the number of Principal Components used in the analysis on the following graphs (Figure 5 – Figure 10).

Figure 5



In figure 5 we plotted the number of time-bins in which the software pointed out a candidate responsible user, for each threshold used in the analysis, with a number of principal components equal to 1. We can observe that this number changes very quickly with small variations in the threshold's value; that's why it's difficult to select a single threshold for using the software and is always suggested to use a set of thresholds.

Figure 6 is the same graph, but plotted from the results of an analysis performed with a number of Principal Components equal to 10. We can observe that the best value of the threshold changes with the number of PCs (with intermediate values of PCs we had intermediate values of the thresholds), becoming smaller with a higher number of Principal Components.

Figure 6

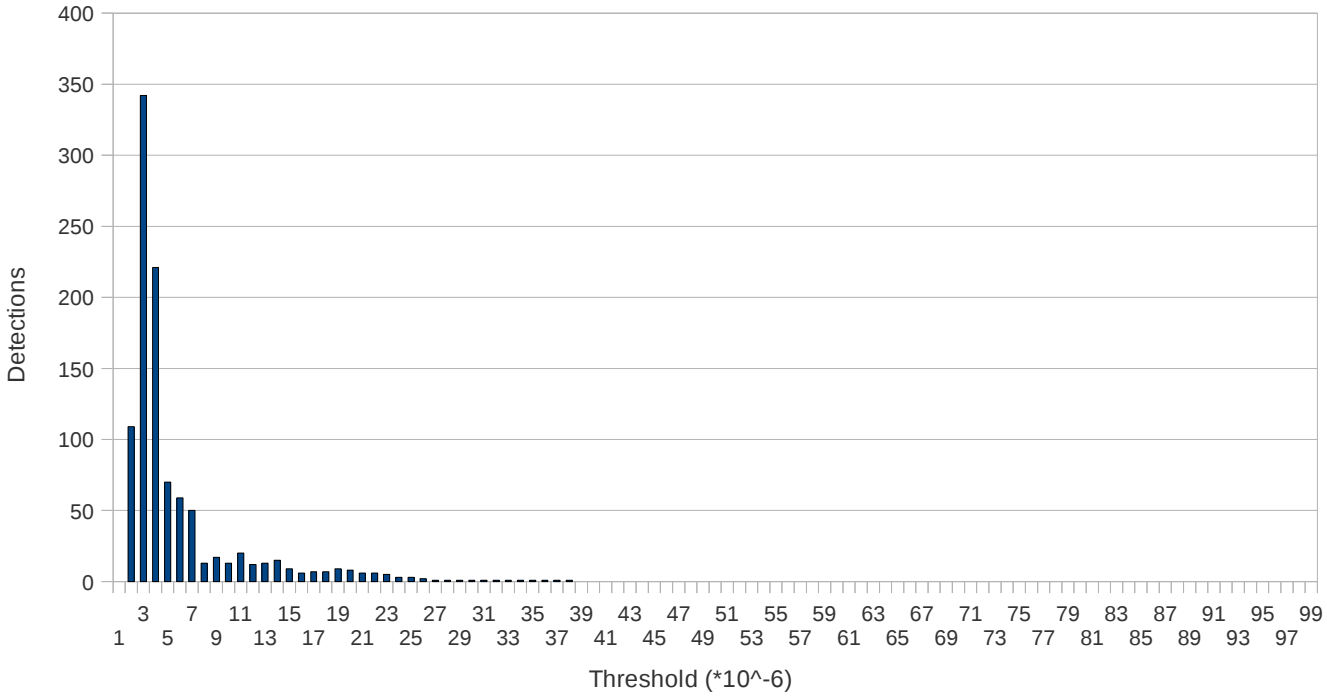


Figure 7

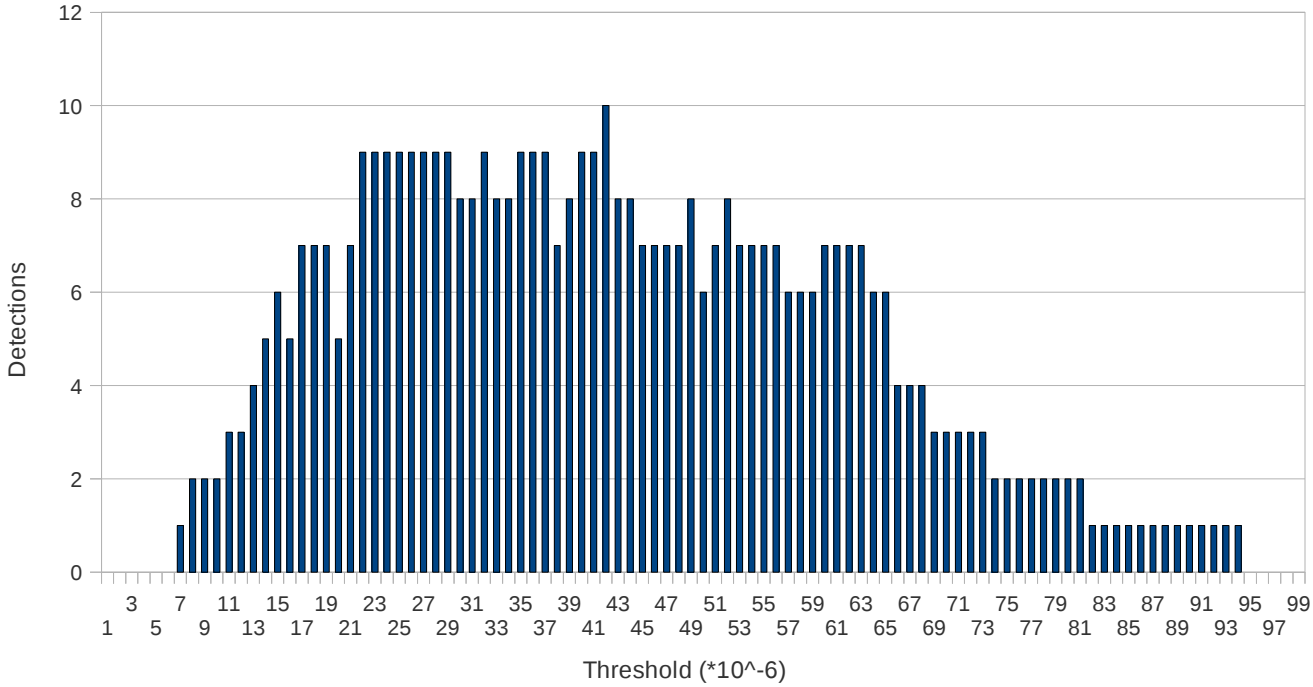
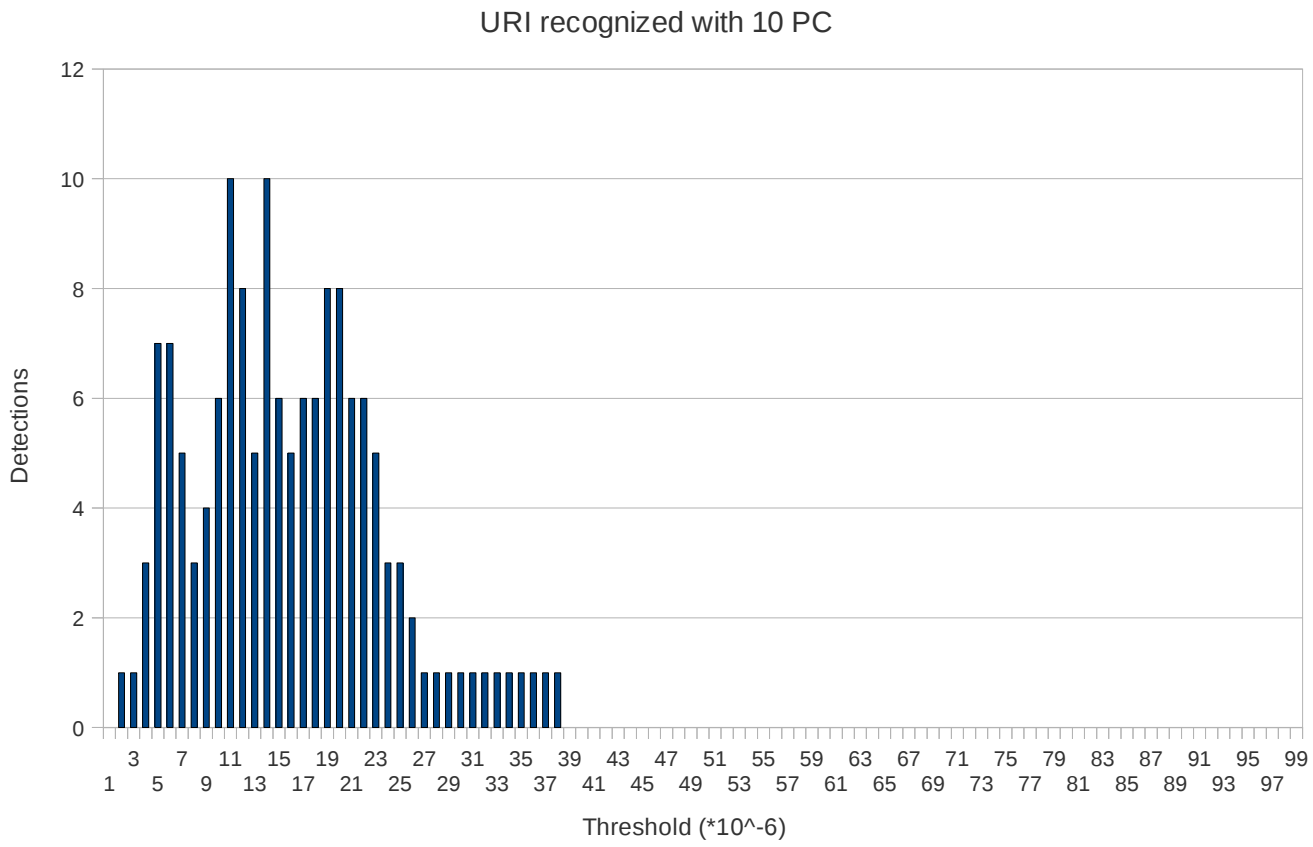


Figure 7 shows the number of different anomalous users identified by the software for each value of the threshold used in the analysis, using a number of PCs equal to 1. We can observe that the value of the "best" threshold in this case is different from the one of the best threshold in terms of time-bins detected. Besides the number of users identified is less dependent on the threshold's value, but it's still dependent on the number of PCs used, as we can see from Figure 8, that plots the results in case of 10 PCs used.

Figure 8



In general we can say that a higher number of PCs leads to smaller value of the "best" threshold (in both cases), and also to higher dependence of the performances on the value of the threshold (the range of threshold's values in which the number of URI detected remains near the top one is smaller).
We can observe this tendency by plotting the "best" values of the thresholds against the number of PCs used in the analysis.

Figure 9

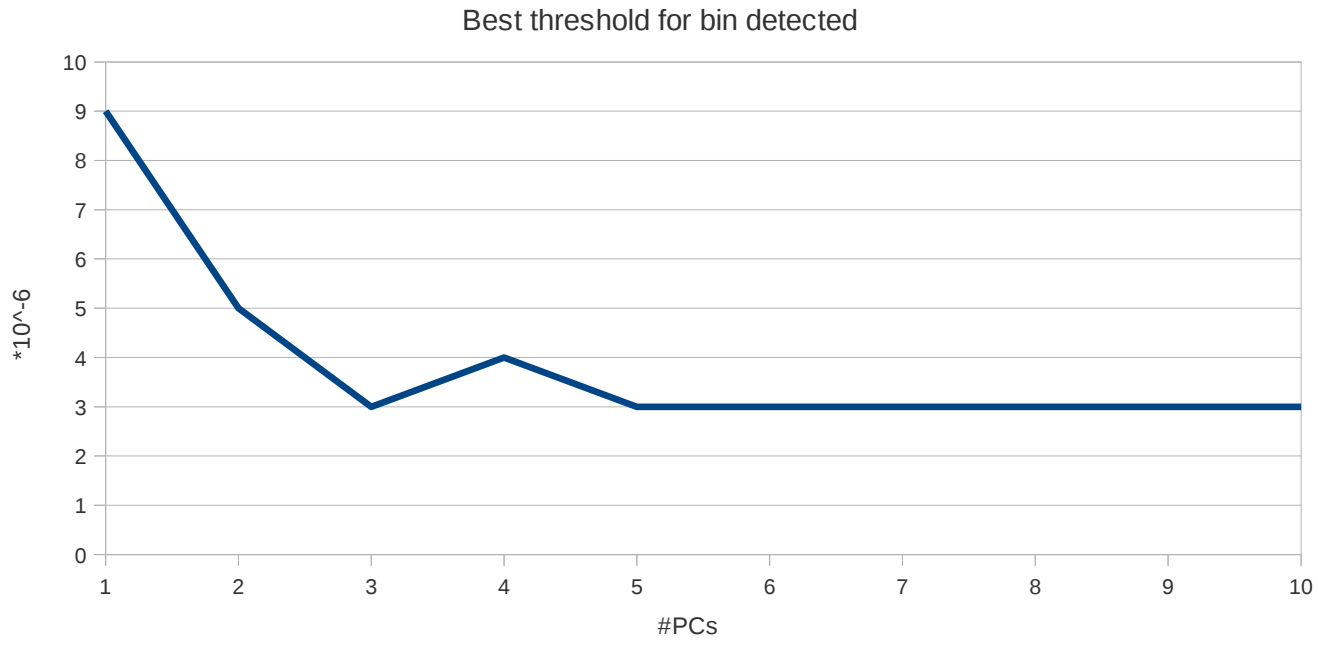
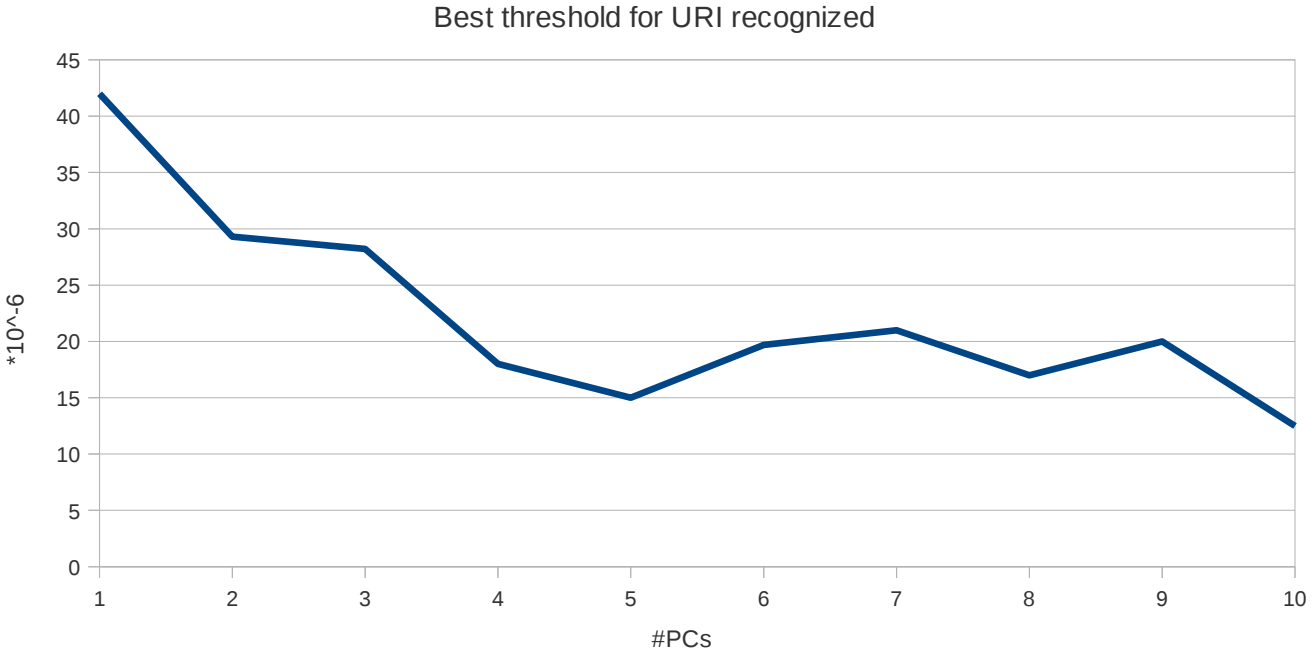


Figure 9 shows the values of the "best" threshold, in terms of quantity of time-bins detected, while Figure 10 shows the values of the "best" threshold in terms of number of different users recognized (because in this case the best results were obtained with more than one value of the threshold we plotted the mean value among the set of "best" thresholds, for each number of PCs).

Figure 10



In both cases we can observe how the best threshold's values tend to decrease with the increasing of the number of Principal Components used. This phenomenon is dependent to the fact that a higher number of Principal Components included in the *normal* subspace leads to the inclusion of the bigger anomalies (in terms of traffic size) into the normal behavior of users in the trace. For this reason (we'll discuss this in the following chapter) the number of anomalies identified in the trace is lower, and so is lower the number of detections and the value of the threshold needed to find the first detections.

4.5 Number of Principal Components

The number of Principal Components included in the *normal* subspace influences the way the algorithm tries to model the standard behavior of the traffic. Using a higher number of PCs leads to a more complex model of the normal traffic, including most of the variability in the trace, but it can lead to the inclusion, in the normal subspace, of the bigger anomalies (in terms of traffic volume and variability). On the other side a smaller number of PCs leads to a less accurate model of the normal behavior, that means the variations from the "normal" behavior needed to be considered anomalous are smaller.
In order to analyze the impact that the choice of this parameter has on the type of anomalies detected by the software we decided to focus on the "most-anomalous" users, the ones detected with the highest values of the threshold (starting from the maximum threshold's value and gradually lowering it these are the first users recognized).
In the following graphs we will compare these first detections in two cases: in an analysis that uses 2 Principal Components and in another one that uses 10 PCs. The graphs plot the ratio among the number of calls attempted by the user detected and the average number of calls for active users (1.57), in each time-bin of the trace, highlighting the time-bin in which those users were detected as responsible of an anomaly.
Figure 11 to 13 show the "most-anomalous" users detected with two principal components. As we can see the software detects high peaks of more than a thousand of calls, for user with a high volume of traffic in all the trace; we call this type of users "top-users" because are in the list of users with the highest number of total calls in the whole trace.
Figure 14 to 16 show the "most-anomalous" users detected with ten principal components. First of all we can observe how the threshold for which we have the first detections becomes smaller (with 2 PCs the first anomalous user identified is with a threshold of 0.000070, while with 10 PCs we have the first detection with a threshold equal to 0.000039); we already observed this phenomenon in the previous chapter, and we can motivate it with the fact that the inclusion of the biggest anomalous users (the top-users) into the normal subspace (that can happen using a high number of Principal Components) makes the detection and identification process more difficult, other than removing from the anomalous subspace some of the anomalous user that are most likely to be detected.
If we focus on the behavior of the "most-anomalous" users detected with 10 PCs we can observe how they represent another kind of anomaly. Focusing on the first anomaly detected (Figure 14) we can see how the software detected a high peak of calls from an user that is quite active, but not as much as the "top-users" detected with 2 PCs. The idea becomes more clear if we observe the second user detected (Figure 15), that is not a "top-user" at all: it's an user that remain silent (no attempted calls) for the

whole duration of the trace, except for one single time-bin, in which there is a peak of 170 calls. A number of calls of that order wouldn't be considered an anomaly for a “top user”, but is absolutely anomalous for an user with this type of behavior.
If we would continue observing the users detected among the whole analysis (with thresholds from the minimum to the maximum value), we would confirm this theory, observing that a low number of Principal Components tends to point out volume anomalies (top-users), while a high number of Principal Components is better for detecting “behavioral” anomalies (for example single-peaks like the one in Figure 15).
As we said before a higher number of Principal Components also leads to lower values (and ranges of values) of the threshold that permit the correct identification of anomalous users, other than a globally lower number of anomalous user identified (merging all the anomalous users detected with any threshold's value).

Figure 11

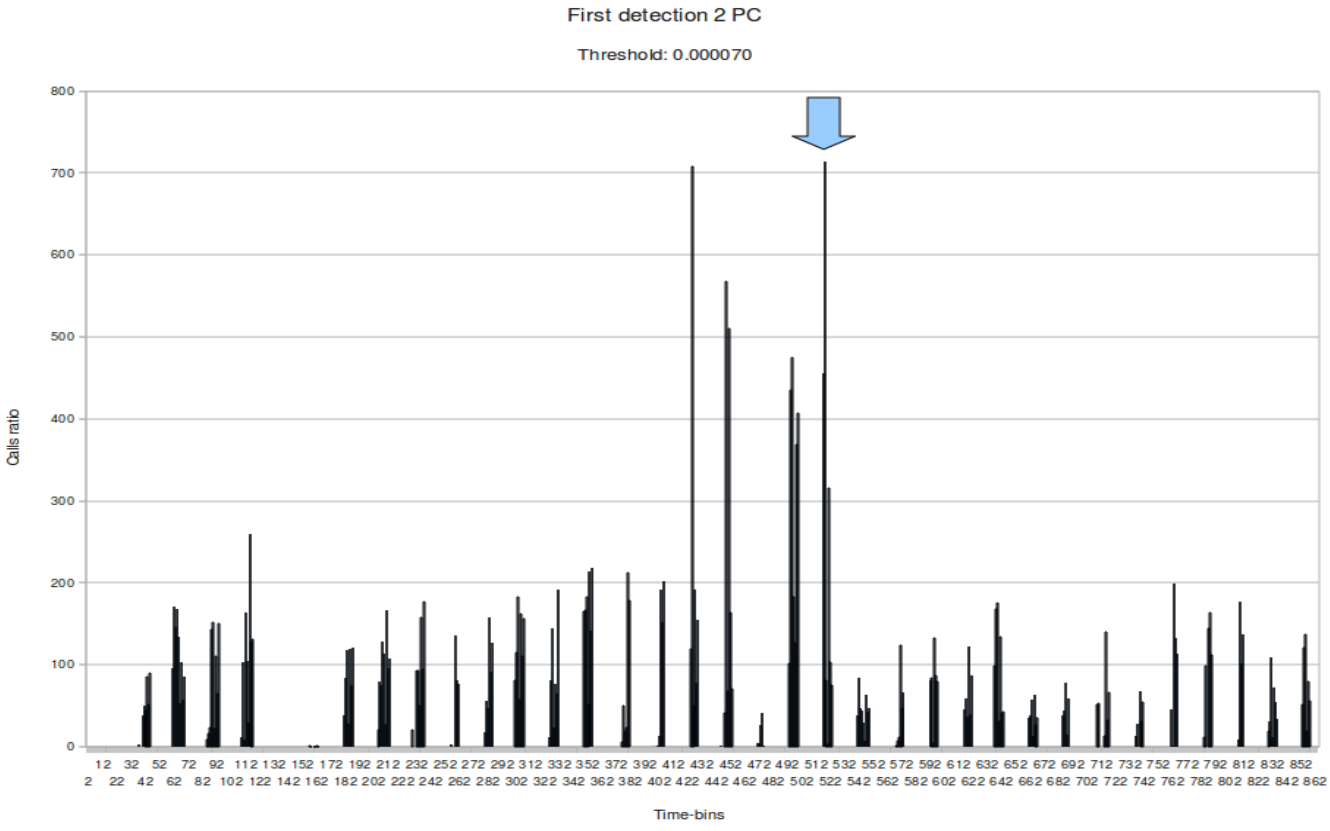


Figure 12

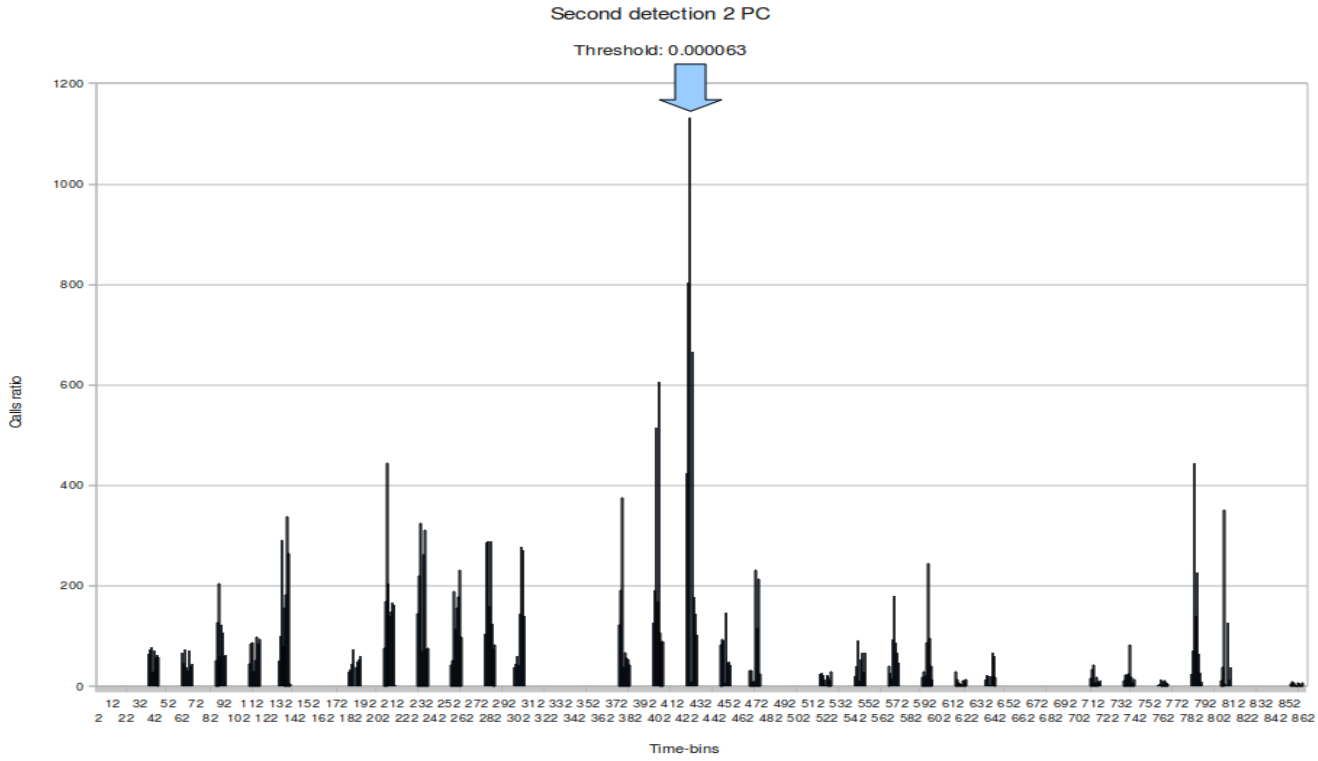


Figure 13

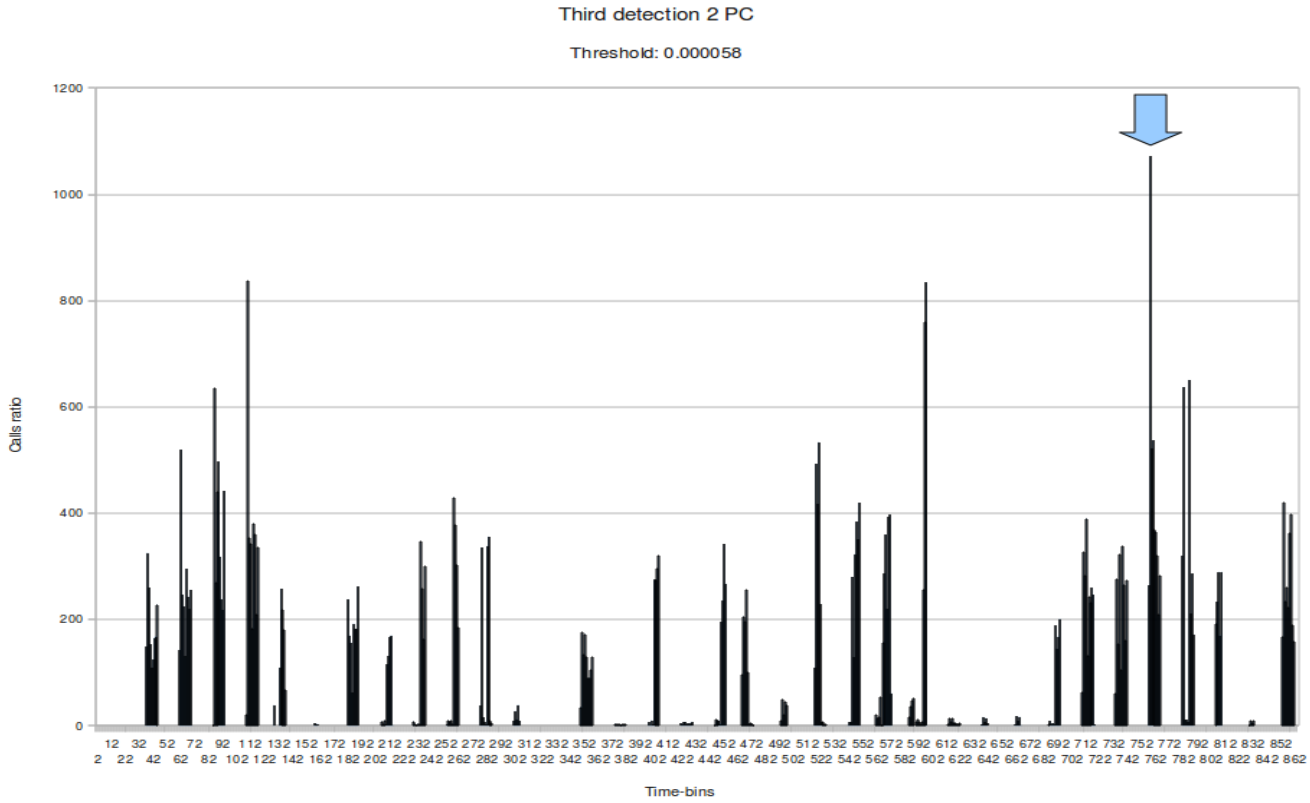


Figure 14

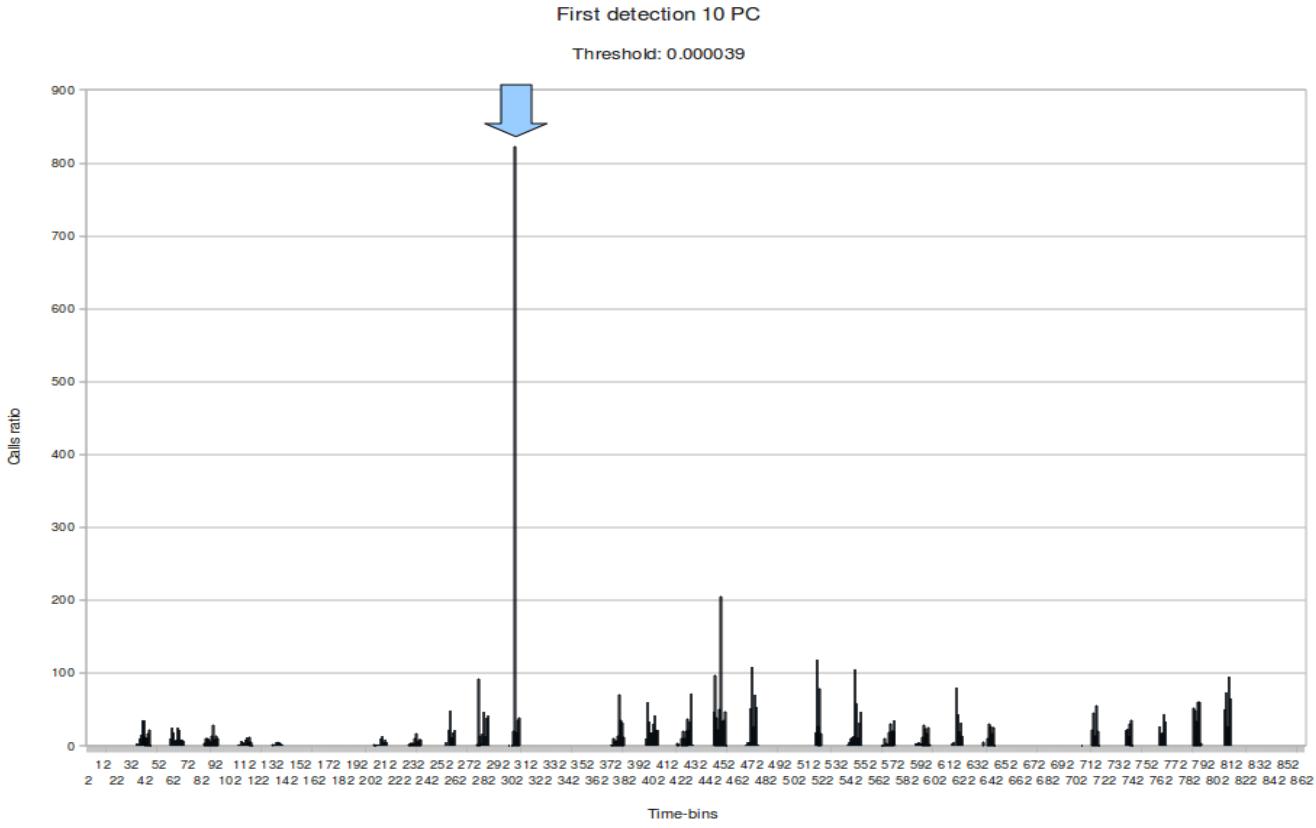


Figure 15

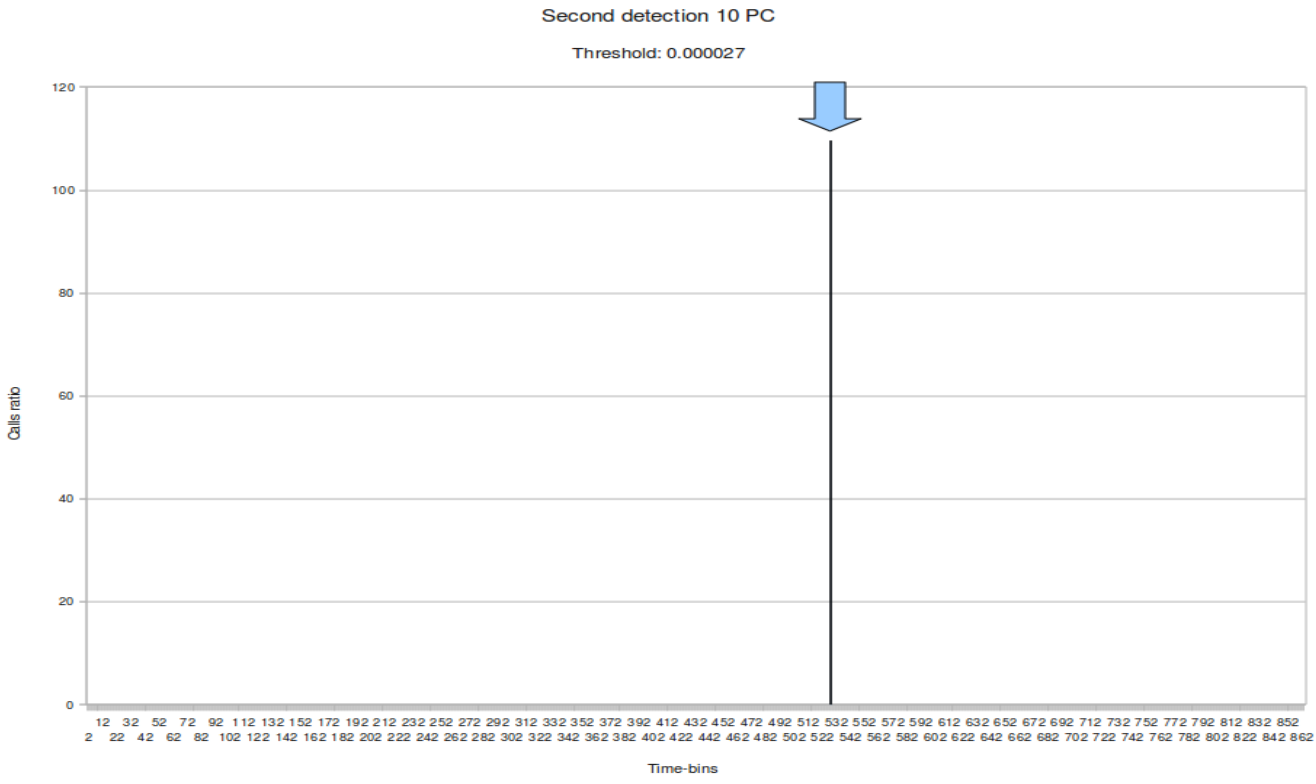
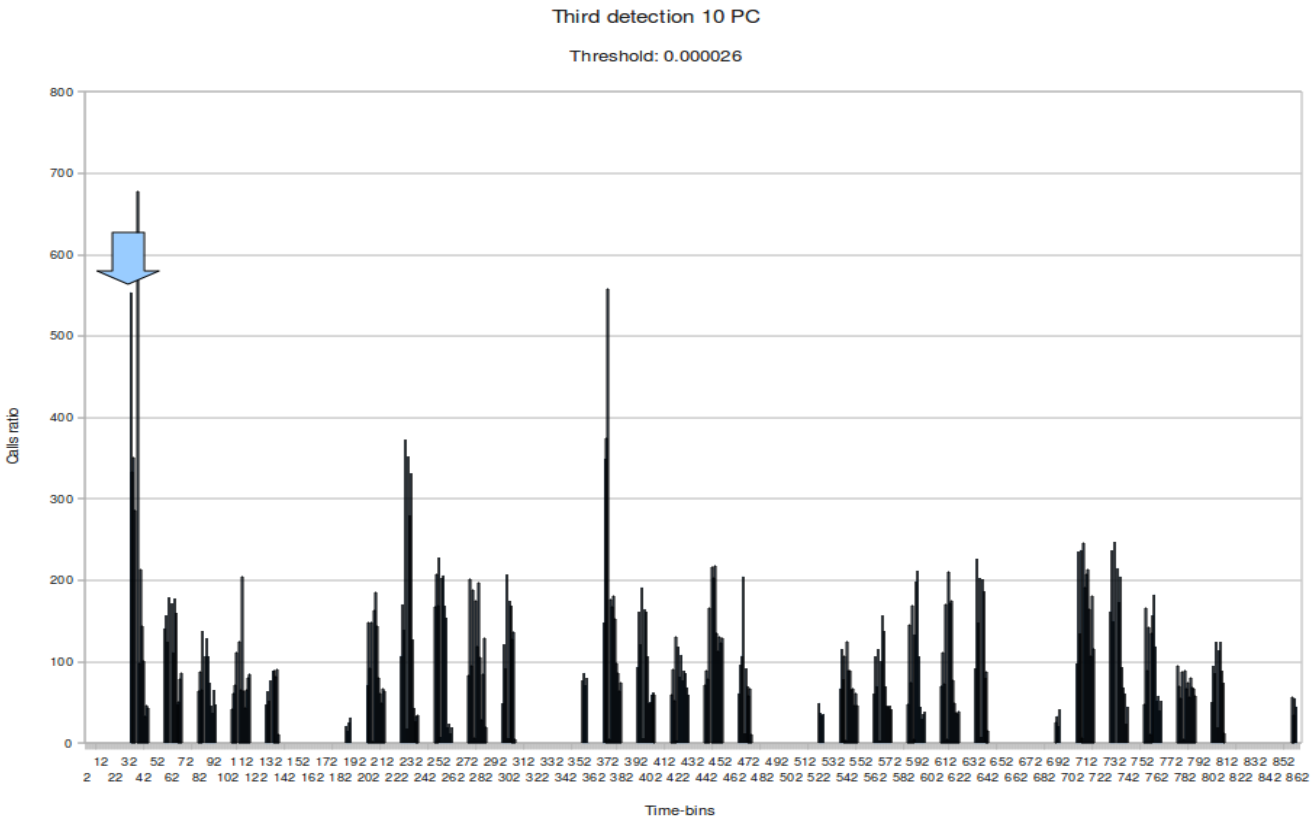


Figure 16



4.6 Considerations on the results

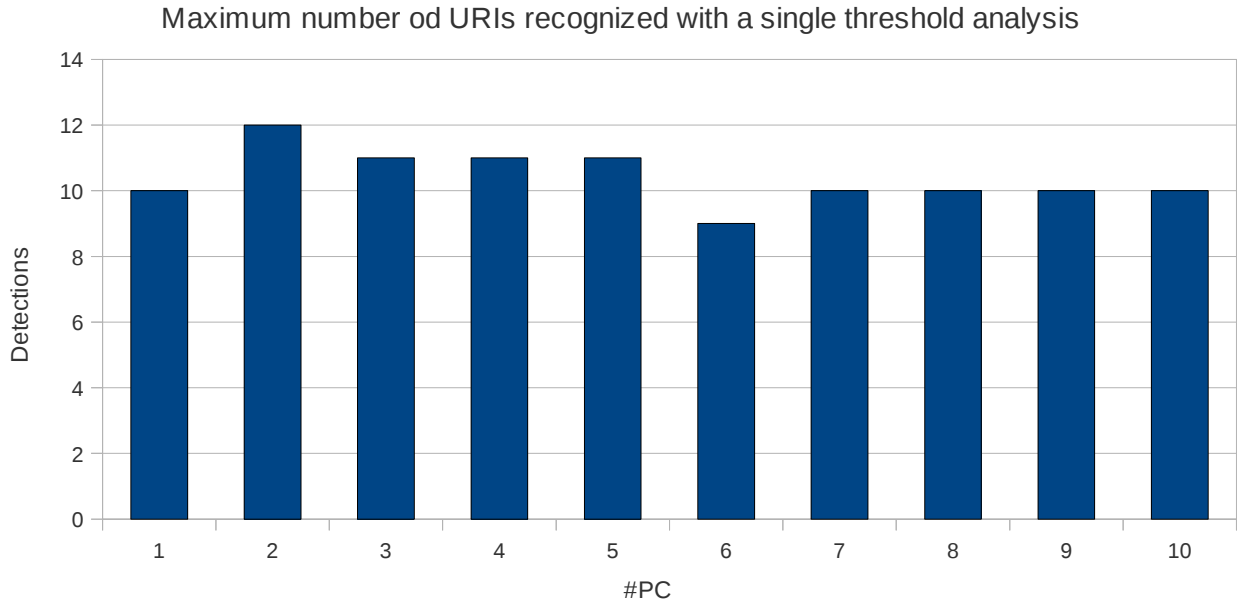
From what we observed on the results of this first analysis we can start making some considerations on the possibility of using this software as an Anomaly Detection System for VoIP traffic.

- The first issue, when trying to use this software on a certain trace, is the difficulty of choosing parameters. We can decide in advance a number of hashing functions equal to 5, because we have seen that using 4 hashing functions the ratio of false alarms in the results was far too big. We can select an appropriate number of Principal Components according to the type of anomalous users we want to detect (low for identifying top-users, higher for identifying behavioral anomalies). The biggest problem is the selection of an appropriate threshold: we saw that the kind of results we obtain are heavily dependent from the threshold's value, and we are not able to decide a single “best” value of the threshold, because the range of thresholds that we want to use to have good performances are dependent on the trace we use and on the number of PCs. For this reason, when we have to analyze a new trace, we have to try a wide-range of thresholds, with a waste of time compared to an hypothetical case in which we know an exact threshold's value to use. To better understand this we can compare the results shown in Figure

17 with the ones in Table 1, observing that the amount of users we can identify using a single threshold (the “best” one) is around a third of the amount of users we can identify using the full range of thresholds.

- The second issue is the possibility of tuning the software in order to find different kind of anomalies (by selecting an appropriate number of Principal Components). These feature is quite useful if compared with other kinds of AD techniques, that mainly focus on volume anomalies, and makes this software quite flexible for different kinds of uses.
- The third issue is a congenital limit of the algorithm: the impossibility of detecting more than one anomalous user per time-bin. That is a big trouble, compared to other type of AD techniques, because it imposes a limit to the number of anomalous users detectable by the system: we can only identify a single responsible user for each anomalous time-bin (and in a trace not every time-bin can be considered anomalous), so we have not only a limited number of users detectable, but the most anomalous users can “cover” the less anomalous (if different users have peaks in the same hours, the one with the smaller peaks will always remain hidden behind the one with the bigger ones, that will always be identified as the responsible for the anomalies in that time-bins).

Figure 17



In the following table we'll try to resume the result of this analysis, writing the total number of anomalous users identified with a “wide thresholds” analysis (from 0.000001 to 0.0001), for each number of PCs used, and trying to classify the users identified as “top-users” (in the list of the top-200 users in terms of number of attempted calls in the whole trace) and not “top-users”. The users classification is a complex procedure, because there is no automatic way of doing it, and there is no clear border between false alarms and “not-top” anomalous users (for example a single peak of 30 calls in one hour could not be considered anomalous, even if it's a sort of behavioral anomaly), so this table can be influenced by this uncertainty.

Table 1

# PCs	Total URIs detected	Top-users	“Not top” anomalous users	False alarms
1	30	18	8	4
2	36	16	16	4
3	33	14	17	2
4	31	13	16	2
5	31	13	16	2
6	29	13	16	0
7	31	14	15	2
8	32	14	16	2
9	33	15	15	3
10	30	14	16	0
Union	60	21	21	18

From the table we can conclude that if we are interested in just finding “top-users” the best number of PCs to select would be 1, while if we are interested in other types of anomalies an higher number would be better (the highest number is with 3 PCs, but using numbers of PCs bigger than it can lead to the detection of lower-volume anomalies, as we have seen for the first detections with 10 PCs, even if in a smaller number).
By the way the intrinsic limit described in the 3rd point leads to the detection of only a part of the anomalies present in the trace, so we decided to perform another type of analysis, to avoid this problem, as we'll see in the following part.

5. Multiple round analysis

5.1 Type of analysis

To avoid the biggest limit of this kind of algorithm, the impossibility of detecting more than one anomalous user per time-bin (that leads to the coverage of some anomalies behind bigger ones), we decided to try a different type of analysis. Since we were interested in finding a list of anomalous users, more than a list of anomalous time-bins, we decided to perform a multiple round analysis, using in each round a modified version of the trace, in which the biggest anomalous user identified in the previous round was canceled from the trace.
In detail, we decided to follow this procedure:

1. We took the same trace as before and we performed the first step of the analysis (matrices creations), that corresponds to the first three phases of the algorithm.
2. We launched the second step of the analysis (detection and identification), using a fixed number of Principal Components and starting from the maximum threshold's value (0.0001).
3. We repeated the second point, lowering each time the threshold used, as many times as the identification phase starts to point out the first (or the first set of) anomaly (anomalies).
4. We stopped the execution of the analysis and we removed from the trace used every entry corresponding to the user (or set of users) identified in the previous point.
5. We restarted from point 1 using the new trace (without the anomalies detected in the previous round).
6. We repeated this points (one round) as long as the software continues to detect new anomalies (this procedure ends when we reach the minimum allowed threshold's value without identifying any anomalous user).

The idea is to remove in each round the biggest anomalies in the trace, to have the possibility of detecting, in the following rounds, smaller anomalies that could remain hidden behind them. Since different numbers of Principal Components used lead to different users identified we performed this analysis separately with different numbers of PCs (1,2,3,5,10).

5.2 Type of detections

A first survey on the results obtained with this kind of analysis can lead us to two types of classification: a classification based on the user's behavior and a classification based on the time-bin in which an user is detected as responsible of an anomaly.
Focusing on the behavior of the users detected we can use the same kind of classification we introduced in the previous part:

- Top users, users with a high volume of attempted calls among the whole duration of the trace, with peaks that can reach up to more than one thousand of calls in one hour; they are part of the

list of the “top-200 users” in terms of number of total calls in the trace.

- “Not-top” users, namely users that are not part of that list; in this class an interesting (and wide) subset is composed by users that remain silent (or with a very low volume of calls) for the whole length of the trace except for a single time-bin in which they have a single peak of calls (over 60, that means more than one call per minute).
- False alarms, or users with a normal behavior (often less than 10 calls per hour, even in the most busy hours).

Focusing on the time-bins in which the users were identified as responsible for an anomaly we can classify each detection as:

- Correct detection: we identify an user in a time-bin in which it has its biggest peak, the higher amount of calls in one hour in his trace. This type of detection is more difficult with top-users, since they have several peaks of calls, but we consider a detection correct only if the higher one is detected.
- Wrong detection: the identification of one anomalous users in a time bin that is not correspondent to his higher peak of calls; this could be a medium peak of calls (not the highest one), a time-bin just after or before the peak (the software signals the discontinuity and not exactly the burst) or a time-bin with a normal behavior (probably an error in the identification method, with that users identified as responsible for an anomaly caused by another user).

These second classification of course in not applied to false-alarms.
In Figure 18-21 we can see some examples of this type of users/detections.

Figure 18

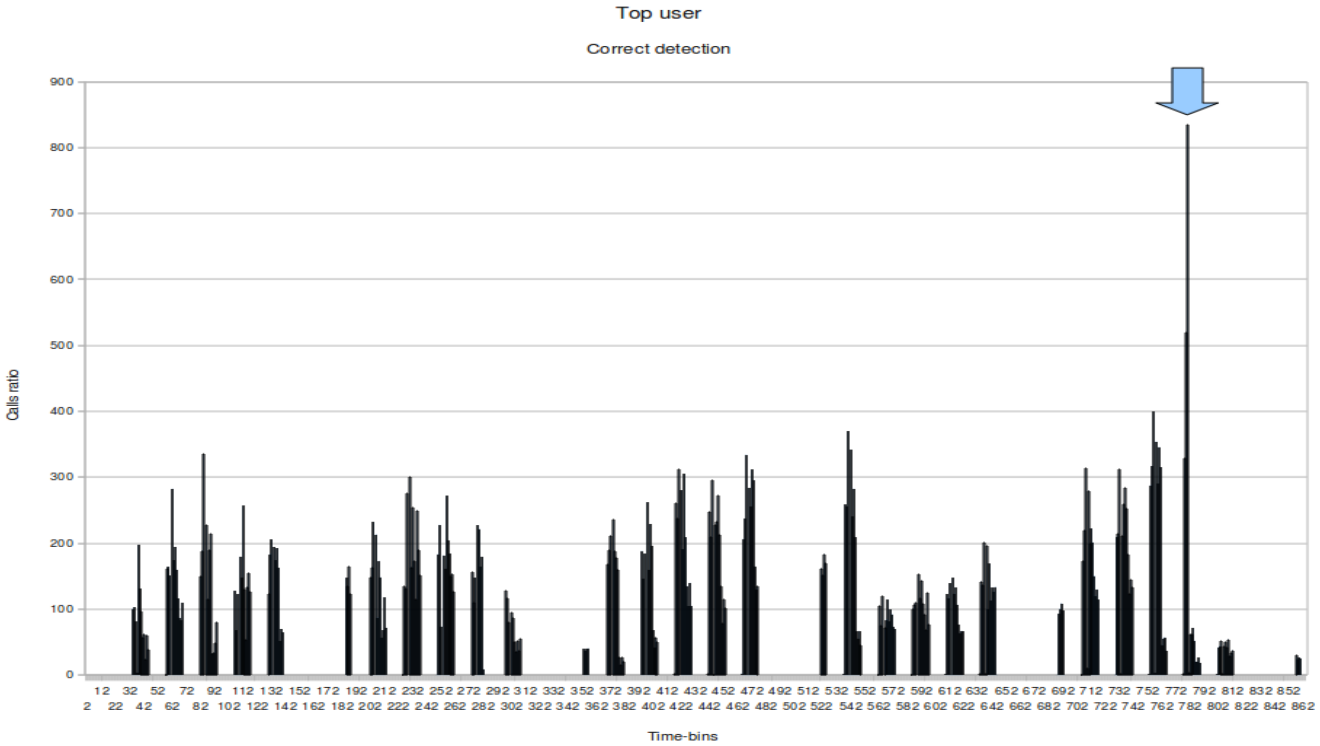


Figure 19

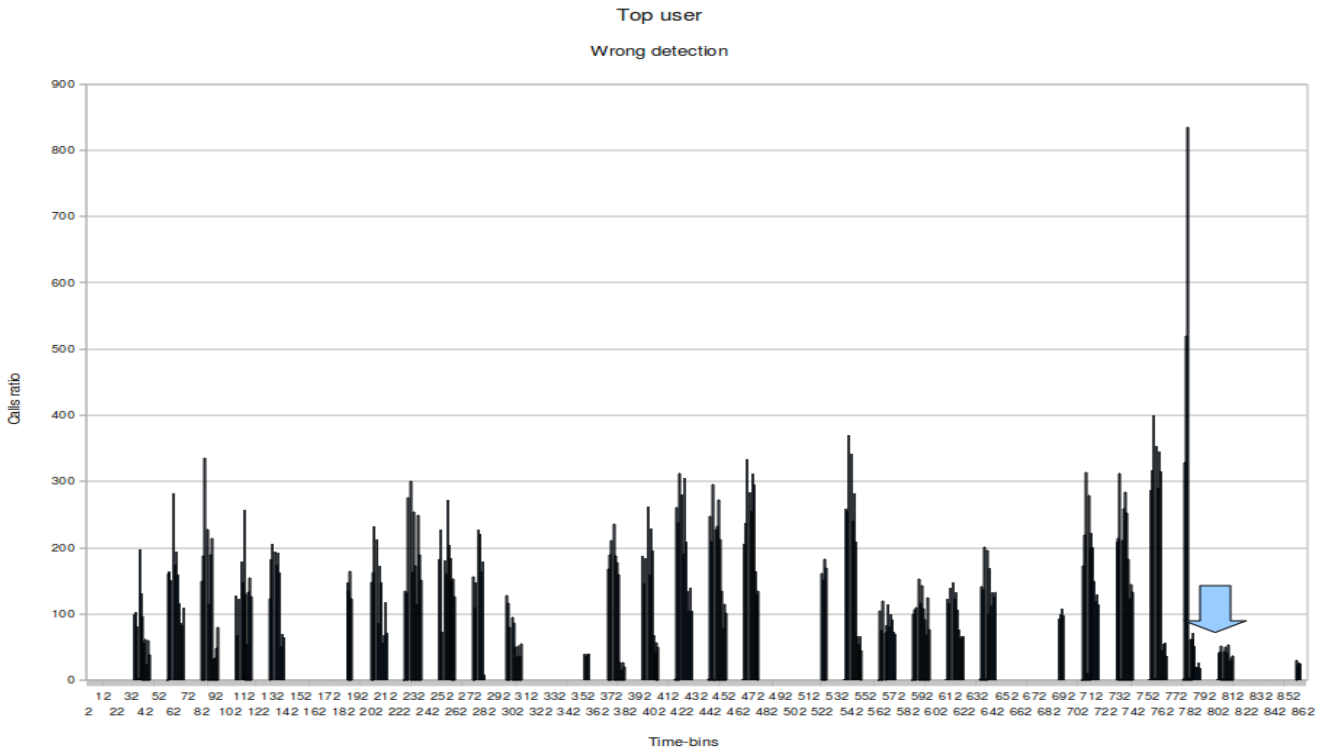


Figure 20

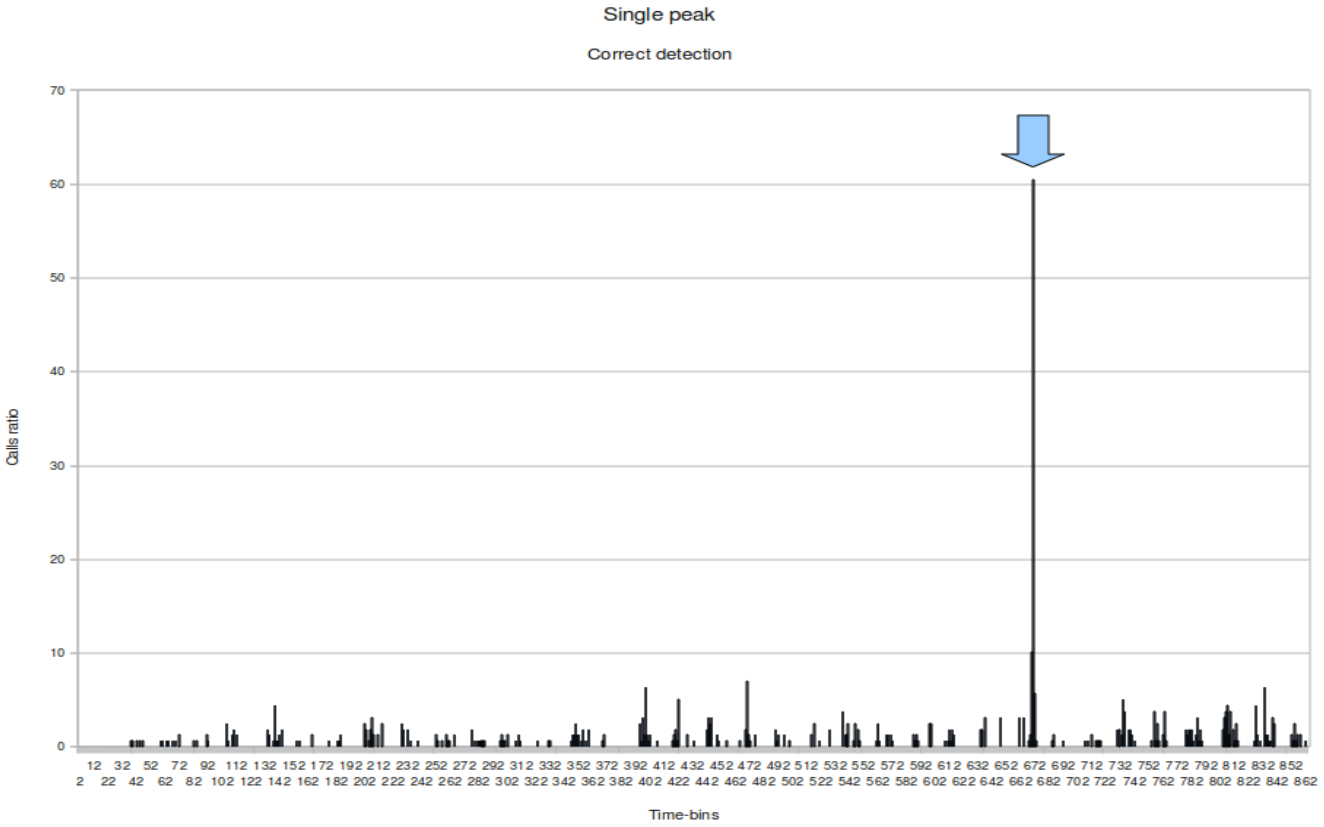
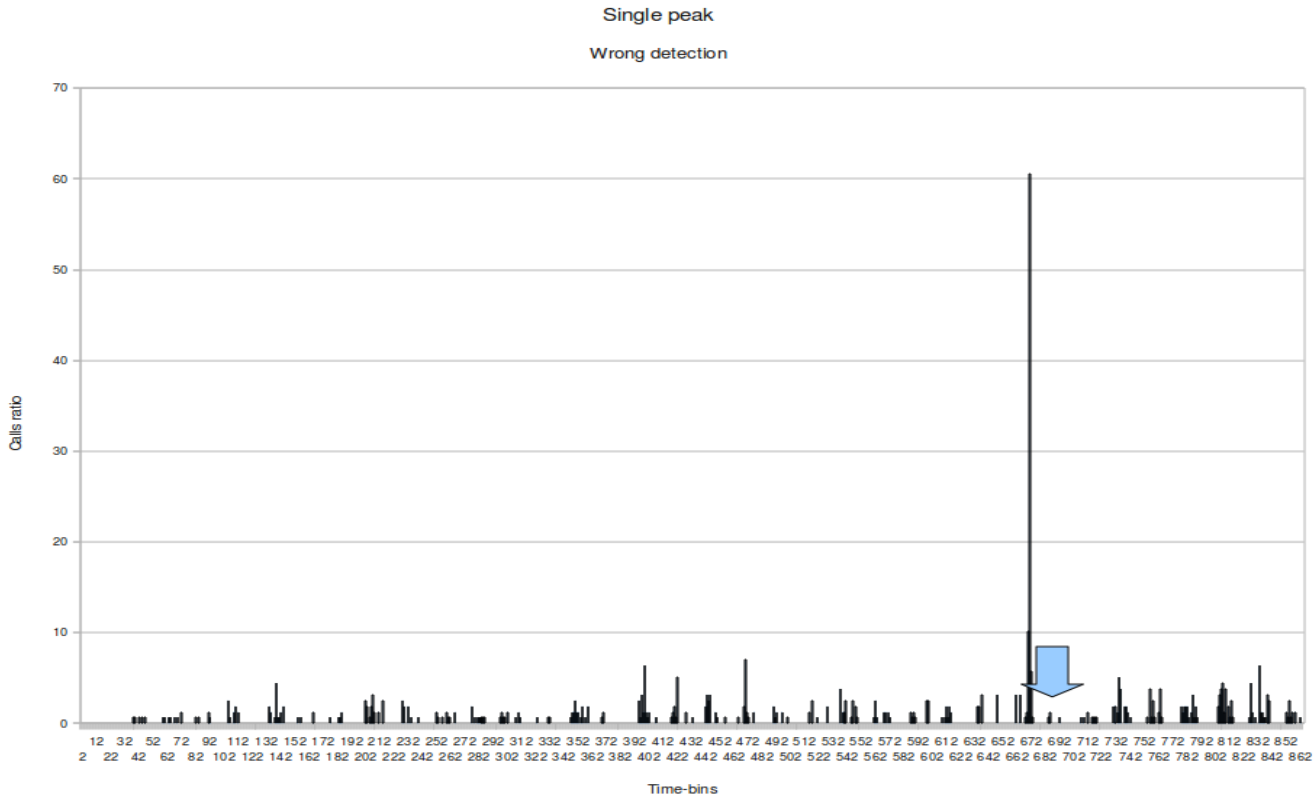


Figure 21



There are some cases in which a certain user is detect in multiple time-bins; in these cases we will classify the user as correctly detected if among these time-bins there is the one with the highest peak of that user in the trace.

5.3 Comparison with the first analysis

The first considerations we can make upon the results are about the comparison with the results we found in the previous type of analysis. As we expect with the multiple round analysis we are able to identify a higher number of anomalous users. We can observe this from Table 2, in which we compared the number of anomalous users detected with 2, 5 and 10 Principal Components in the two cases.

Table 2

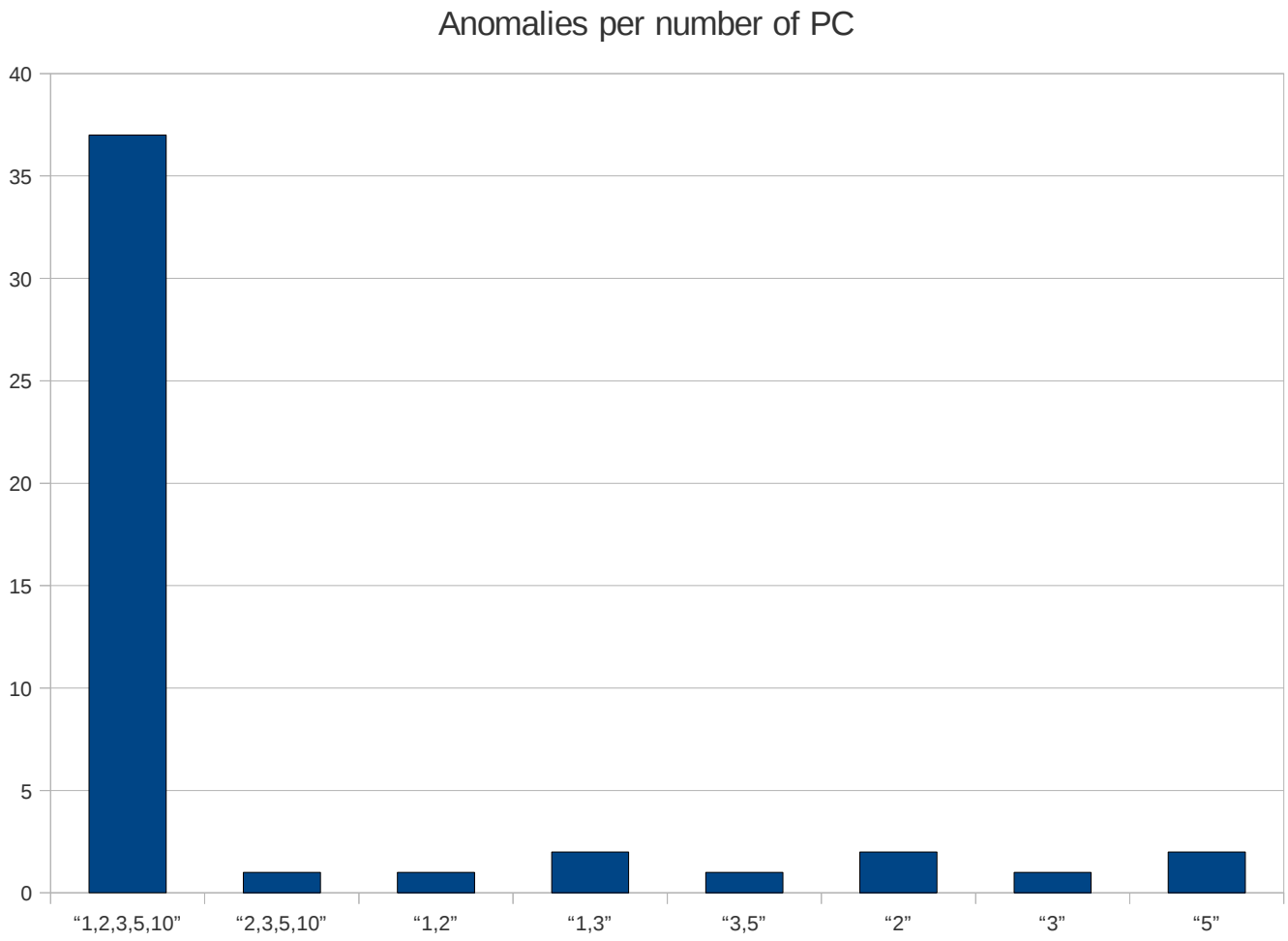
First analysis				**Second Analysis**		
Number of PC	*URI detected*	*False alarms*	*Exclusive detections*	*URI detected*	*False alarms*	*Exclusive detections*
2	32	4	2	45	0	13
5	29	2	0	42	1	13
10	30	0	0	37	3	7

We can see that with the second type of analysis we are able to identify around 30% more anomalies than in the previous case; we detect most of the anomalous users detected in the previous analysis (with 10 PCs for example we detect all of them) and a good number of new anomalies, that are exclusively detected with the multiple round analysis. There is still the tendency to detect more anomalies with a lower number of PCs (as a high number of PCs tends to include some anomalous users in the normal subspace), but we can observe a different distribution of false alarms: in the first type of analysis most of the false alarms came out with a low number of Principal Components (because in general we had more detections); in the second type of analysis most of the false alarms come out with a higher number of PCs. In general we have a lower global number of false alarms detected, under the 8% of the global number of anomalies detected, because of the way this type of analysis works, detecting each round only the most anomalous user still present in the trace. We obtain better performances (more users detected and less false alarms) with a low number of Principal Components because using few PCs permits us to leave the biggest anomalies (top-users with highest traffic volume) out of the *normal* subspace, detecting those anomalous users in the first rounds and removing them from the trace, so in the following rounds a smaller number of Principal Components is sufficient to correctly model the normal behavior of the traffic. A higher number of Principal Components (even if it's better to detect smaller anomalies in a single round analysis) is not the best choice for a multiple round analysis, since some of the biggest anomalies are not detected by the software (or at least not in the first rounds), so they remain in the trace, influencing the modeling of the *normal* subspace for many rounds, and leading to the detection of some "normal" users as anomalous (false alarms).

5.4 Results classification

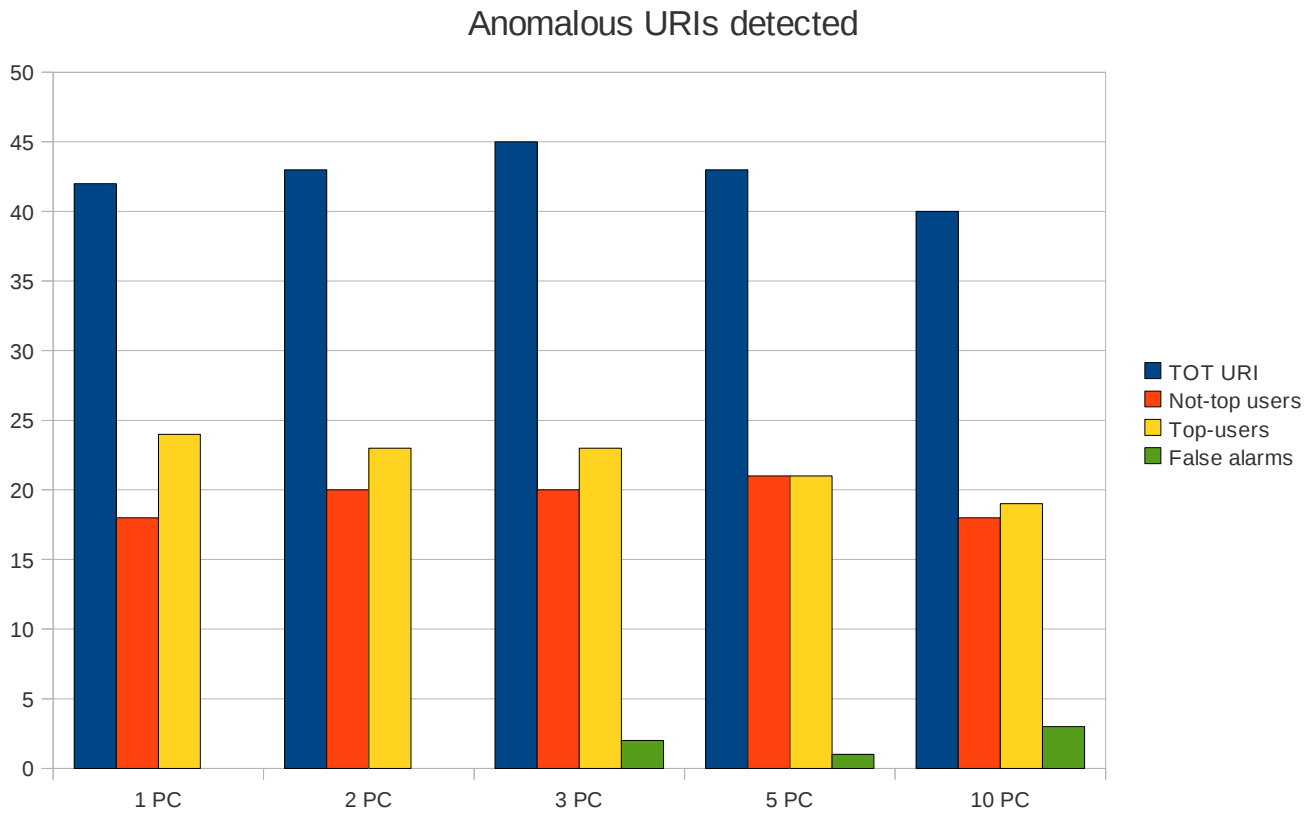
Just watching the number of anomalous users detected in the multiple round analysis, we can confirm that the best choice is a small number of Principal Components used, even if with this type of analysis the PCs number is a less fundamental parameter to set: almost all the users detected are pointed out with any number of PCs (as we said, we tried with 1,2,3,5,10), while just a few of them are detected with smaller sets of PCs. We can see this in Figure 22, in which we plotted the amount of users identified, over the sets of PCs number for which those identifications were possible.

Figure 22



Analyzing the type of users identified with each number of Principal Components we can split the users in the 3 categories mentioned before: top-users, “not-top” users and false alarms. In Figure 23 we can observe the total number of users identified with each number of PCs and the classification of those users.

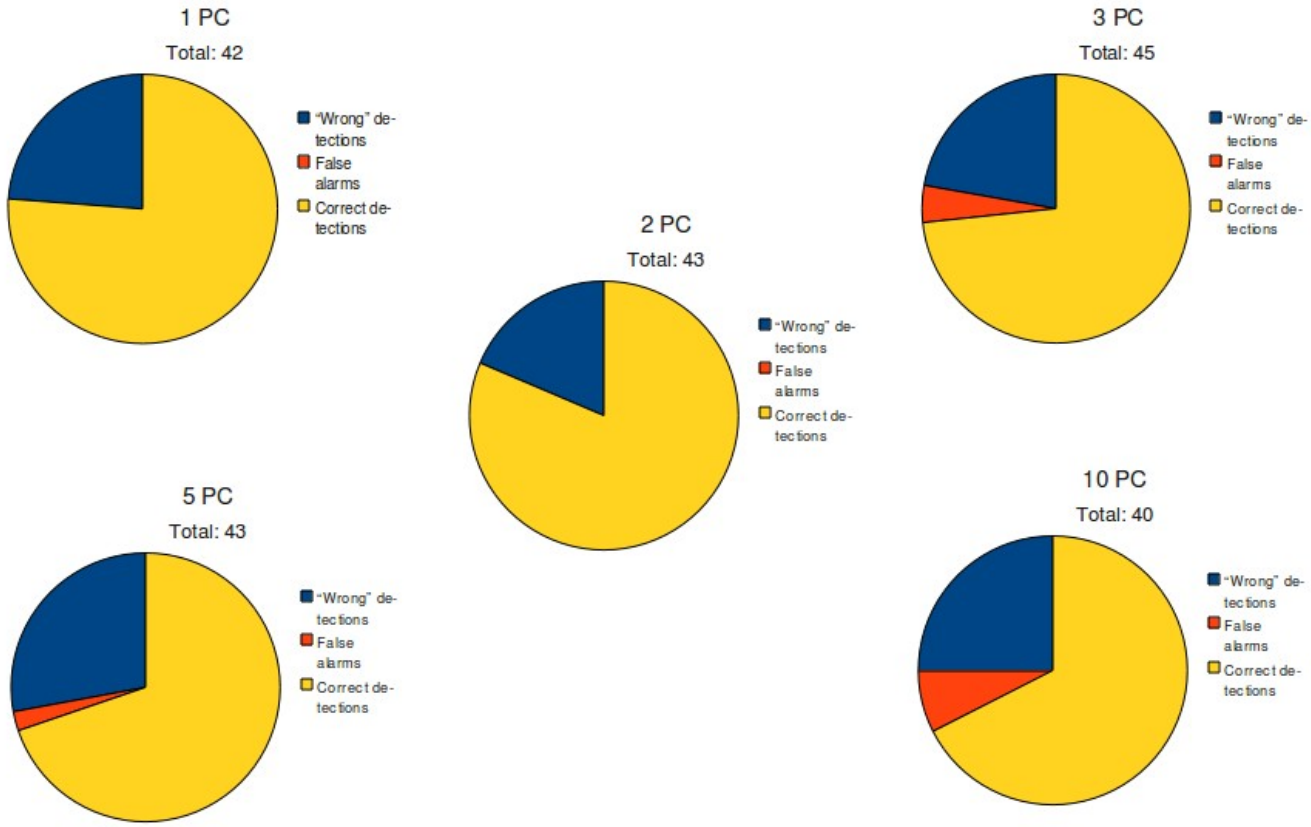
Figure 23



As we said in the previous chapter a higher number of PCs tends to point out more false alarms (for the reasons described before), but from this graph we can also notice how there is still the tendency, for lower numbers of PCs, to detect more Top-users than higher numbers of PCs; however there is not the same tendency we noticed in the first kind of analysis regarding “not-top” users, with higher numbers of PCs that in this case seem to not be the best solution for identifying this kind of anomalous users. To understand this type of results we must consider the way this kind of analysis is conducted: each round the anomalous users identified are removed from the trace, giving the possibility, in the following rounds, to identify anomalous users that could remain hidden behind the ones previously identified. A small number of PCs is more suited for this kind of process, since it permits the detection of high volume anomalies (top-users), that are the ones that mostly influence the global traffic behavior, making the modeling of the normal subspace more difficult. With a multiple round analysis launched with a small number of PCs (2 or 3 seems to be the best choice, since 1 is too few to model this kind of traffic) we obtain a consistent improvement in modeling the *normal* behavior of the traffic each round, because we are able to remove, in the first rounds, the biggest top-users in the trace; that permits also the detection of smaller anomalies (“not-top” users) in the final rounds, since the algorithm is capable of modeling more accurately the *normal* and *anomalous* subspaces even with a low number of PCs. On the other side a high number of PCs permits the identification of some “not-top” users already from the first rounds, but these kind of detections have not the same impact on the traffic modeling

improvement, and the following rounds can lead to less detections or to false-alarms detections.
Shifting to the analysis of detection types, we can classify each detection, as we said before, as correct (if a certain user is detected in a time-bin correspondent to its highest peak), wrong (if the user is anomalous but it's not detected in the moment corresponding to the biggest peak) or false alarm (if the user has not an anomalous behavior). If we perform this classification on the results obtained with each number of principal components, we can observe the results shown in Figure 24. There seems not to be a strict correlation with number of PCs and number of correct detections, but the best results are anyway obtained with 2 PCs.

Figure 24



5.5 *Comparison with VoIp Seal scores*

In order to compare the performances of this software as an Anomaly Detector with other kinds of AD techniques we decided to perform a comparison of the results of our software with the ones obtained using VoIp Seal [89], a system developed by NEC for detecting anomalies and in particular telemarketers in telephone networks.
VoIP SEAL's basic concept consists of combining several metrics that can be measured at the application level; users are then classified according to the combination of specific metric values.
The main analysis stage of VoIP SEAL analyzes each call separately, by considering application level semantics. For each call multiple risk assessment modules (RAMs) are invoked. Each RAM performs a certain statistical analysis on the call and returns an anomaly score (s1 to sn), ranging from zero (no anomaly detected) to one (very anomalous). In the current implementation, the individual scores are combined into an overall anomaly score for each call, by calculating a weighted sum of the module scores . Each calls is the classified as "Normal", "Suspicious" and "Anomalous" by means of this global score value; similarly, users are categorized as "Normal", "Suspicious" and "Anomalous" depending on the average score of all the calls they initiated.
We based the comparison between our software and VoIP SEAL on two different scores for each user:

- Average global score (the average global score value computed on all calls attempted by that user)
- Maximum global score (the global score of the "most anomalous" call attempted by that user)

We took the anomalous users detected using our multiple round analysis and we classified them using these two metrics. Observing the results we noticed that these two AD techniques use different concepts of what is considered anomalous: in Figure 25 and 26 we can see that the majority of users identified by our software have a very low score, that means these users are not considered anomalous by VoIP Seal.

Figure 25

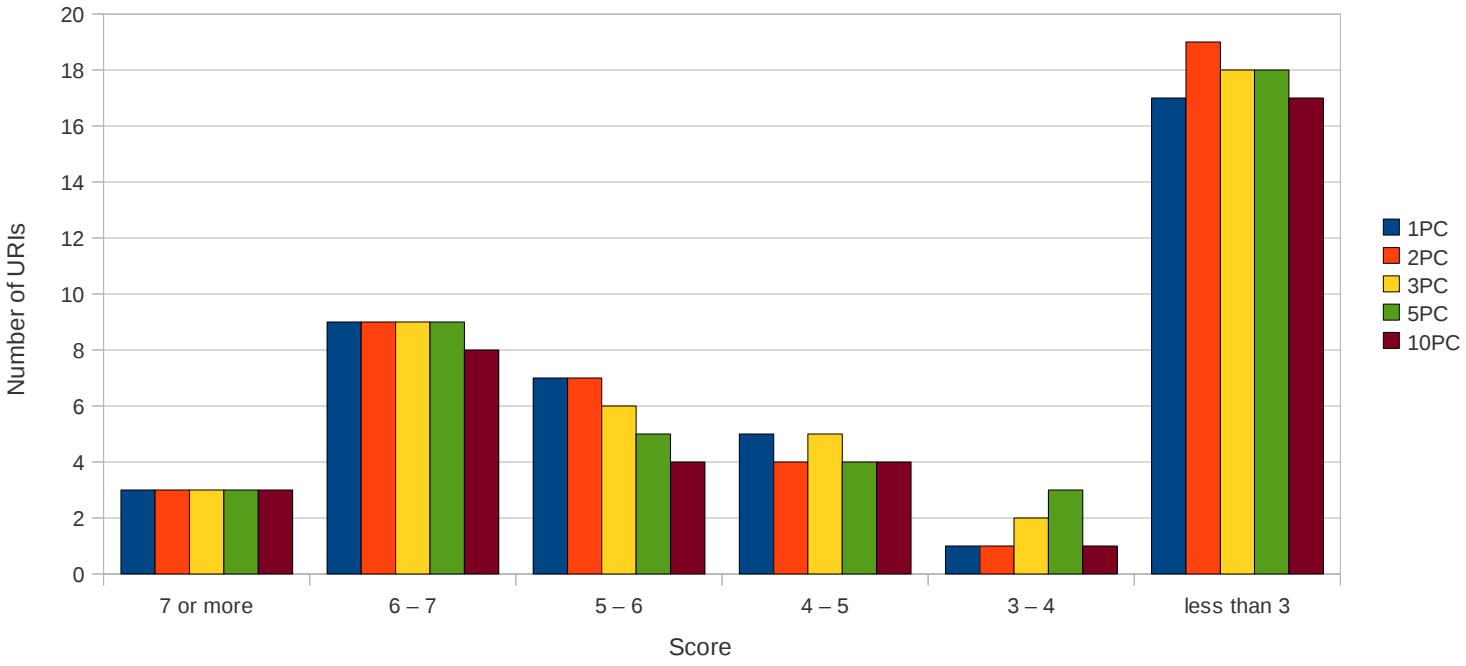
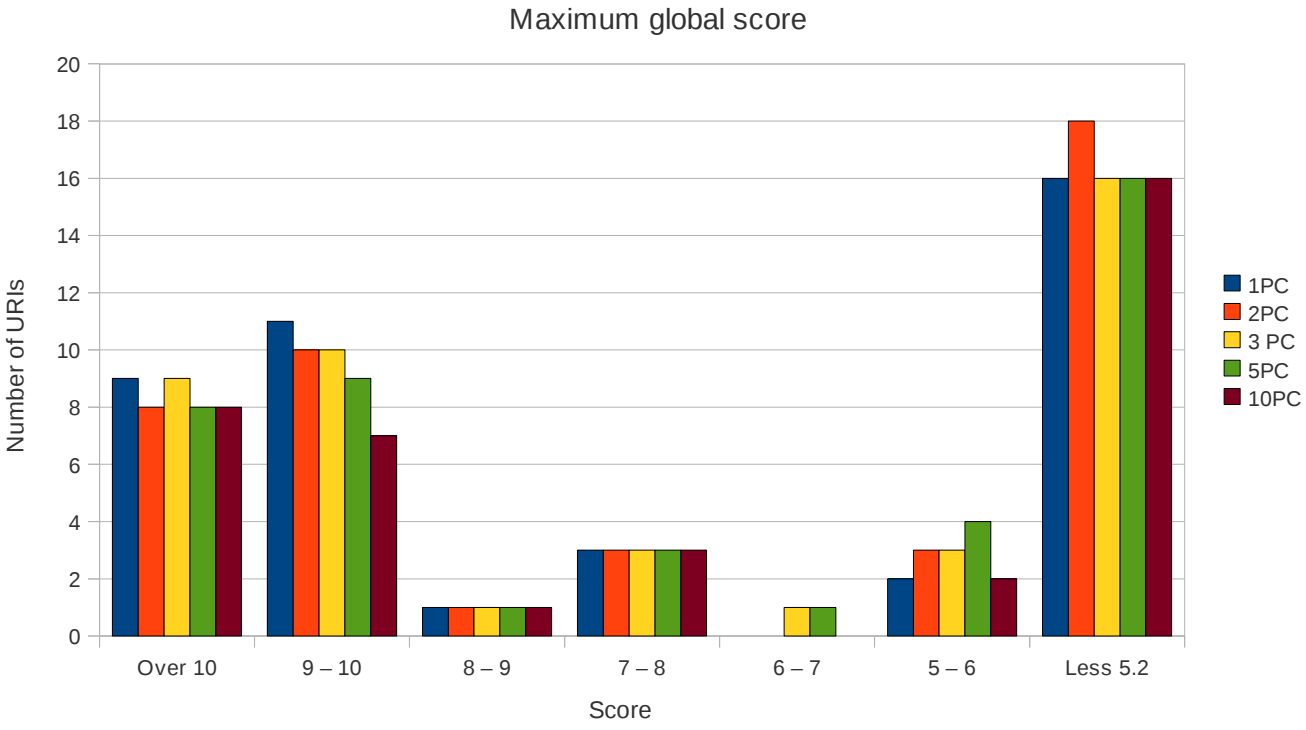


Figure 26



To analyze the performances of our algorithm compared to VoIP Seal we can compare the number of users identified with our multiple round analysis for each range of scores (using the two types of it), among all the users in the same range present in the trace. We did this comparison only with the most significant ranges of scores (the higher ones), obtaining the results showed in Table 3.
Observing this table we can notice how the quantity of users detected with our system is only a small part of the total number of users with a "high" score present in the trace.
As we said there seems not to be a clear correlation between the scores and the amount of anomalous users detected with our algorithm, as the two AD systems have a different way of defining anomalous a certain user (VoIP SEAL is focused on detecting telemarketer, so has good performances in detecting volume anomalies but is not able to detect behavioral anomalies).

Table 3

Maximum score

Score	Detected	Total	Percentage
Over 10	9	29	31.03%
9 – 10	11	62	17.74%
8 – 9	1	81	1.23%
7 – 8	3	71	4.23%
6 – 7	2	63	3.17%
5 – 6	5	627	0.80%

Average score

Score	Detected	Total	Percentage
Over 7	3	9	33.33%
6 – 7	9	22	40.91%
5 – 6	8	49	16.33%
4 – 5	5	76	6.58%
3 – 4	3	79	3.80%

5.6 Considerations on the results

The multiple round analysis demonstrates how the limit imposed by the possibility of detecting maximum one user per time-bin was a big issue when trying to apply this algorithm as an Anomaly Detection technique for VoIP traffic: with this second type of analysis, in fact, we reached much better results, detecting a bigger number of anomalous users overall and avoiding the problem of selecting an appropriate number of Principal Components (we observed that a number of PCs equal to 2 is sufficient to detect almost all the anomalous users that we can detect with this type of software). However we should also consider the computational requirements of this type of analysis: to compute a full multiple round analysis with 2 Principal Components, for example, we needed 36 rounds, each round corresponding almost to the full analysis performed in the first section (even if we don't scan the full range of thresholds in each round, we must compute every time the first step of the software, the matrix construction, that requires, more or less, half the time needed to compute the second step for 10 different numbers of PCs and 100 different thresholds).
The massive computational requirements can be reduced using some variations of the process, for example removing each round not only the first anomalous user identified (scanning the range of thresholds from the maximum to the minimum value) but all the users identified with a full range of thresholds analysis; that could lead to a fastest process of identification of the same users (in particular reducing the number of times the first step of the software must be computed, by reducing the number of rounds required to identify all the anomalous users) and maybe to the identification of more users (but probably also more false alarms), but we didn't exploit this chance, because we wanted to focus on the possibility of applying this software as an Anomaly Detection techniques with a normal, single round, analysis, selecting an appropriate number of PCs and threshold's value. To better analyze the software's performances with this kind of use we decided to perform another type of analysis, that we will discuss in the next section.
Last words are on the comparison with VoIP Seal that showed us how this kind of Anomaly Detection technique can be used in combination with other types of techniques, such VoIP Seal, that are more effective in identifying a high number of anomalous users; still our technique could be useful because it permits the identification of different kinds of anomalies, pointing out users that with other AD softwares are not considered anomalous.

6. Artificial anomalies

6.1 Type of analysis

At the end of the multiple rounds analysis, that we discussed in the previous section, we obtained a trace in which all the anomalies detectable with our software were been removed; we considered this trace a clean trace (launching the software using this trace as an input led to no anomalous users detected).
Using this trace as a base we decided to use different types of artificial anomalies to measure the performances of the software and his capability of detecting different types of anomalies. We decided to insert in the clean trace 10 different anomalous users, varying every time their type of behavior (anomaly type).
Since the most interesting aspect of this software is the possibility of detecting behavioral anomalies (top-users can be identified in a more efficient way with other types of AD techniques) we decided to model this artificial anomalies using the following shapes:

- Single spike anomalies (a single burst of calls in a single time-bin, different for each of the 10 anomalous users)
- Rect-type anomalies (a burst of calls that lasts for 5/10 consecutive time-bins, with a starting hour different for each of the 10 anomalous users)
- Triangle-type anomalies (a gradually increasing burst of calls, starting from zero, reaching a top peak value and going back to zero, with different starting hours for each of the 10 anomalous users and total durations of 5/10 time-bins)

We tested the software using these types of anomalies, choosing 10 random time-bins as a starting hour for each anomalous user (time that remain fixed even changing the type of anomalies); in the rest of the trace those users remained silent (no attempted calls).
On the traced obtained inserting a certain type of anomaly upon the “clean” trace we decided to perform the first type of analysis (first section), the single round analysis that used numbers of Principal Components from 1 to 10 and threshold's values from 0.000001 to 0.0001, to better understand the software's performances in detecting each type of anomaly as depending from the parameters values.

6.2 Single spike anomalies

The first type of anomalies we injected in the “clean” trace were single spike anomalies; in detail we added records corresponding to 10 different users (with new URIs that were not already present in the trace) that remained silent for the whole duration of the trace (no attempted calls in any hour), except for a single time-bin (correspondent to 1 hour), different for each user, in which they had a burst of calls.
The initial selection of the 10 time-bins in which the anomalous users had their peaks was made in a random way (and then they remained fixed, even with other types of anomalies).
We performed the full (all thresholds, all PCs numbers from 1 to 10) analysis several times, changing each time the amount of calls we had in the peaks (we will start calling this parameter “anomaly height”). We started with small peaks of 5 calls to reach peaks of 10000 calls, to test the software's performances as dependent from this parameter.
In Figure 27 we can see the shape of the anomalies, with an anomaly height of 100 calls: we plotted for each time-bin the amount of “artificial” calls injected in the trace (of course they must not be considered as a single anomaly, since each of those peaks corresponds to a different user).
In Figure 28 we can observe the positions of the artificial anomalies in the global trace: we plotted the total calls' ratio on the maximum value of them (among all users, including the artificial ones) in each time-bin, highlighting the time-bins in which we added the anomalies. We can observe that the global behavior is almost the same (compare Figure 1), since the amount of calls we added (100 in this specific case) is almost negligible in the global amount of calls.

Figure 27

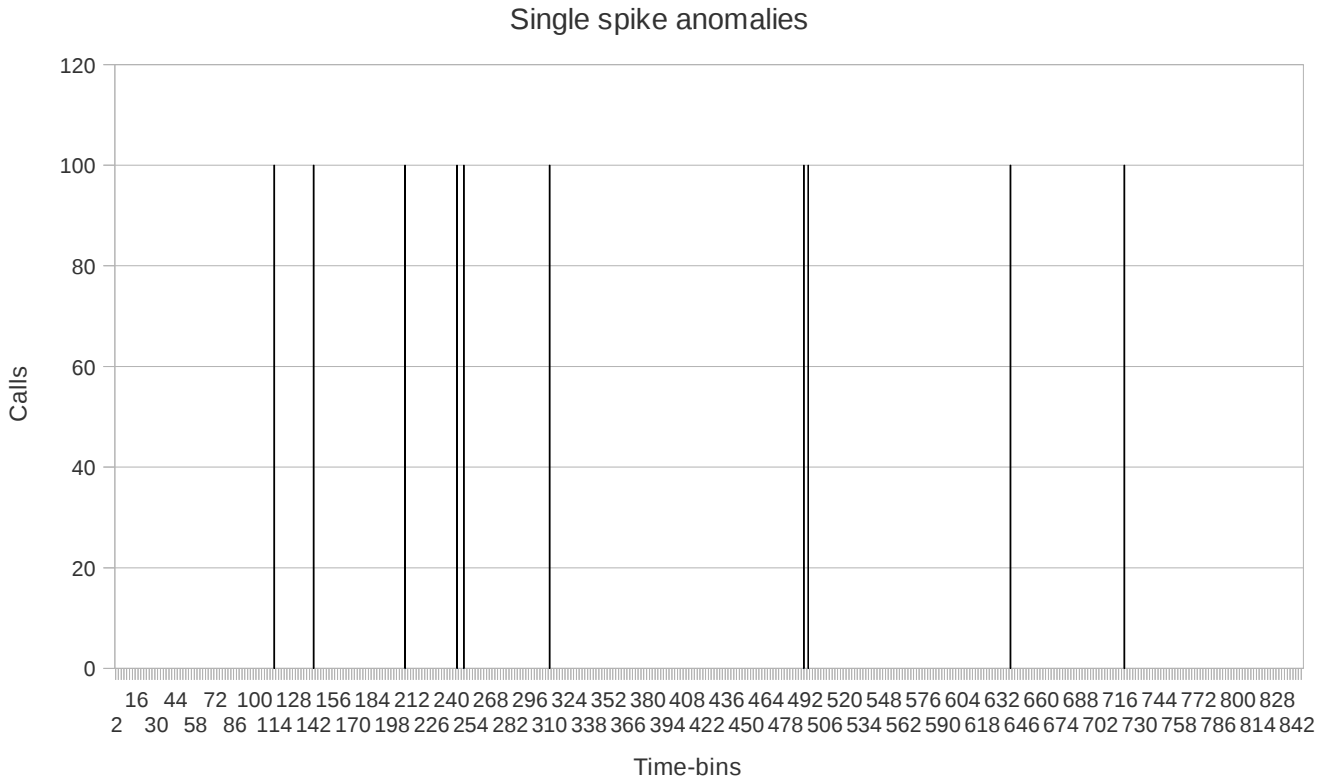
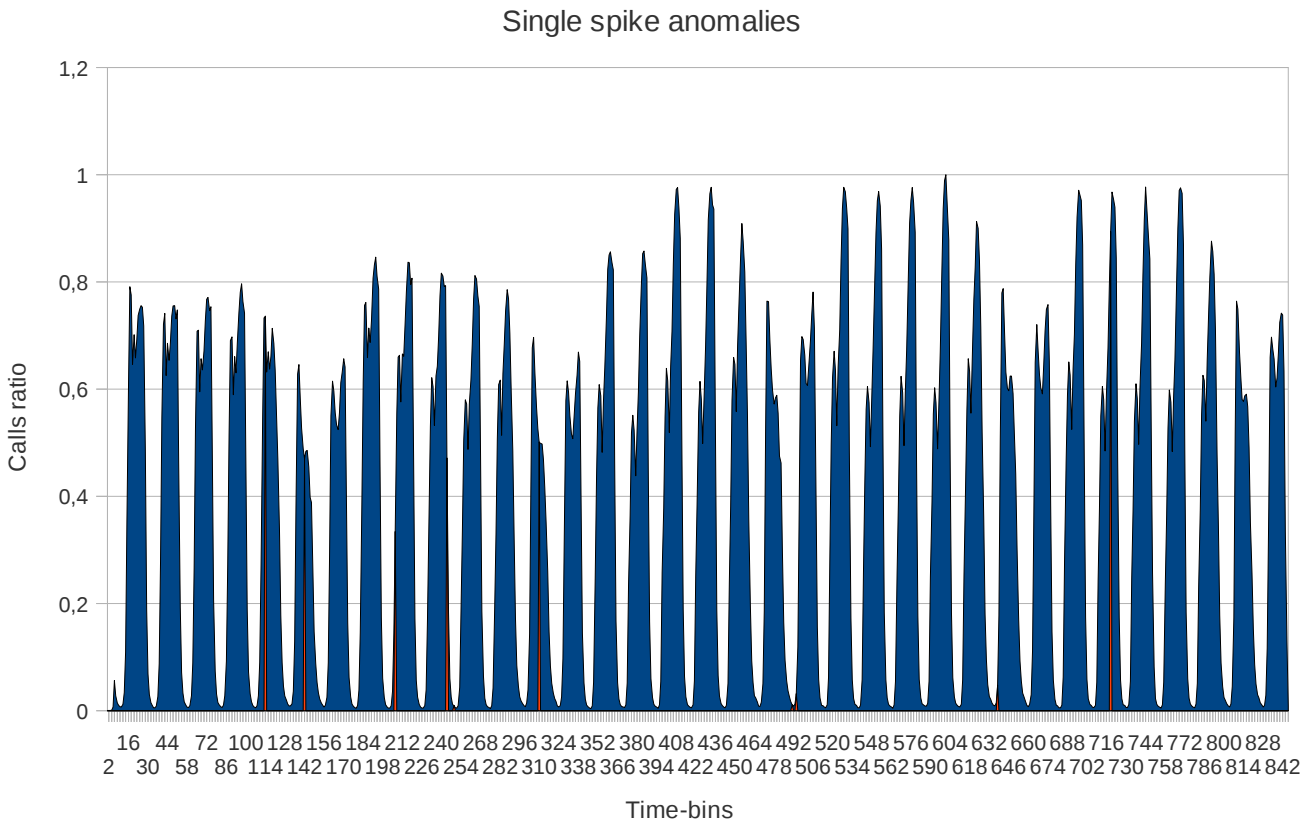


Figure 28



In particular we can observe how two pairs of anomalies are happening in very near time-bins (a pair in time-bins 243,248 and the other in time-bins 490,493); we'll see that this kind of situation leads to detection problems for the software. We should also notice how the anomalies happen in different types of time-bins: for example the anomaly in time-bin 718 is in a busy hour (with almost 100000 calls), while the anomaly in time-bin 637 is in a low traffic hour (with less than 10000) calls; this will also have a deep impact on the software's performances in detecting these anomalies.

To understand how the position of an anomaly in the trace can impact the possibility of detecting it we first focused on the order in which the inserted anomalies were detected: we started with a very low anomaly height (5 calls) and we increased it until a further increase wouldn't lead to a bigger number of anomalous users detected. With the first trials we had no detections, but with an anomaly height of 100 calls we started to detect the first anomaly: in detail we detected the anomaly happening in time-bin 637, that corresponded to a very low-traffic hour in the trace.

Increasing the anomaly height other anomalies started to be pointed out, and we noticed that in general a smaller number of global calls in the time-bin in which the anomaly was inserted led to an earlier detection by the software; in detail the order of detections was the following:

1. Time-bin 637 (4568 global calls, excluding the artificial ones)
2. Time-bin 493 (3035 calls)
3. Time-bin 206 (32873 calls)
4. Time-bin 243 (46438 calls)
5. Time-bin 141 (46390 calls)

6. Time-bin 309 (49378 calls)
7. Time-bin 113 (72566 calls)
8. Time-bin 718 (88166 calls)

With an anomaly height of 900 calls we could identify these 8 anomalous users. Using bigger numbers of calls led to a better identification of these same users (detected with more thresholds and more number of Principal Components), but not to the detection of the other two users (even with peaks of 10000 calls we were not able to detect them). The two missing users were positioned in time-bins 248 (888 calls) and 490 (1014 calls), really near to the anomalies in time-bins 243 and 493 (the fourth and the second detected); even if they were happening in time-bins with a very low global number of calls we were not able to identify them, probably because of the interference of the other two near anomalies. Further analysis would be required to understand the reasons of this phenomenon, but we left this as an opportunity for future work.
In order to show how the amount of global calls influences the software's capability of detecting a certain single spike anomaly we plotted the number of times each anomalous user was detected among the full analysis (with 100 thresholds and 10 different numbers of PCs, so with a maximum number of detections equal to 1000) varying the anomaly height. We can observe this graphs in Figure 29 and 30 where we plotted the number of detections for each time-bin together with the global behavior of the trace (as in Figure 28) for anomaly height of 100 calls (first user detected) and 1000 calls (eight users detected); from the second one we can observe how time-bins with a lower global number of calls are detected more times than others.

Figure 29

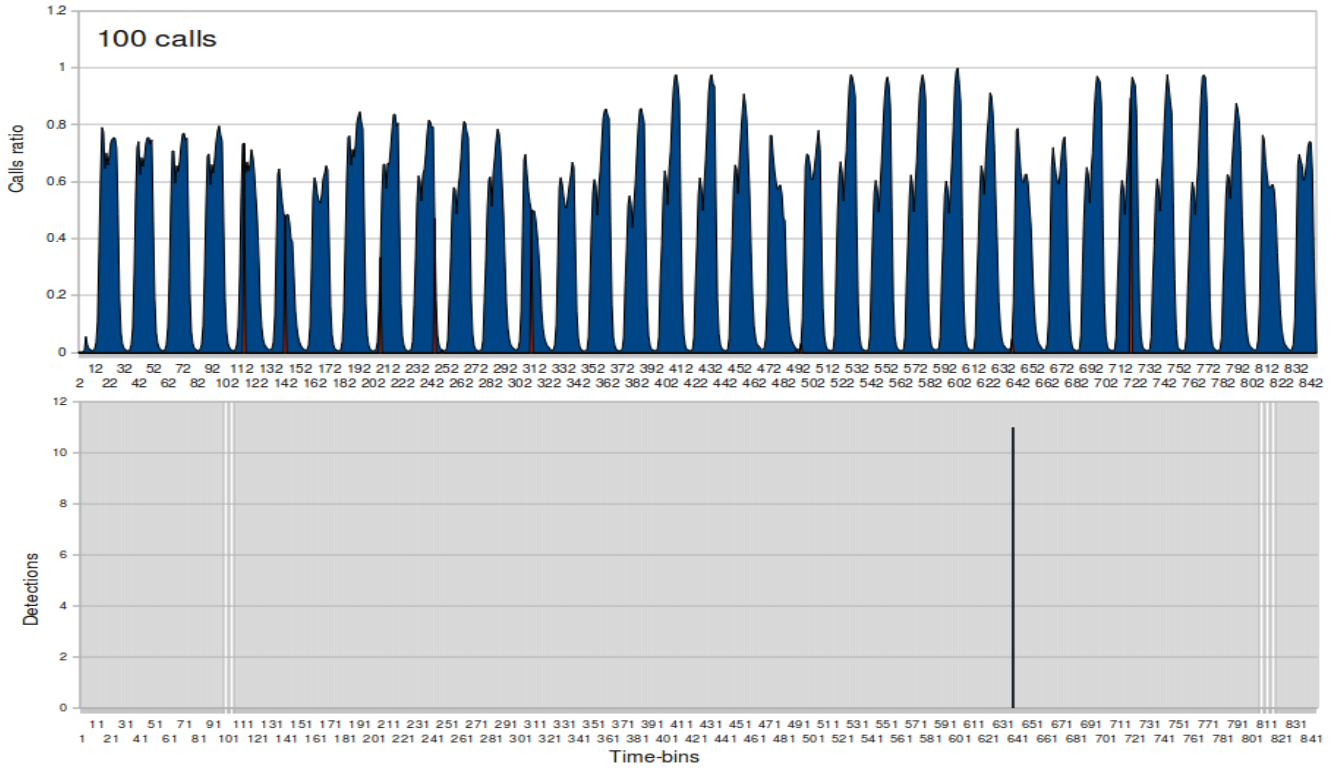
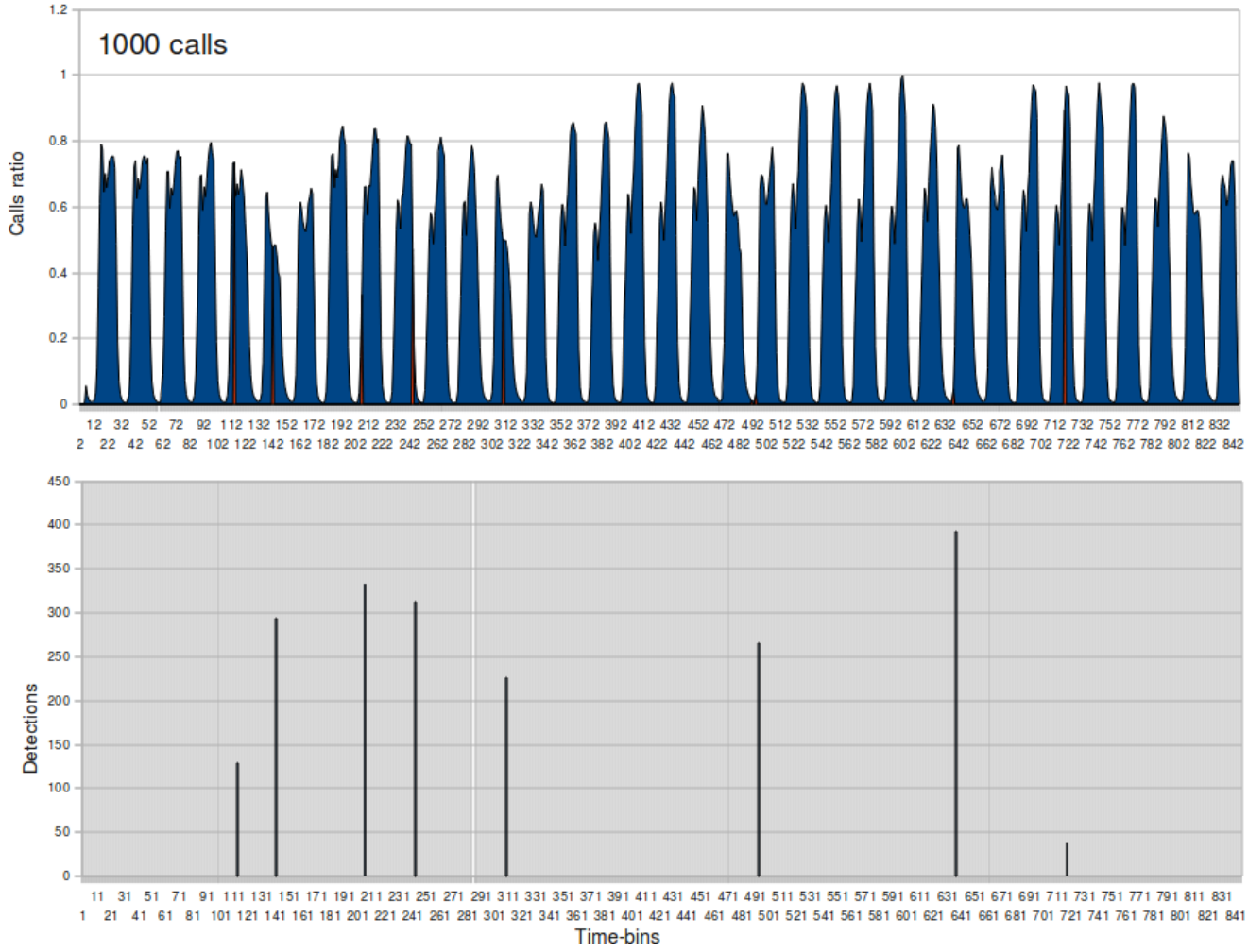
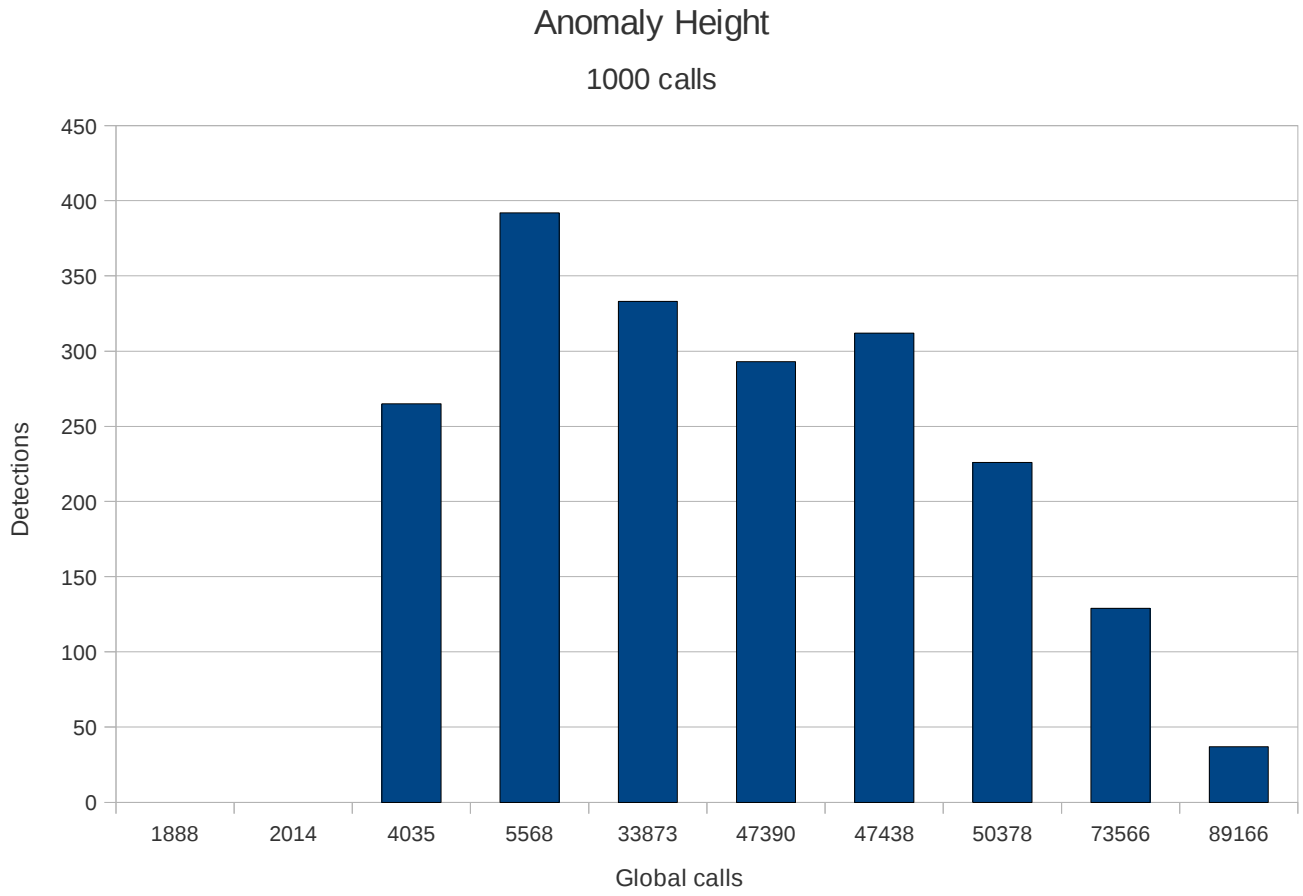


Figure 30



We can point out this tendency to better detect time-bins with a lower number of global calls by plotting the number of times each anomalous time bin was detected (we didn't included not-anomalous time-bins, cause they were never been detected) versus the amount of global calls in that time-bin, ordering the results by the number of global calls. We did this in Figure 31, for an anomaly height of 1000 calls: we can observe a small tendency to have more detections in time-bins with a lower number of global calls (with the exceptions of the two time-bins that were never been detected, even if they were the two with the lowest number of calls).

Figure 31



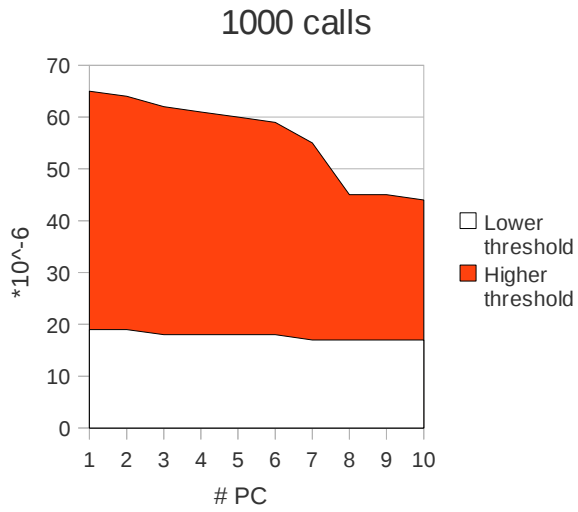
One of our targets, when starting this kind of analysis, was observing the software's performances as dependent from the parameters values (thresholds and numbers of PCs), so we decided to focus on the range of (consecutive) thresholds that permitted the identification of a certain anomalous user. From the results we could see how this range is dependent from the number of Principal Components used in the analysis as well as from the anomaly height and the position of the anomaly in the trace. We already analyzed how the anomaly's position impacts the capability of detecting it (anomalies positioned in time-bins with a low number of calls are detected more times, in a full thresholds analysis, that means they are detected with a wider range of thresholds), so we focused on the dependence from the number of PCs.

We took into consideration the first anomaly detected (the one in time-bin 637) and we plotted on several graphs (one for each anomaly height) the range of thresholds in which that anomaly was detected versus the number of PCs used in the analysis. We can see one of this graphs in Figure 32 (the one with anomaly height of 1000 calls), in which the top line is the maximum threshold for which that anomaly is detected and the bottom line is the minimum threshold (the red zone is the range of thresholds in which that anomaly is detected with the appropriate number of PCs).

From the graph we can see how a smaller number of PCs permit a better detection of the anomaly, since

the range of thresholds for which that anomaly is detected is wider. That happens in general for each anomaly added and for each anomaly height, with different width of the range (depending from the position of the anomaly, as we discussed before, and from the anomaly height, as we will discuss in the following).

Figure 32



We can discuss this type of performances in a more detailed way by focusing on two ratios:

- Anomalous time-bins detected by a single analysis (one number of PCs and one threshold) upon the total number of anomalous time-bins in the trace
- Anomalous users recognized by a single analysis upon the total number of anomalous users inserted

With single spike anomalies these ratios will assume the same value, as each anomalous users correspond to only one anomalous time-bin, but for other types of artificial anomalies it will make sense distinguishing between these two types of performance's metrics. In order to understand which are the “best” thresholds for detecting the maximum number of anomalies and how much precision we need in the threshold's selection (how wide are the ranges of thresholds that permit the best performances), as well as to observe how these values are dependent from the number of PCs used in the analysis we decided to plot the two ratios described before (even if in this case they will assume the same value) versus the threshold's value, for each number of PCs. In Figure 33 and 34 we can observe this kind of graph with an anomaly height of 1000 calls and using 1 and 10 Principal Components.

Figure 33

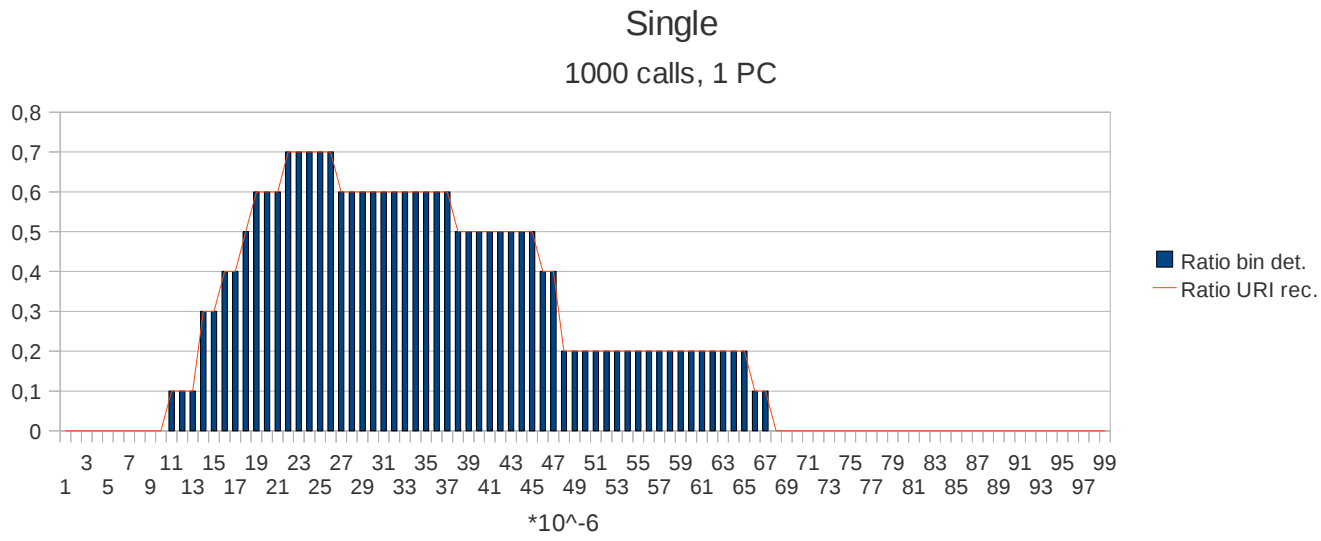
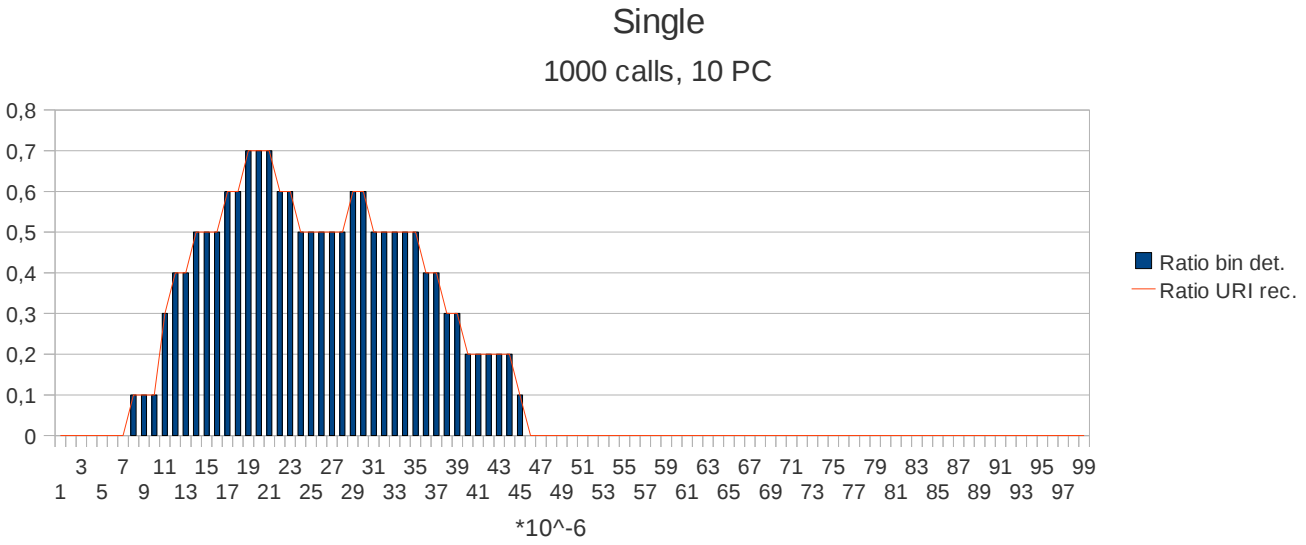


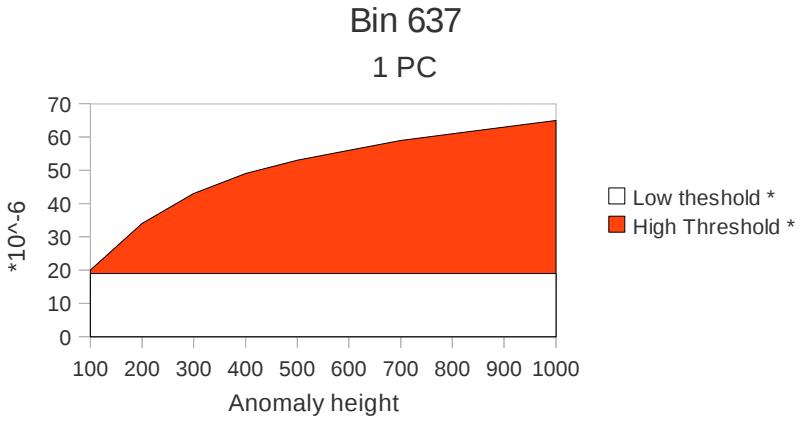
Figure 34



The first observation we can do on these graphs is that there is no threshold's value that permit the detection of the 8 detectable users, even if in Figure 30 we saw that with an anomaly height of 1000 calls we could detect 8 of the 10 anomalous users inserted. The reason of this incongruence is that Figure 30 was referred to a “full” analysis (all thresholds and all PCs numbers), while Figure 33 and 34 are referred to a single analysis (performed selecting one threshold's value and one number of Principal Components); the fact that with this second type of analysis we can only identify up to 7 users means that different users are identified using different ranges of thresholds, and it doesn't always happen that these ranges overlap. We already observed in Figure 32 how an anomaly is not identified with all the possible threshold's values from a high-limit value to zero: there is also a low-limit value, under which that specific anomaly is not detected. As we said in the first chapter this phenomenon is related to the

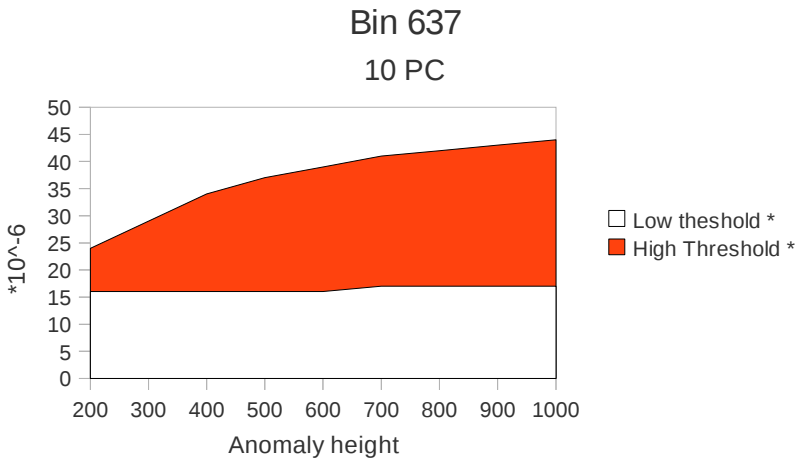
identification procedure the software uses; it can lead to some performance's degradation when the low threshold's limits of some anomalies are higher then the high threshold's limits of others, that means it doesn't exist a single threshold (and neither a range of thresholds) that permits the identification of these different sets of anomalies.
Other considerations we can make on this graphs are that the “best” threshold (the one that permit the identification of the biggest number of users or time-bins) decreases with the number of Principal Components used, as well as the width of the threshold's range that permit to obtain that “best” performance. We already observed this kind of phenomenon with the previous types of analysis (single and multiple round with the original trace) and is related to the fact that a smaller number of Principal Components is more suitable to leave anomalous users in the *anomalous* subspace (with a normal trace that leads also to the inclusion of normal traffic in that subspace, but it doesn't apply to this case, since we are using a previously “cleaned” trace).
Another dependence we are interested to analyze, when observing the software's performances, is with the anomaly height. We could think that a bigger burst of calls would be automatically better detected then a smaller one, but this is true only to a certain point.
In Figure 35 we plotted the threshold's range that permits the identification of the anomaly in time-bin 637 (the first one detected), using one Principal Component, against the anomaly height; the top line is the maximum threshold that permits the detection, the bottom one is the minimum, so the red zone represents the range. We can observe how the lower threshold remains approximately stable, while the higher one tends to grow up with the anomaly height (a peak of 1000 calls is easily detectable than a peak of 100 calls), so the range of thresholds increase as well.

Figure 35



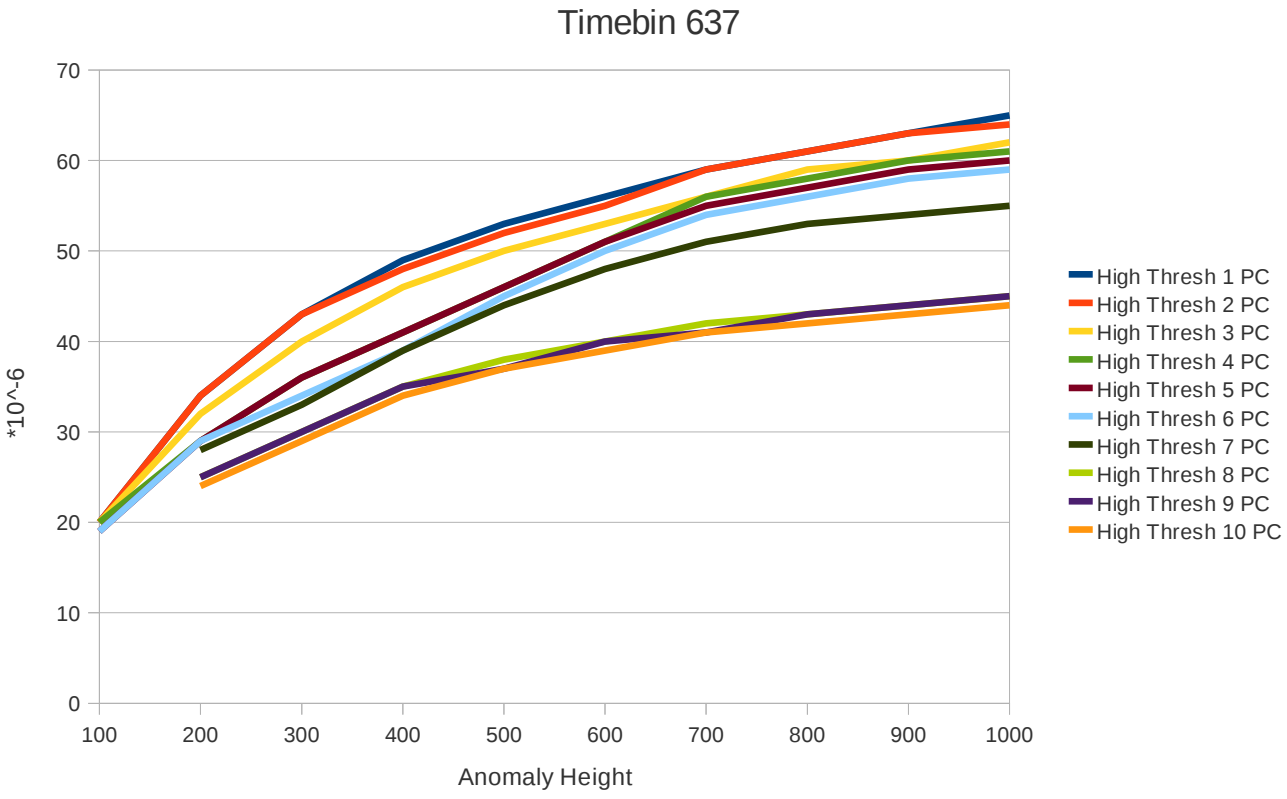
As we said before, using higher numbers of Principal Components leads to smaller threshold's values (and smaller ranges of thresholds), so comparing this graph with the same one computed with 10 PCs would lead to lower values of the thresholds. We can see that from Figure 36, where we plotted the same graph as Figure 35, but using 10 PCs: we can observe how the high threshold's values are smaller (and for example there are no thresholds that permit detections with 100 calls), as well as the threshold's ranges.

Figure 36



But a comparison of just the high thresholds values, as the one showed in Figure 37 (in which we plotted on the same graphs the high thresholds values against the anomaly's height, for each number of PCs) leads us to some other considerations.

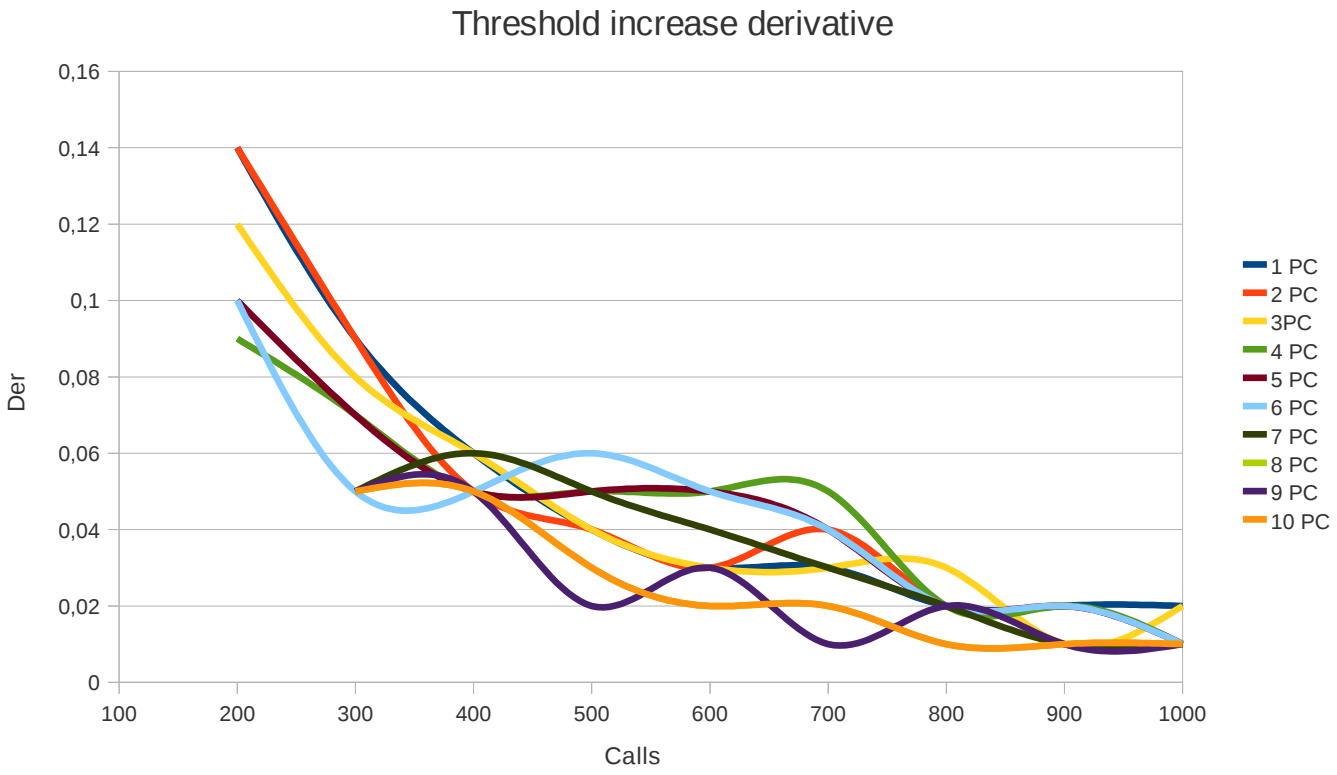
Figure 37



We can observe that thresholds from higher numbers of PCs not only are smaller than thresholds from

lower numbers of PCs, but they also grow slower with the increasing of anomaly's height. We can better see this aspect by plotting the derivative of the threshold's increase versus the anomaly's height (derivative at step *n* is computed as [threshold(*n*)-threshold(*n*-1)]/[calls(*n*)-calls(*n-1*)]), for each number of PCs used. We did this in Figure 38, where we can see how the derivatives correspondent to higher numbers of PCs are always lower than derivatives correspondent to lower numbers of PCs.

Figure 38

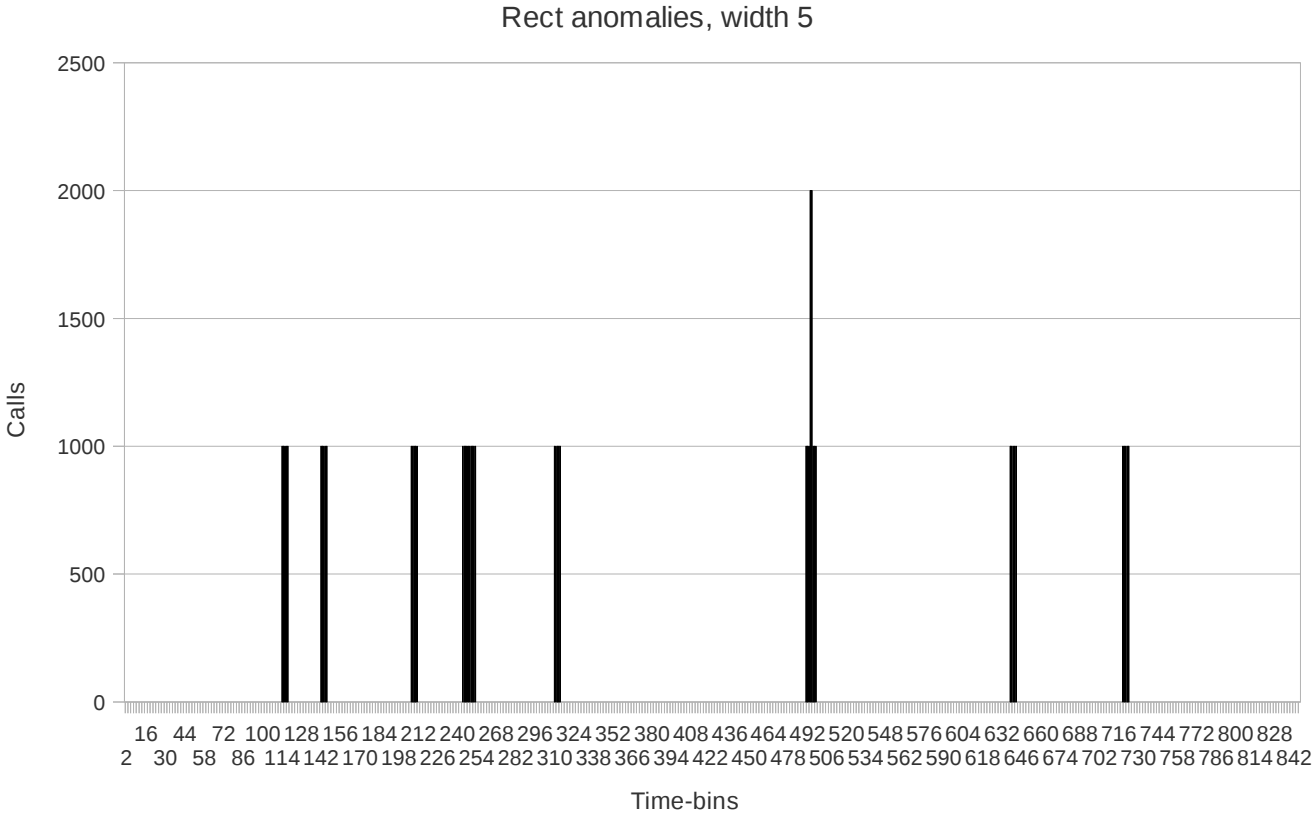


From these graphs we can observe how the derivative tends to decrease with the growing up of anomaly's height (thresholds grow slower); in the following chapters we'll see that they can also become negative with the calls growing, leading to a decrease of the threshold's set that permit the anomaly's identification.
The reason behind this phenomenon is related to the fact that really big anomalies can start to influence the trace at the point that they start to be included in the *normal* subspace. Of course this can easily happen with higher numbers of PCs (they capture a higher amount of the traffic variance, and are more susceptible of including anomalous traffic in the normal behavior), but increasing the anomaly's height can definitely cause the same problem also to an analysis performed with a low number of PCs.

6.3 Rectangular anomalies, width 5

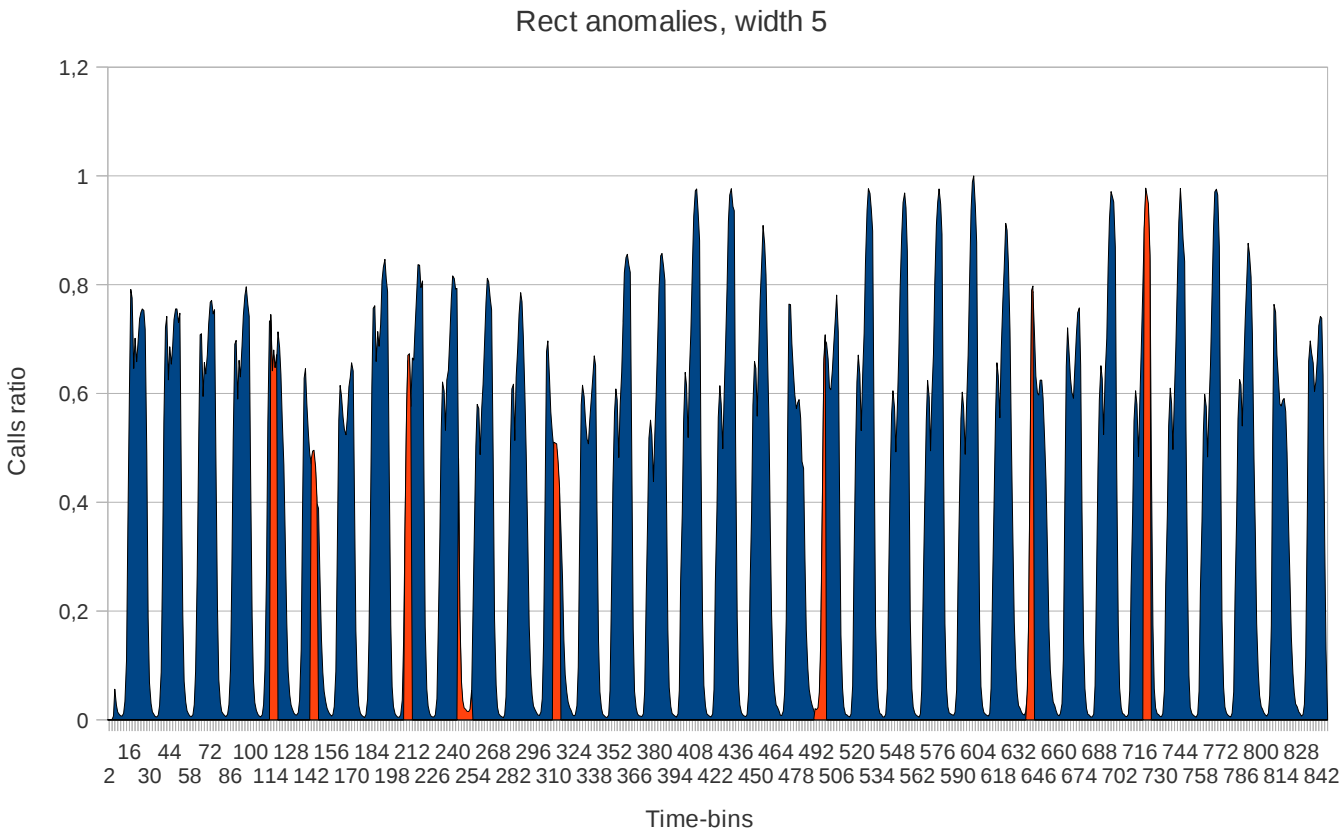
The second type of anomalies we injected in the trace are rectangular shaped anomalies, that we created by simply extending each of the 10 single spike anomalies for 5 time-bins. We obtained 10 users that remain silent for the whole duration of the trace, except for 5 consecutive time-bins in which they have a burst of calls (the same amount for all the 5 time-bins). The starting time-bins for these anomalies are the same of the single spike anomalies, so we obtained one case of overlapping: one anomalous user is active in time bins 490-495 and another one in time-bins 493-497, so in time-bins 493-495 we have two anomalies happening at the same time. That is a major problem for our software, since it's only able to detect one anomalous user per time-bin, so we are sure that in those time-bins we will only be able to detect one of the two anomalous users.
We performed the analysis starting with an anomaly's height of 5 calls and reaching an height of 10000 calls; in Figure 39 we can see the anomalous calls inserted, with an anomaly's height of 1000 calls; in particular we can note the overlapping zone in time-bins 493-495.

Figure 39



In Figure 40 we can observe the positions of the artificial anomalies in the global trace: we plotted the ratio of total calls (among all users, included the artificial ones) in each time-bin, highlighting the time-bins in which we added the anomalies (we used an anomaly height of 1000 calls).

Figure 40



As with the first type of anomalies we began focusing on the order of detections: we considered the order in which the time-bins were identified starting with a low anomaly's height (5 calls) and increasing it until all the anomalous users were identified.
The first time-bin identified was time-bin 252, a time-bin with a quite low amount of global calls (4320 excluding the anomalous) positioned at the end of 10 consecutive time-bins with anomalous calls (one anomalous user was active in time-bins 243-247 and another one in time-bins 248-252).
The order in which the anomalous users were detected was the following (for each anomaly we indicated the number of global calls, except the anomalous ones, in the first time-bin detected for that anomaly):

1. Anomaly in time-bins 248-252 (4320 calls in the first time-bin detected: time-bin 252)
2. Anomaly in time-bins 243-247 (2177 calls: time-bin 247)
3. Anomaly in time-bins 637-641 (5568 calls: time-bin 637)
4. Anomaly in time-bins 493-497 (5035 calls: time-bin 493), with the first time-bin detected that is part of the overlapping zone
5. Anomaly in time-bins 206-210 (33873 calls: time-bin 206)
6. Anomaly in time-bins 141-145 (40299 calls: time-bin 145)
7. Anomaly in time-bins 309-313 (41993 calls: time-bin 313)

8. Anomaly in time-bins 113-117 (66252 calls: time-bin 117)
9. Anomaly in time-bins 718-722 (84334 calls: time-bin 722)
10. Anomaly in time-bins 490-494 (13715 calls: time bin 494), with the first time-bin detected that is part of the overlapping zone

The last anomalous user identified deserves a better analysis: while the others were all detected with anomaly's heights lower than 1000 calls, it could only be identified with 10000 calls, in a time-bin in which it overlaps with the anomaly in time-bins 493-497 (detected first in another time-bin of the overlapping zone with an anomaly's height of 200 calls).

Except for that last user we can observe a similar tendency as the one experienced with single-peak anomalies, with anomalies positioned in time-bins with a lower amount of global calls that tend to be recognized before anomalies positioned in time-bins with a higher amount of global calls. We can also see that for each anomalous user the first time-bin identified was the first or the last, that does make sense since the software tries to detect changes in the behavior of each user.

We could conclude that the order in which the anomalies are identified is decided by the amount of global calls each anomalous user has in his border time-bins (the first or the last in which that user is active).

It would be interesting to plot the number of times each time-bin was detected in the global analysis (all thresholds and all numbers of PCs) for every anomaly's height, but for space reasons we will only show the first detection (with 70 anomalous calls in each anomalous time-bin),in Figure 41, and the situation with 10000 calls per anomalous time-bin, in Figure 42.

Figure 41

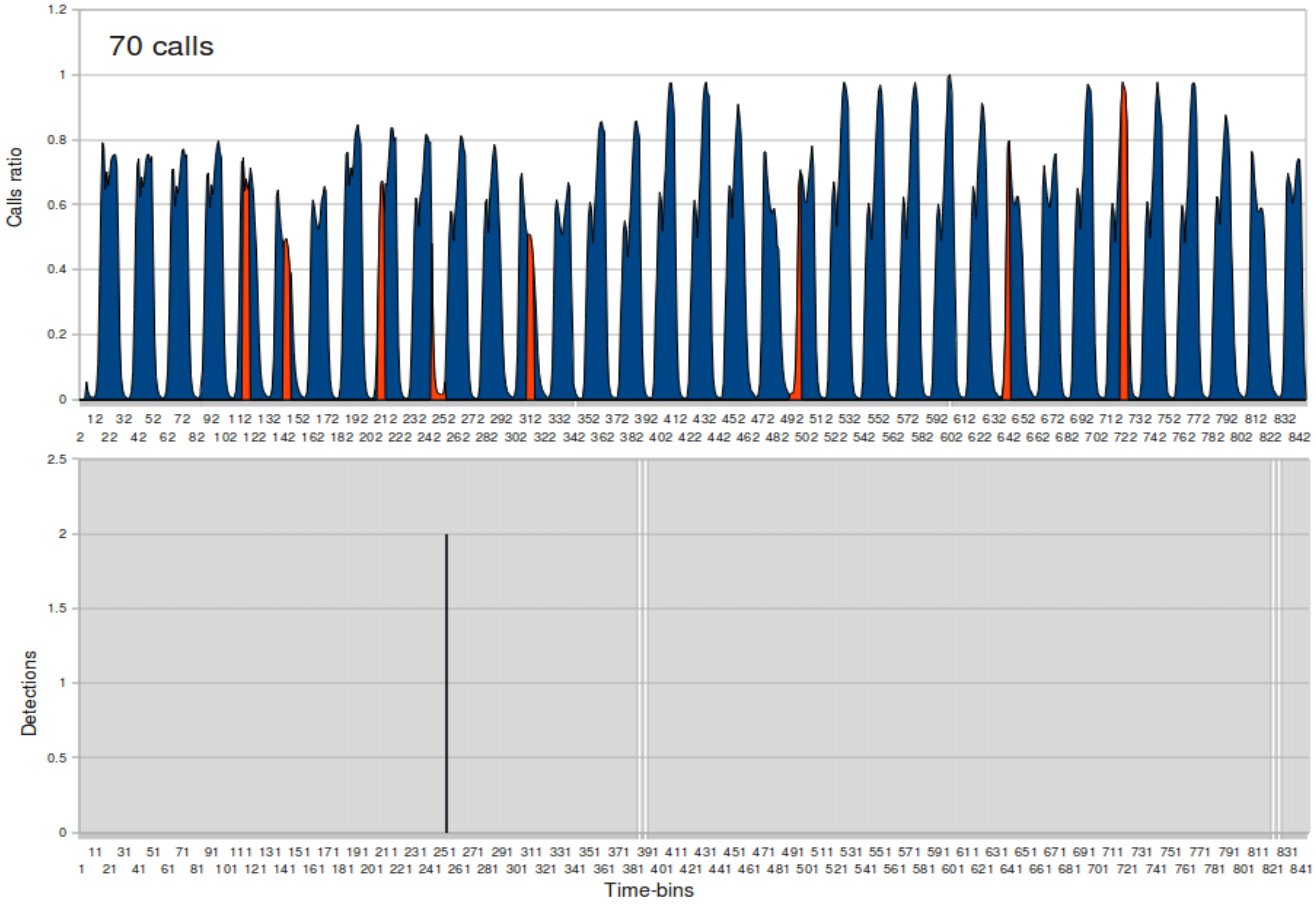
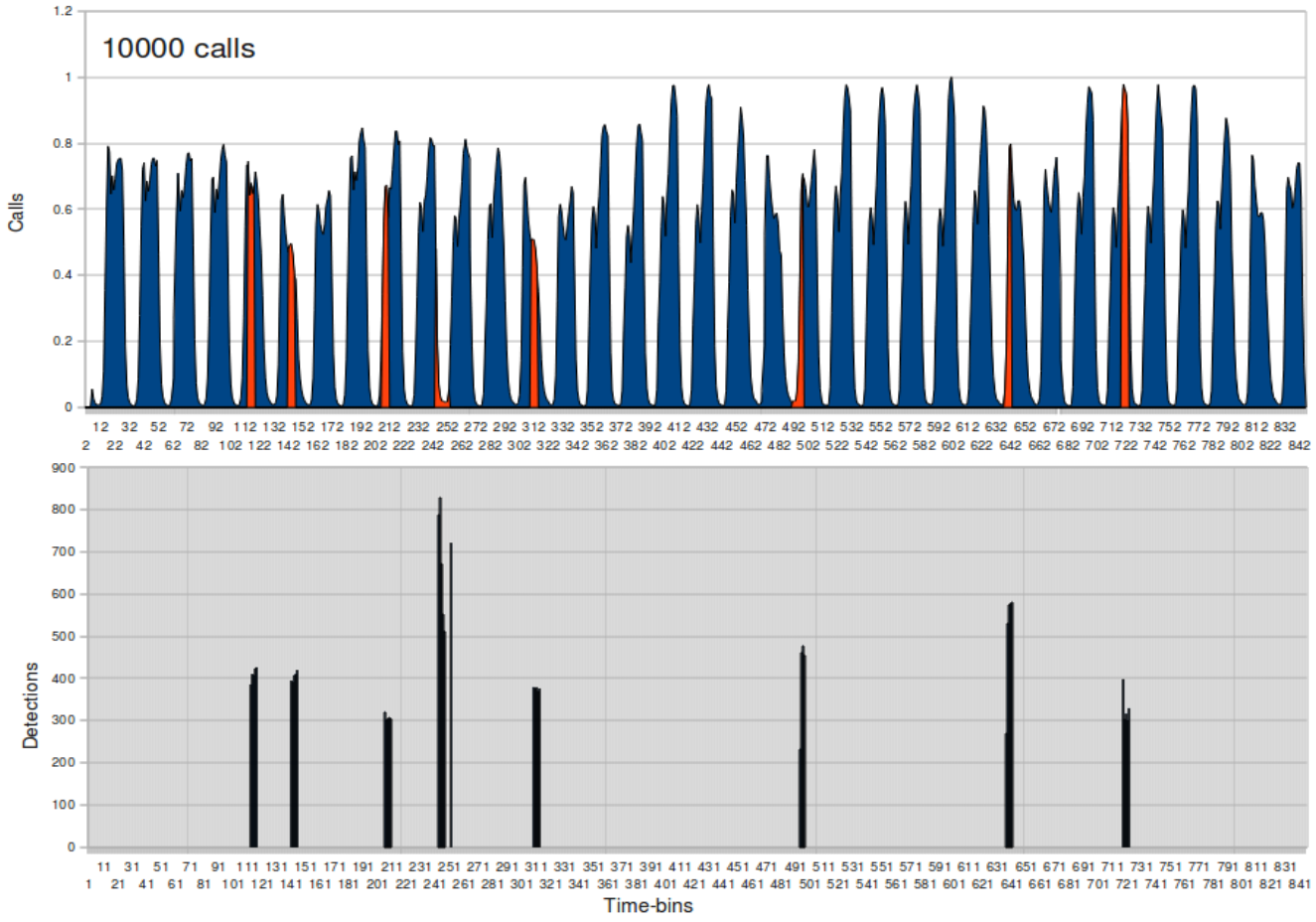
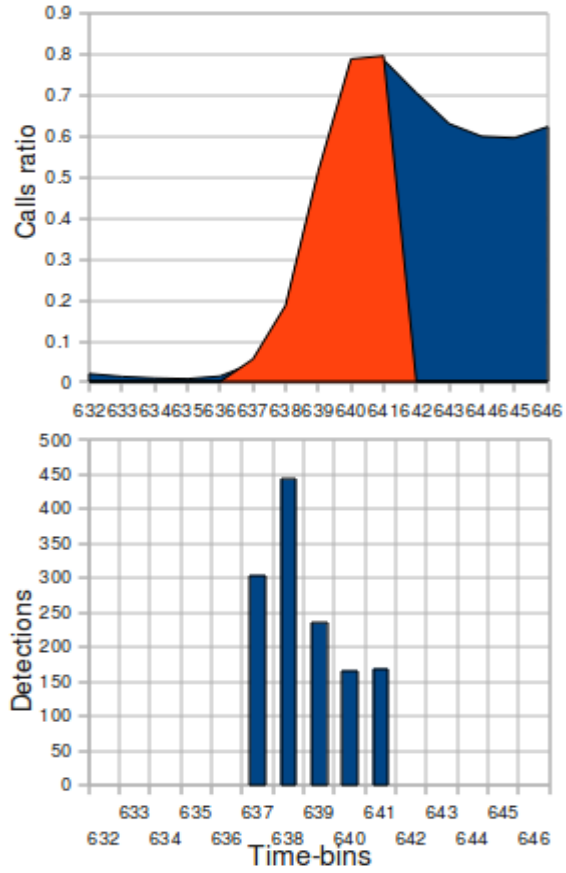


Figure 42



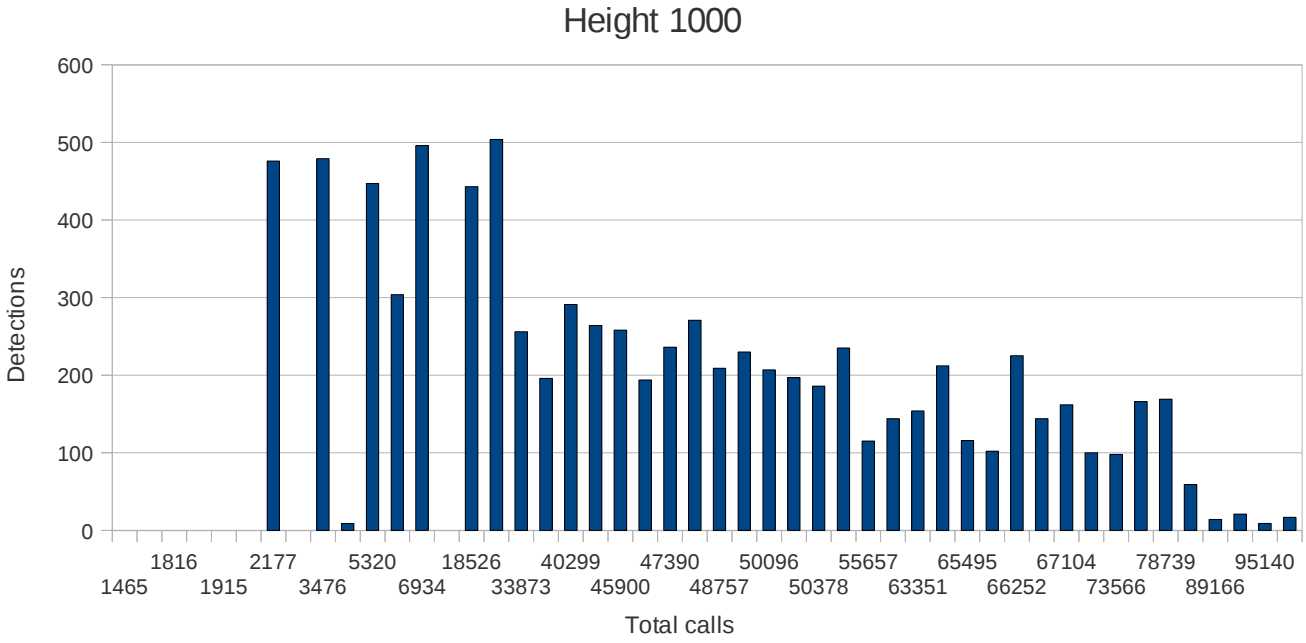
The fact that usually the first time-bin detected for an anomalous user is a border one doesn't mean that increasing the anomaly's height that time-bin will keep being the most detected: we can see this observing Figure 43, where we focused on the anomaly in time-bins 637-641, plotting the amount of detections we had for each time-bin, with an anomaly height of 1000 calls, compared with the global amount of calls' ratio (including the anomalous) in those time-bins.

Figure 43



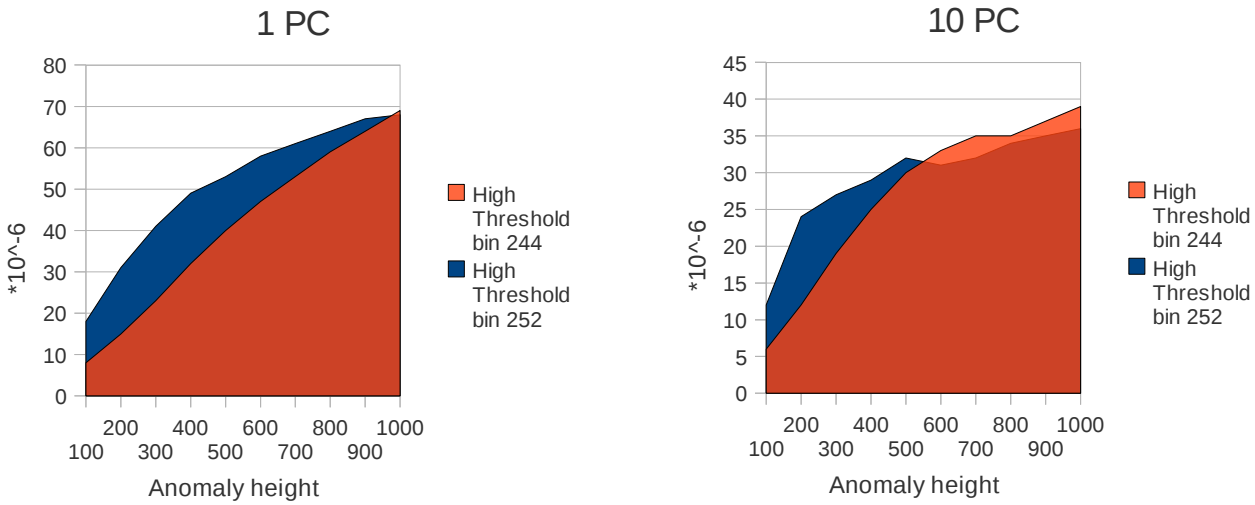
We can observe the tendency to better detect time-bins with a lower number of global calls; tendency that we can confirm plotting the number of times each anomalous time-bin was detected (we didn't include not-anomalous time-bins, cause they were never been detected) versus the amount of global calls in that time-bin, ordering the results by the number of global calls. We did in Figure 44, with an anomaly height of 1000 calls; with the exceptions of time-bins related with overlapping anomalies (the ones with amounts of calls from 1400 to 2000) the number of times each time-bin is detected increases with the decreasing of the amount of global calls in that time-bin.

Figure 44



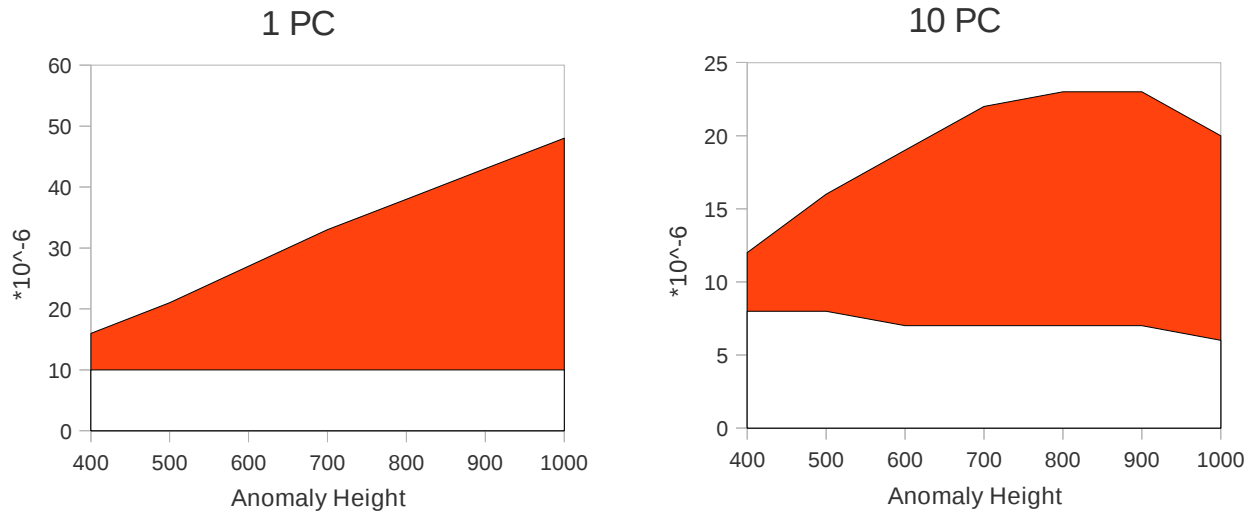
As with the first type of anomalies the second topic on which we focused is the sensitivity of the threshold to the anomaly's height and the number of Principal Components.
We first observed the thresholds range that permits the detection of a certain anomalous time-bin, as depending from the anomaly size, varying the number of principal components used; we did this observation for three different time-bins: time bin 244 (the one with most detections with 1000 calls), time bin 252 (the first one detected, with 70 calls) and time bin 145.
In Figure 45 we plotted the maximum thresholds that permitted the identification of the anomalous users in time-bins 244 and 252 against the anomaly's height, for 1 and 10 Principal Components used.

Figure 45



We can observe how the anomaly in time-bin 252 is better detected with a low anomaly's height, while the one in time-bin 244 is better detected with higher anomaly's heights. We can also observe how the thresholds grow slower with the increasing of anomaly's height, for higher numbers of Principal Components.
We can better see this behavior in Figure 46, where we plotted the range of thresholds that permit the identification of the anomaly in time-bin 145 versus the anomaly's height, with 1 and 10 PCs used.

Figure 46



Comparing the two graphs we can see how the number of Principal Components used can influence the software's performances when increasing the anomaly's height. With single peak anomalies we already saw how a higher number of PCs can make a threshold increase slower, but here we can see that it can even make a threshold decrease. That is not an uncommon phenomenon and we can see this by extending the analysis to the results obtained with 10000 calls per anomalous time-bin. If we focus on the higher thresholds obtained with 1000 and 10000 calls, for each number of PCs, for the three time-bins that we are analyzing we can observe that:

- For time-bin 244 the thresholds increase (passing from 1000 to 10000 calls) using 1 to 8 PCs and decrease using 9 or 10 PCs
- For time-bin 252 the thresholds increase for any number of PCs (from 1 to 10)
- For time-bin 145 the thresholds increase using 1 to 4 PCs and decrease using 5 to 10 PCs

We can observe this different behaviors form Figure 47-49, where we plotted the high thresholds versus the anomaly's height for each number of PCs (for the three time-bins considered), adding the final passage from 1000 to 10000 calls (breaking the x-axis scale).

Figure 47

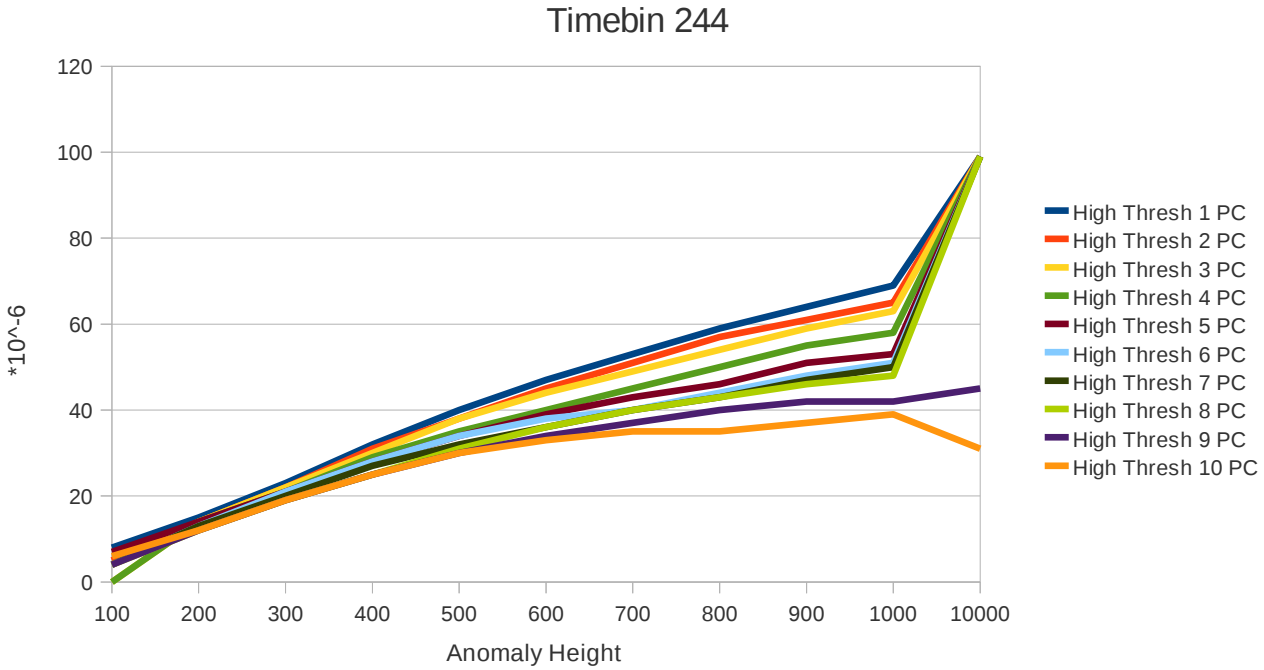


Figure 48

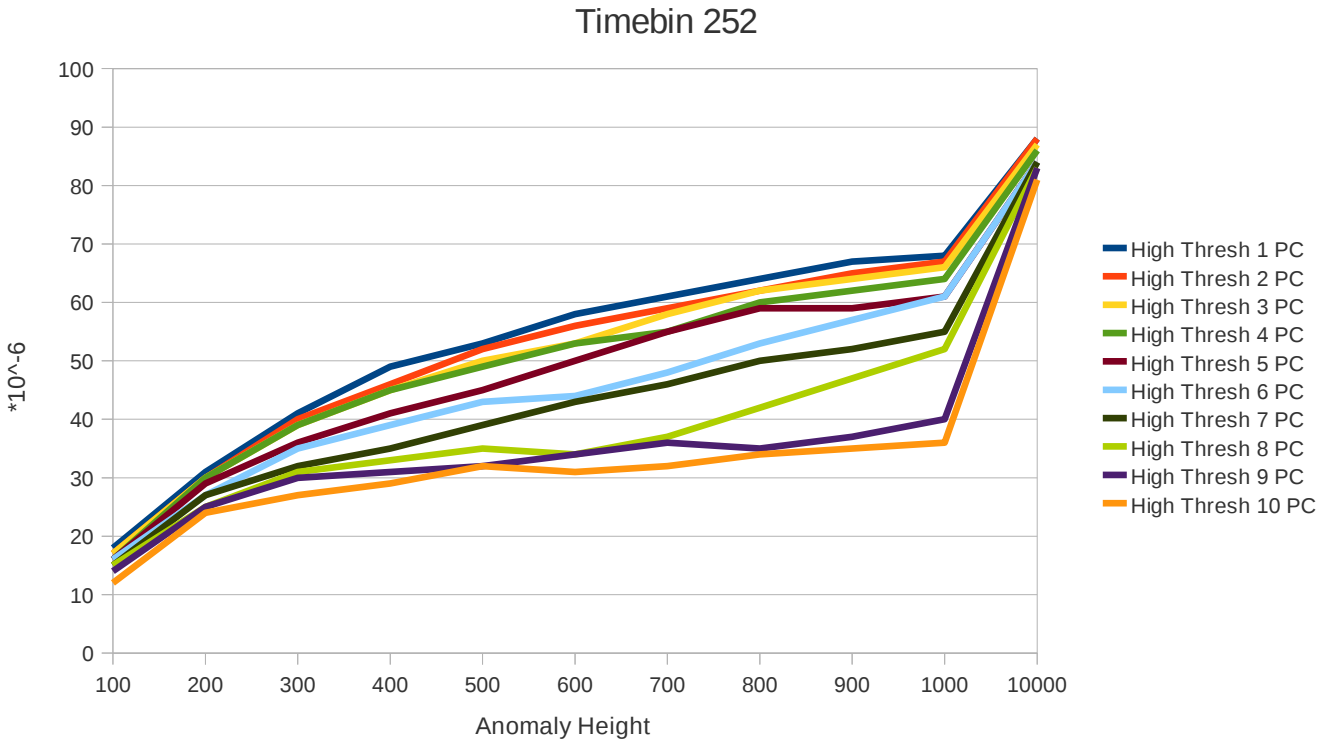
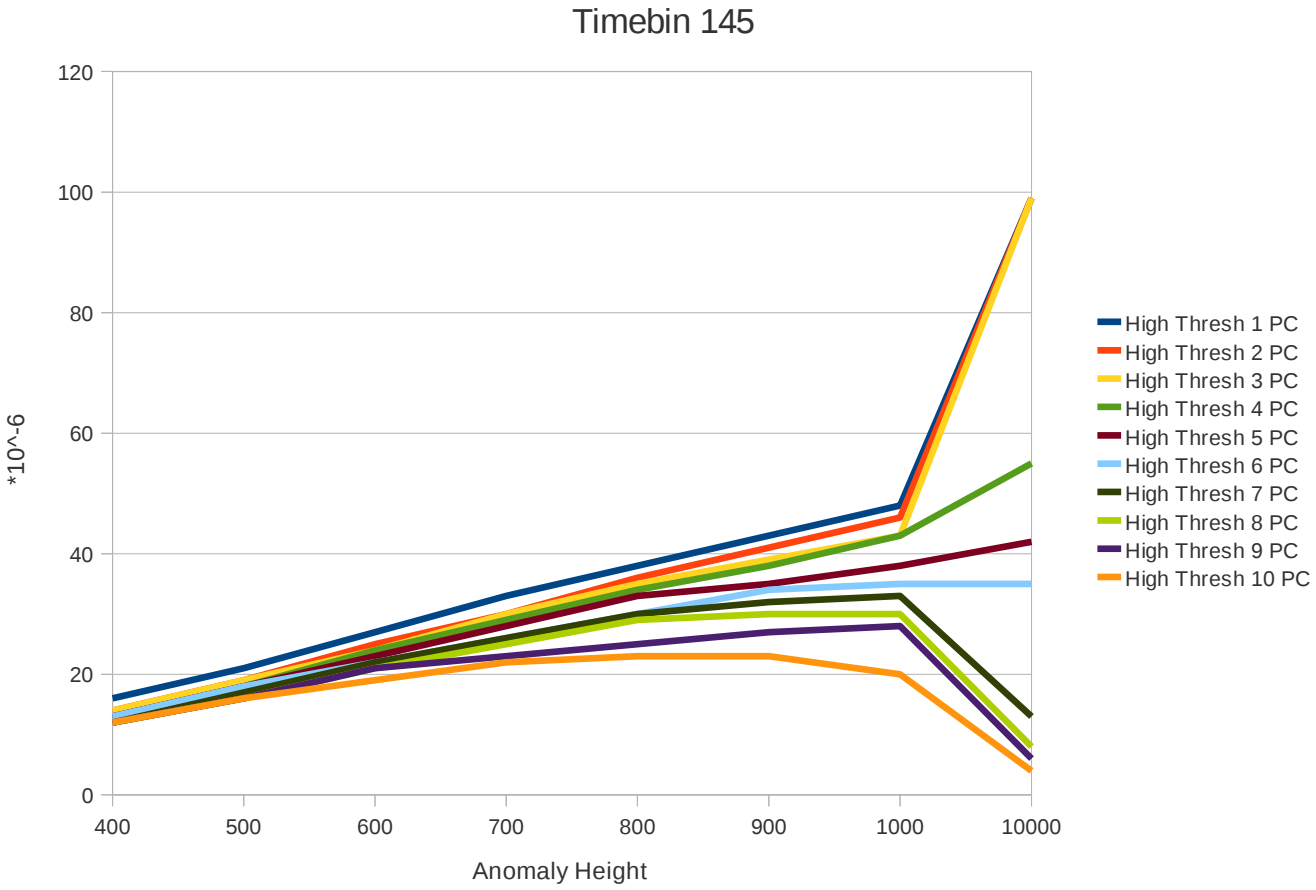
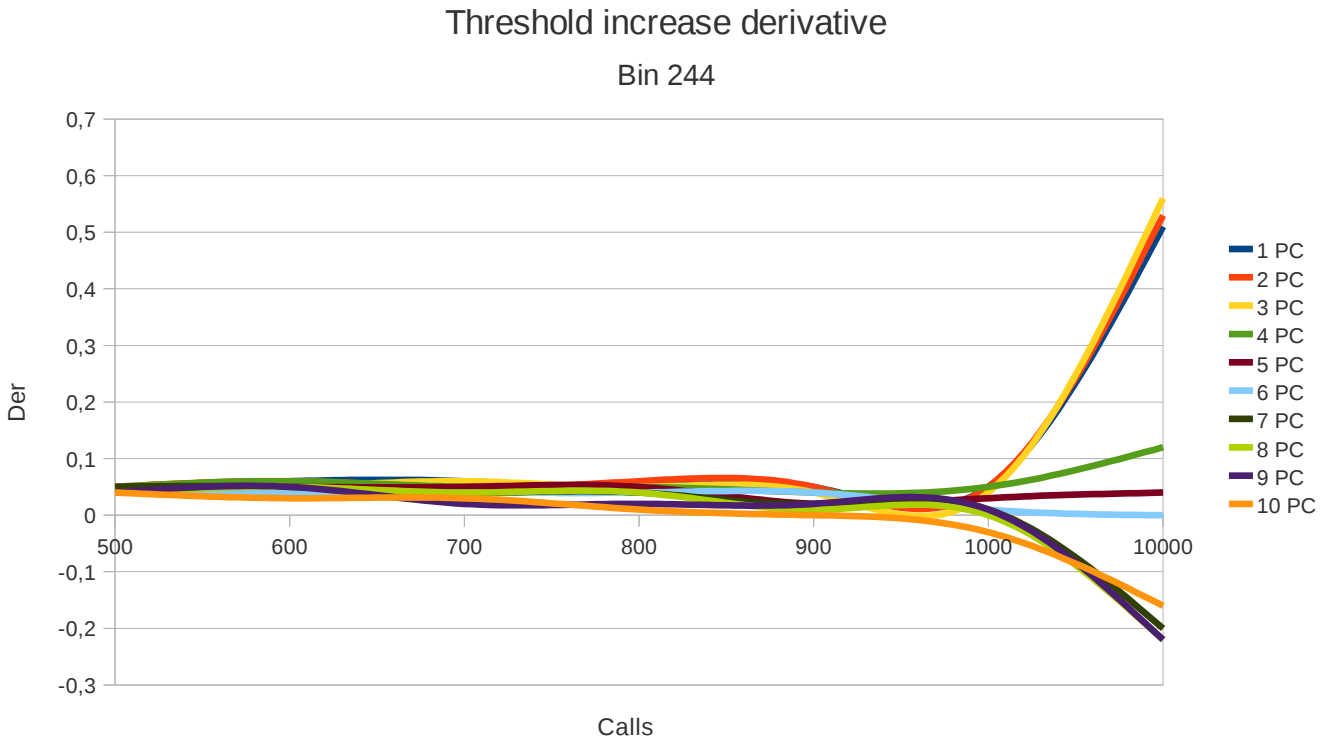


Figure 49



For time bin 145, the one with the behavior most dependent to the number of PCs, we can also observe the derivative of the thresholds increase (Figure 50), that shows us how a higher number of Principal Components makes the thresholds grow slower with the anomaly's height increasing, because of the fact that a higher number of PCs is most likely to include in the *normal* subspace even anomalous traffic, so it requires smaller anomalies to start suffering of this computational error.

Figure 50



The other types of considerations we can do about the thresholds regard the selection of the “best” threshold to be selected to obtain the best performances with a single analysis (one threshold and one number of PCs) using the software.
We used the two criteria introduced in the first chapter (number of anomalous users recognized upon the total number of anomalous users injected in the trace and number of anomalous time-bins detected upon the total number of anomalous time-bins present in the trace).
For each number of PCs and each threshold's value we computed these two ratios, obtaining results as the ones showed in Figure 51 (with 1 PC) and Figure 52 (with 10 PC).

Figure 51

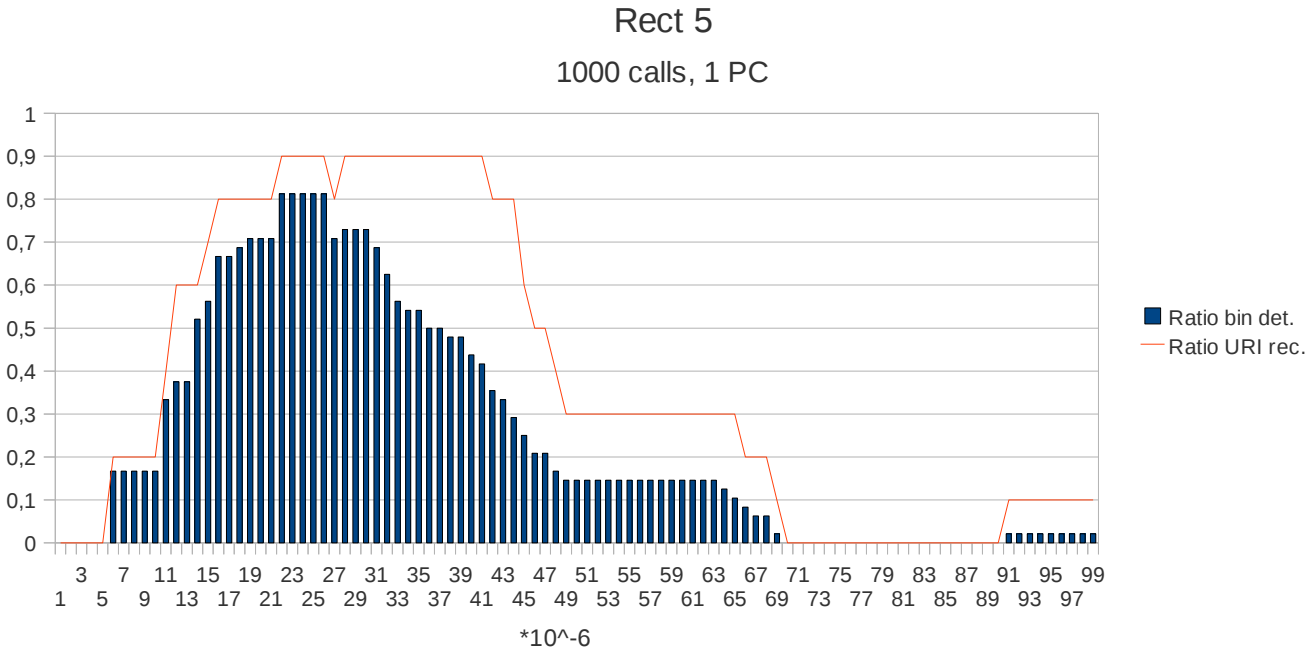
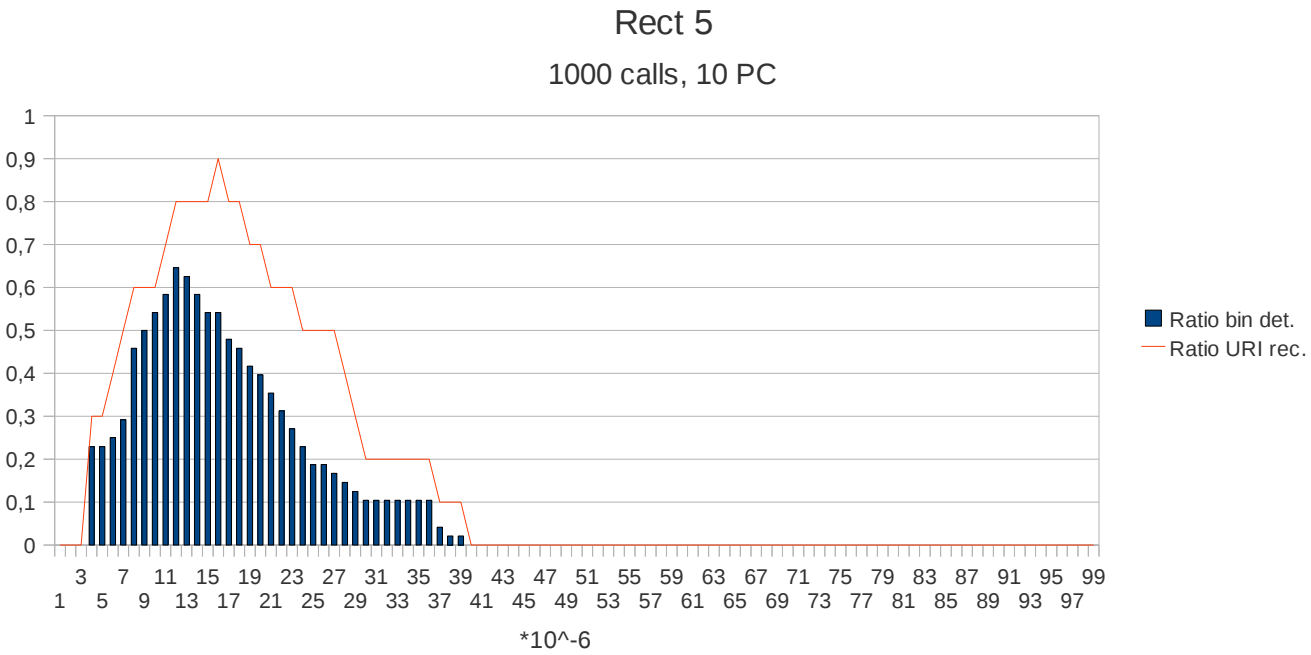


Figure 52



From these graphs we can observe the same tendency analyzed in the previous chapter, with the best threshold's value that tends to decrease with the increasing of the PCs number, as well as the width of the range of thresholds that permit the best performances. The graphs were plotted on the results obtained with an anomaly height of 1000 calls, so the fact that the maximum ratio of recognizable users is 9 on 10 is not due to the fact that the ranges of thresholds in which the anomalies were identified do not overlap, but simply to the fact that with 1000 calls in each anomalous time-bin we are not able to detect the anomaly in time-bins 490-494 with any threshold's value and any number of PCs (as we said before).

6.4 Rectangular anomalies, width 10

The successive step was using the same type of anomalies (rectangular shape) but with a longer duration: 10 consecutive time-bins instead of 5. We used the same starting time-bins, so we obtained two cases of overlapping: the two anomalies in time-bins 243-252 and 248-257, and the two anomalies in time-bins 490-499 and 493-502.
In Figure 53 we can observe the positions of the anomalous calls inserted in the trace, in the case of an anomaly's height of 1000 calls. In the analysis we used different heights, starting from 5 up to 1000 calls.
In Figure 54 we highlighted the anomalies positions, into the global call trace, for an anomaly height of 1000 calls.

Figure 53

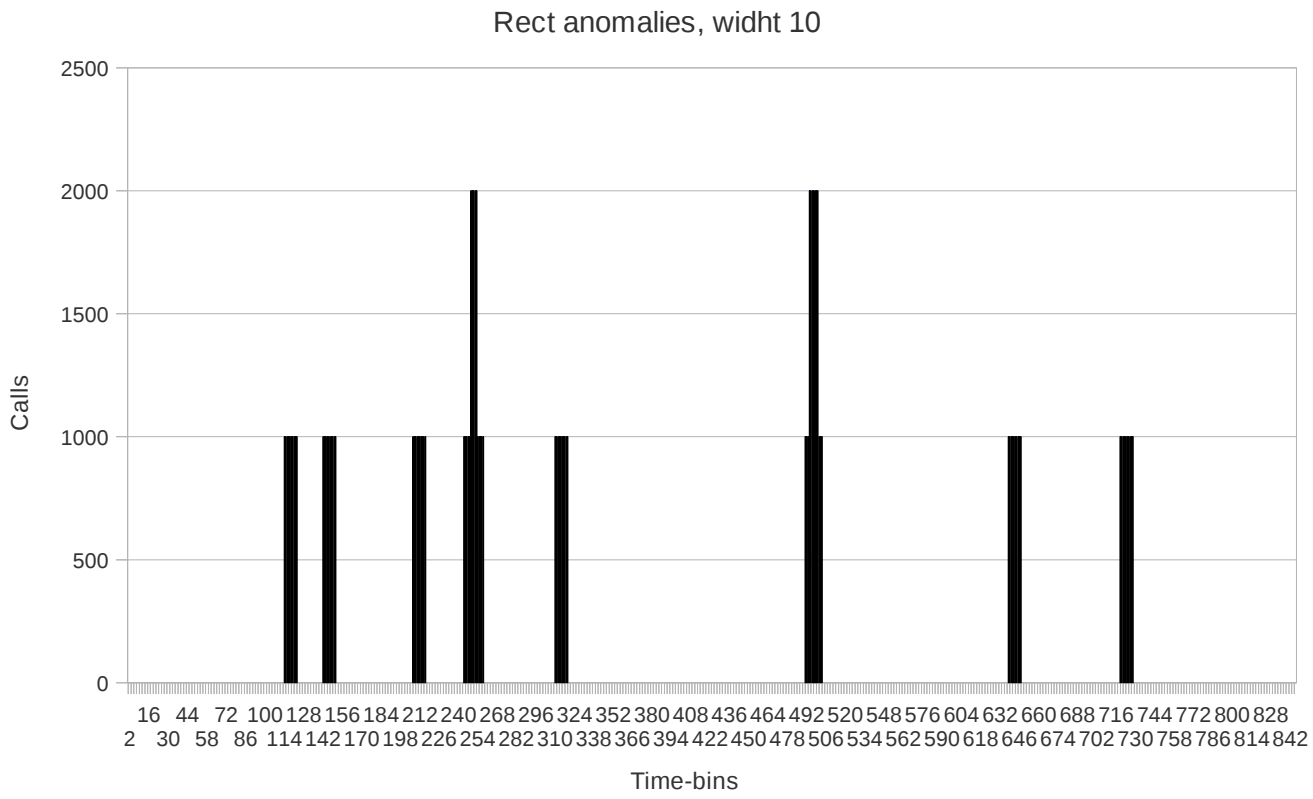
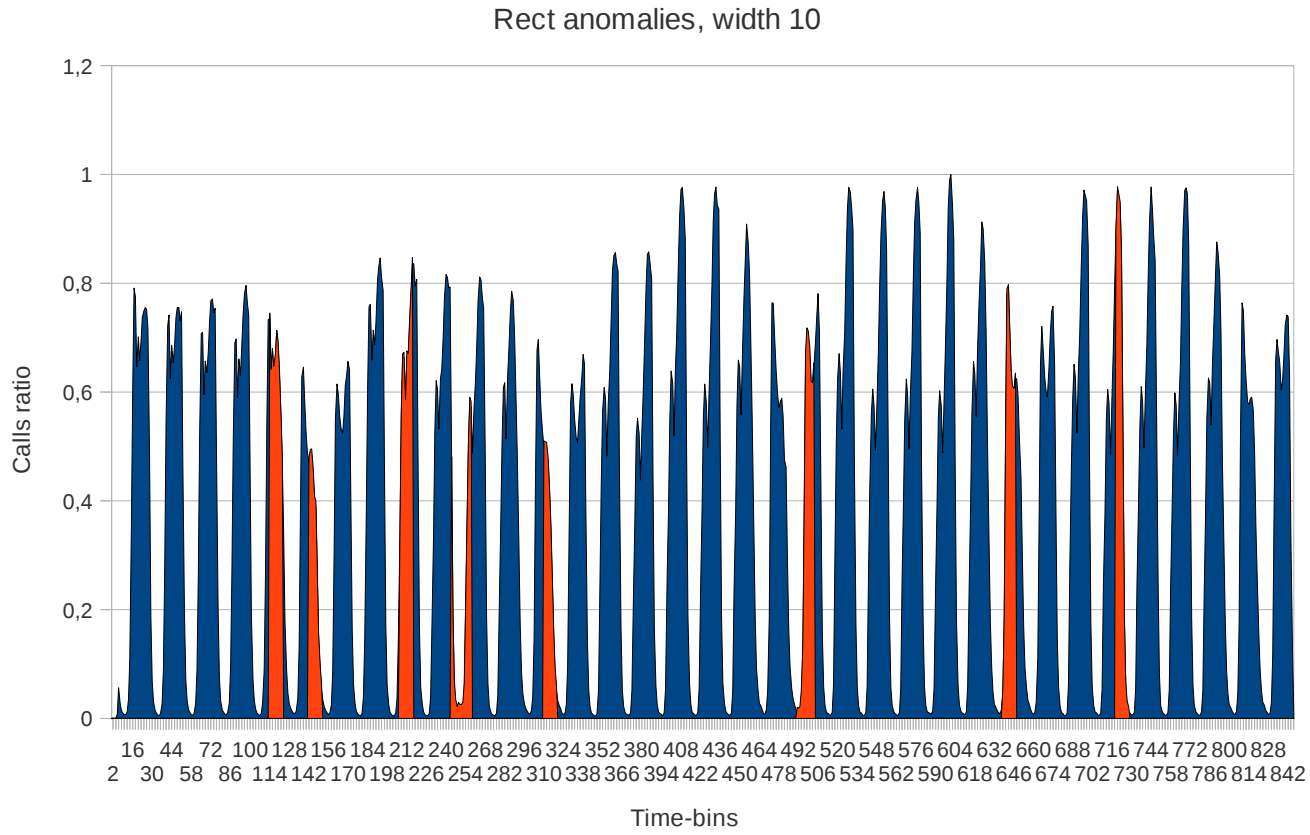


Figure 54



The first aspect we considered was, even in this case, the order in which the different anomalies were detected. We still have the problem that in the overlapping zones we are not able to detect more than one anomalous user, but in this case we are able to detect all 10 anomalies (in at least one time-bin each) with an anomaly's height lower than 1000 calls.
This is the order in which the anomalous users were detected (for every of them we indicated the first time-bin detected and the amount of global calls in that time-bin), starting from an anomaly's height of 60 calls (first detections) and ending when all the anomalies were detected (1000 calls):

1. Anomaly in time-bins 141-150 (8664 calls in the first time-bin detected: time-bin 149)
2. Anomaly in time-bins 718-727 (1073 calls: time-bin 727)
3. Anomaly in time-bins 309-318 (5029 calls: time-bin 318)
4. Anomaly in time-bins 243-252 (4320 calls: time-bin 252), with the first time-bin detected that is part of the overlapping zone
5. Anomaly in time-bins 637-646 (4568 calls: time-bin 637)
6. Anomaly in time-bins 248-257 (18888 calls: time-bin 253), with the first time-bin that is the the first one after the overlapping zone
7. Anomaly in time-bins 493-502 (5035 calls: time-bin 493), with the first time-bin that is part of the overlapping zone
8. Anomaly in time-bins 206-215 (32873 calls: time-bin 206)
9. Anomaly in time-bins 113-122 (46260 calls: time-bin 122)

10. Anomaly in time-bins 490-499 (65459 calls: time-bin 499)

In Figures 55 and 56 we plotted the number of times each time-bin was detected together with the global amount of calls' ratio in each time-bin, in the cases of 60 calls (firs time-bins detected) and 1000 calls (all anomalies detected, each of them in at least one time-bin).

Figure 55

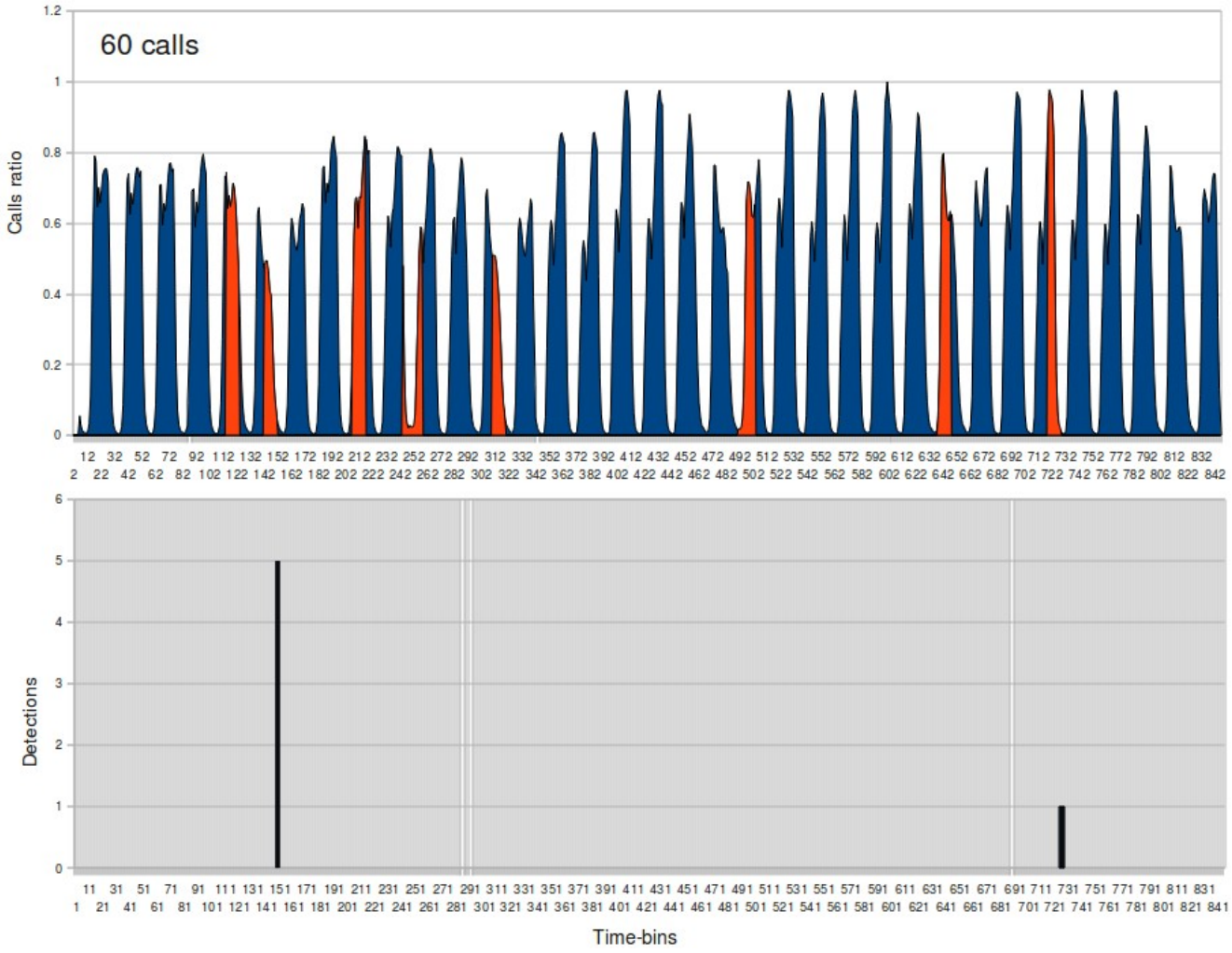
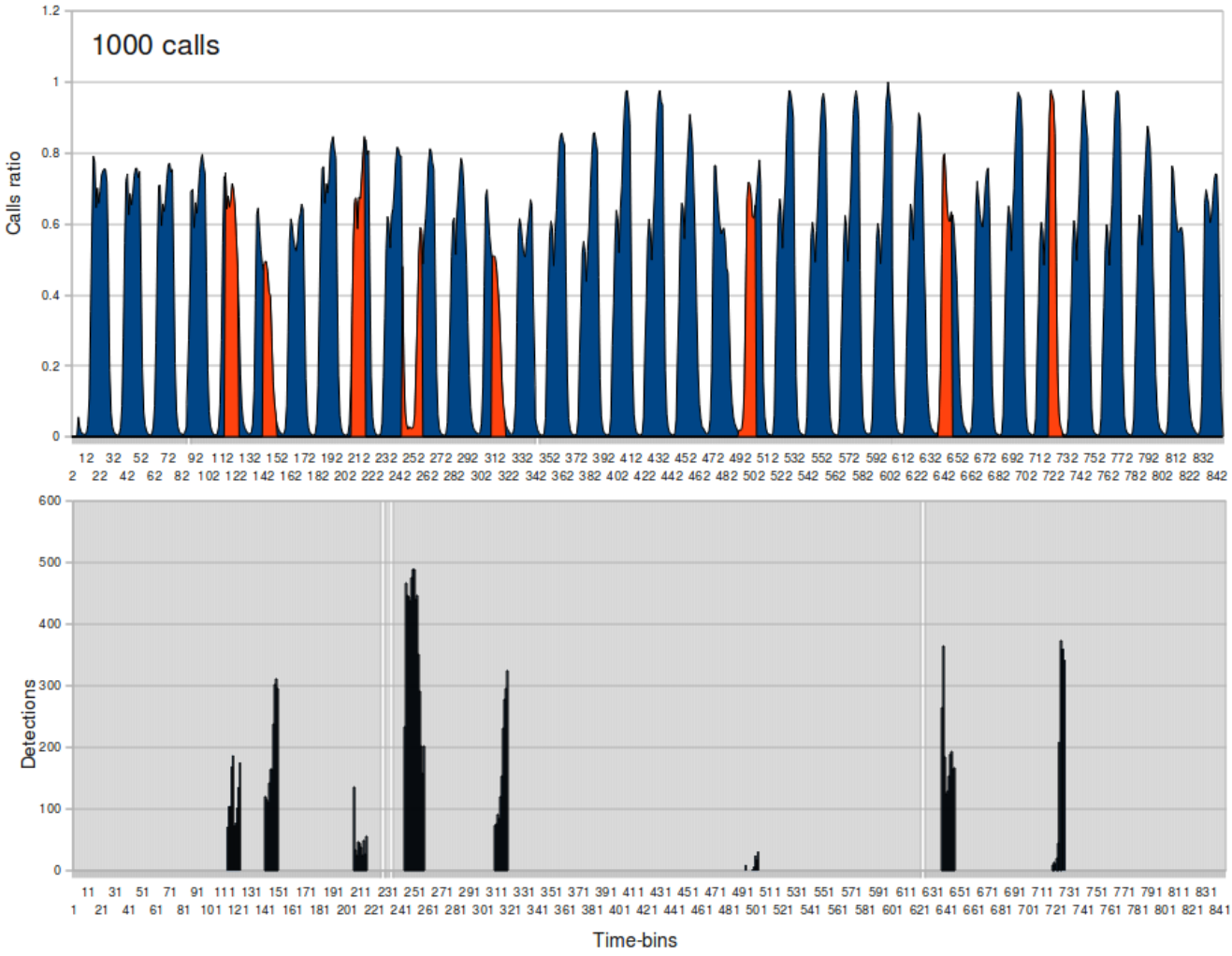


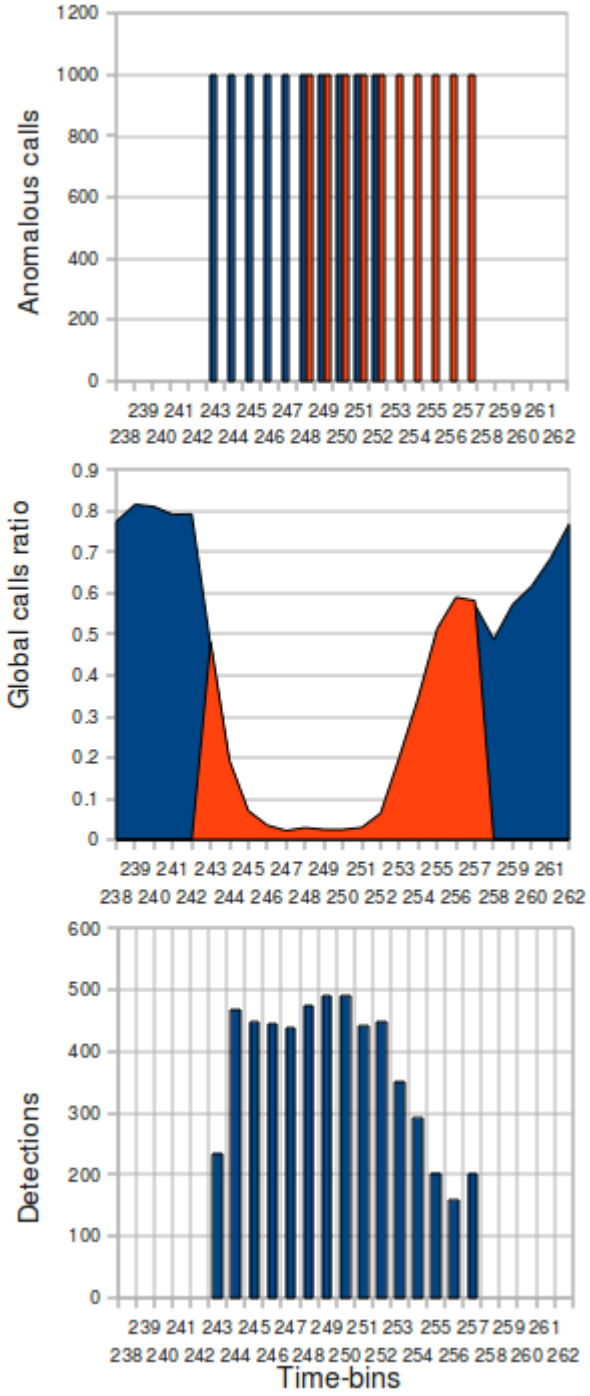
Figure 56



We can still observe the tendency to first detect anomalies that happen in time-bins with a lower global amount of calls, together with the one to detect first the border time-bins of each anomaly. This behavior is different in the overlapping zones, where the time-bins that are most likely to be detected are the ones on the border of this zones (even if they are not the first or the last time-bin of one of the two anomalies).

We can see these tendencies from Figure 57, where we plotted together the amount of anomalous calls in time-bins 243-257 (the different colors are because of the fact that in these time-bins there are two overlapping anomalies, so in time-bins from 248 to 252 the amount of anomalous calls is double), the ratio of global calls in the same time-bins and the number of detections obtained with a full analysis (all threshold and all numbers of PCs), with an anomaly's height of 1000 calls.

Figure 57

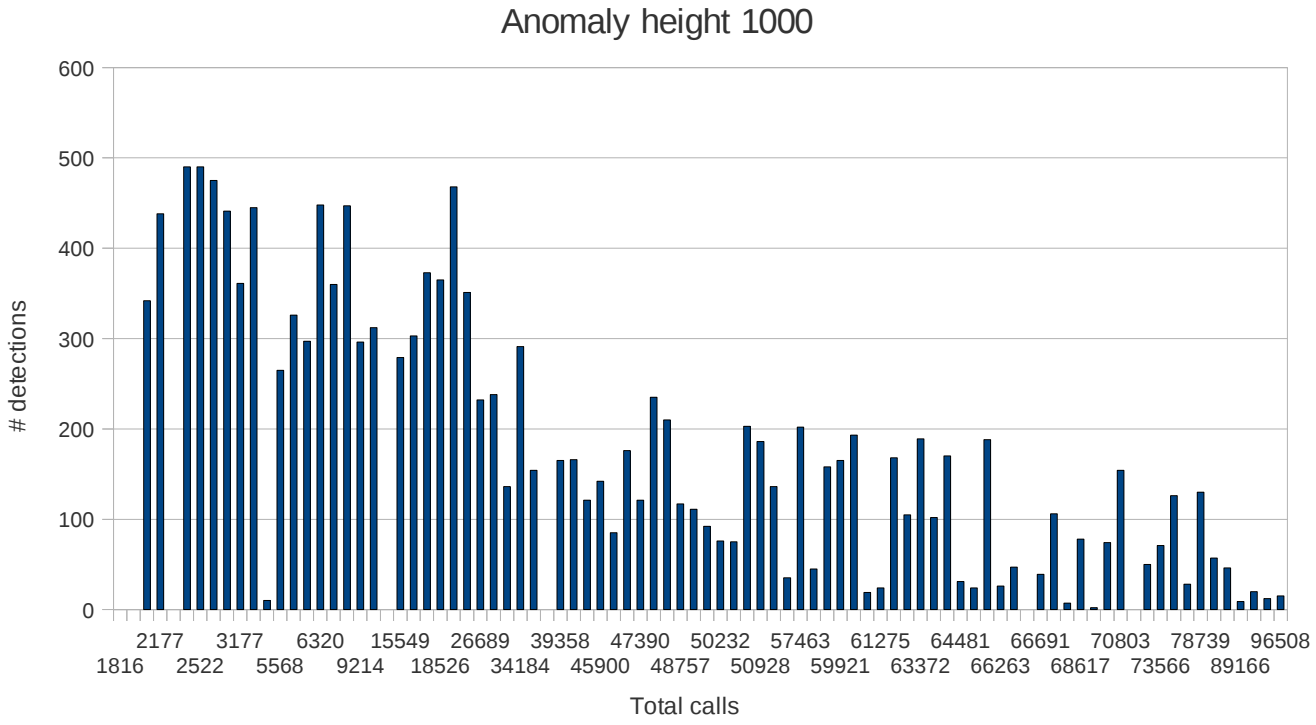


Watching the graphs we can see how the top-detected time-bins (that, as we said in the previous chapter, are not always correspondent to the first ones detected) are the ones in the overlapping zone, while out of that zone the detection's trend seems to follow the amount of global calls in the trace (more global calls in the time-bin means less detections), without any “special regard” to the border time-bins (since in this case the first ones detected are in the overlapping zone).

We can confirm the tendency of better detecting time-bins with a lower global amount of calls by plotting the number of times each anomalous time bin was detected (we didn't include not-anomalous time-bins, cause they were never been detected) versus the amount of global calls in that time-bin,

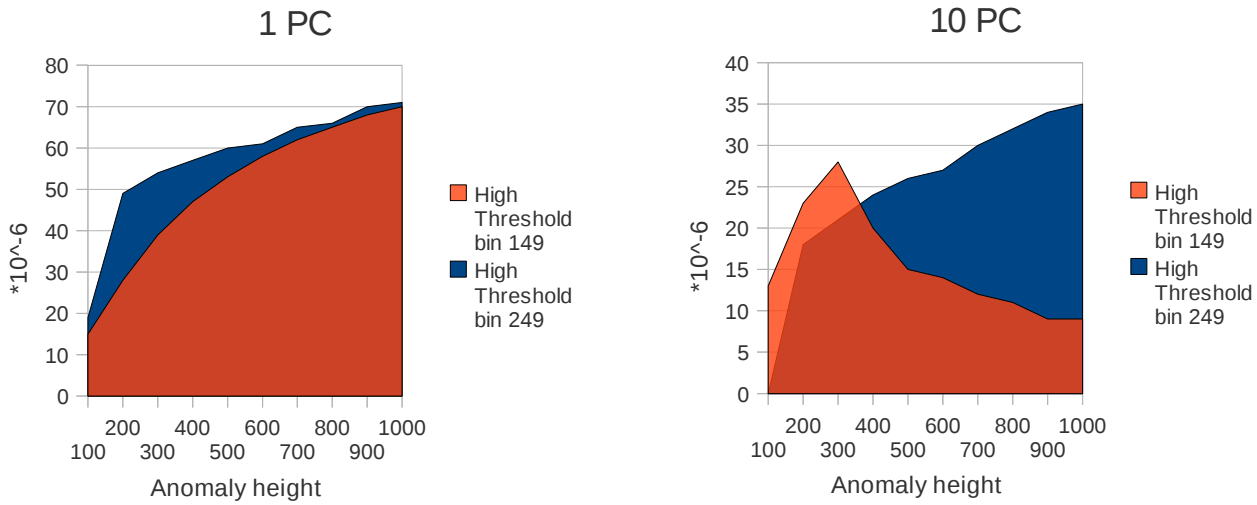
ordering the results by the number of global calls. We did it in Figure 58, with an anomaly's height of 1000 calls; with the exceptions of time-bins related with overlapping anomalies the number of times each time-bin is detected increases with the decreasing of the amount of global calls in the time-bin.

Figure 58



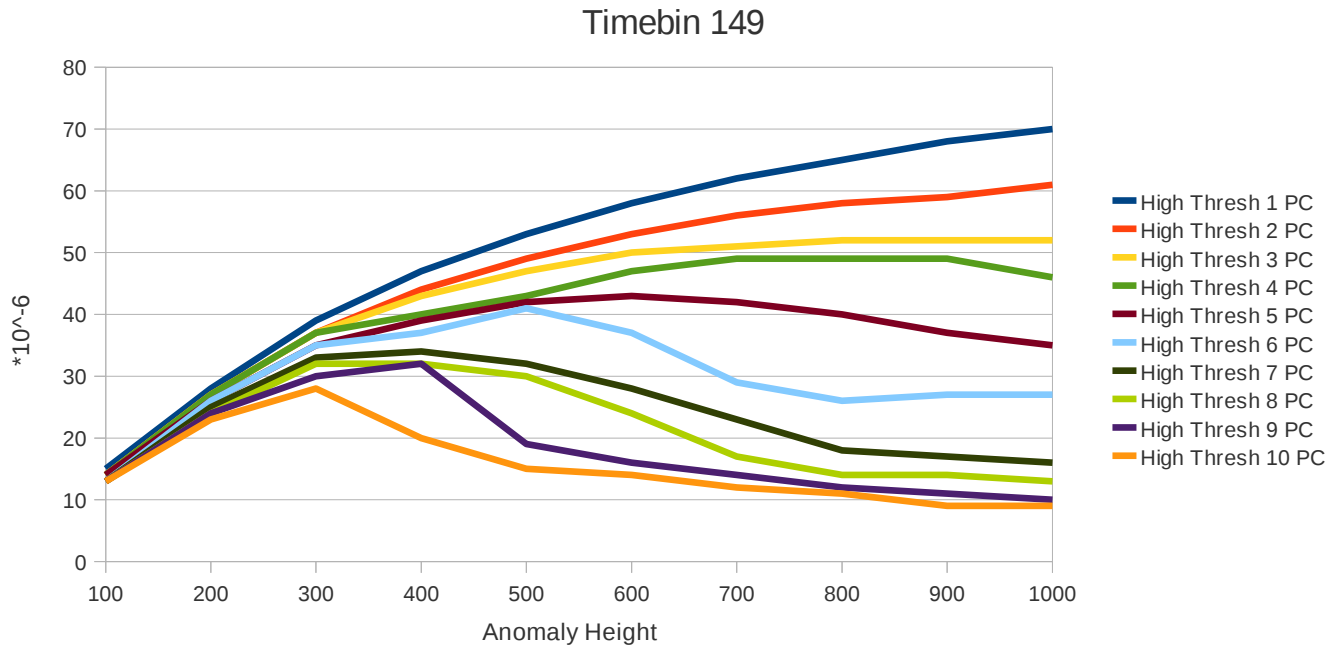
As with the other types of anomalies the second topic on which we focused is the sensitivity of the threshold to the anomaly's height and the number of Principal Components.
We first observed the thresholds range that permits the detection of a certain anomalous time-bin, as depending from the anomaly's size, varying the number of principal components used; we did this observation for two different time-bins: time bin 249 (the one with most detections with 1000 calls) and time bin 149 (the first one detected, with 60 calls).
In Figure 59 we plotted the maximum thresholds that permitted the identification of the anomalous users in time-bins 149 and 249 against the anomaly's height, for 1 and 10 Principal Components used.

Figure 59



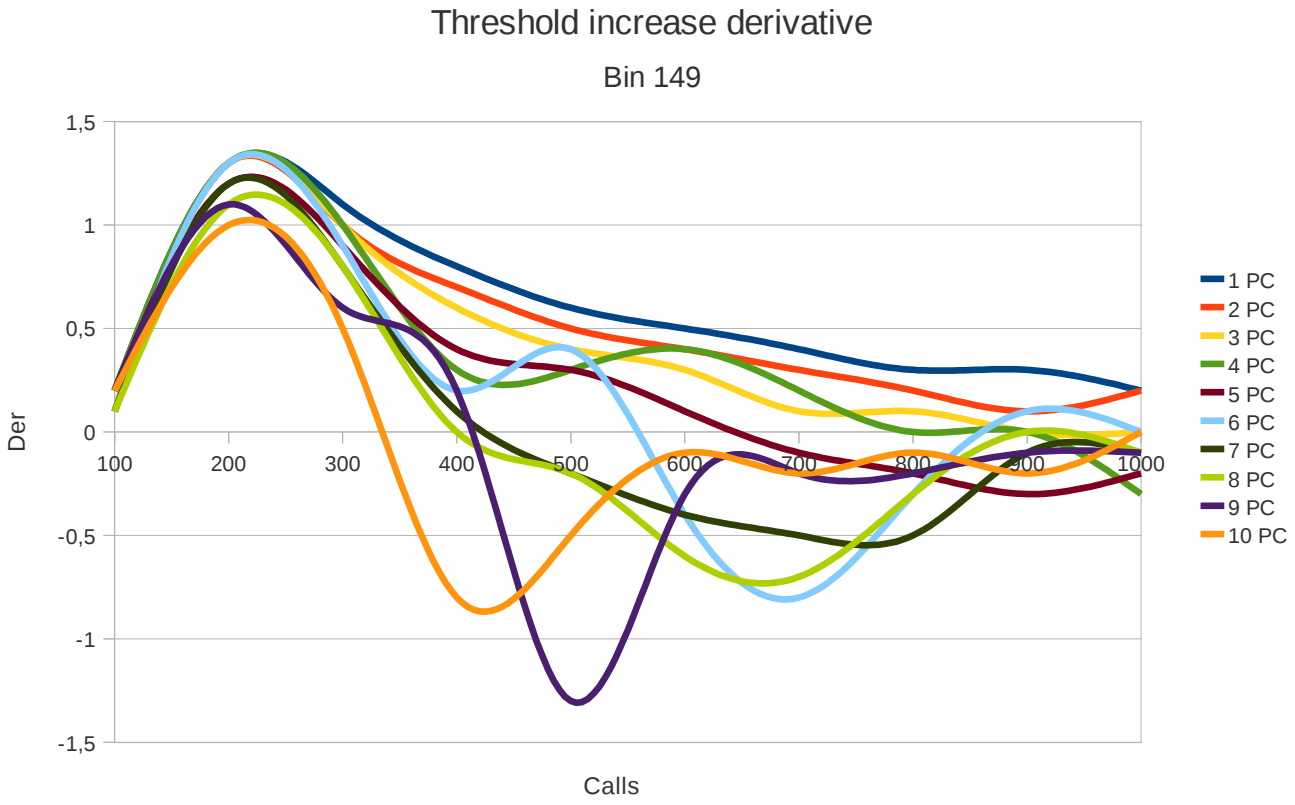
Looking at it we can see how time-bin 149 is better detected with a low anomaly's height, while time-bin 249 is better detected with a higher amount of calls in the anomalous time-bins; we can also observe how the software's performances in detecting time-bin 149 are more dependent on the number of PCs, as with 1 PC the thresholds tend to grow (from 100 to 1000 calls) with the anomaly height, while with 10 PC they start to decrease after reaching the top value with 300 calls.
To better understand this phenomenon we can plot the high threshold's values versus the anomaly's height, for all the 10 numbers of PCs we took into consideration in our analysis. We can see the results in Figure 60.

Figure 60



With 1 Principal Component the high threshold's value continues to grow up even with more than 1000 calls in the anomalous time-bins, but with higher numbers of PCs the behavior is different: we can observe that the number of PCs influences the point after which the threshold's values start to decrease, with this point's value that becomes smaller with a higher number of PCs used in the analysis. We think that this phenomenon has a general validity; we are not able to observe it in every time-bin just because the anomaly's height in which that peak value is reached is often bigger than the maximum amount of anomalous calls per time-bin that we took into consideration in our analysis.
We can confirm the considerations made before (and in the previous chapters) by plotting the derivative of the threshold's increase with different number of PCs: from the graphs we can observe how the derivative, for higher numbers of PCs, is always lower than the one correspondent to a lower number of PCs, and with the increasing of anomaly's height it can become negative (this happens earlier for higher numbers of PCs, as we just said). We did this in Figure 61.

Figure 61



As in the previous chapters, the other types of considerations we can do about the thresholds regard the selection of the “best” threshold to be selected to obtain the best performances with a single analysis (one threshold and one number of PCs) using the software.
We used the two criteria introduced in the first chapter (number of anomalous users recognized upon the total number of anomalous users injected in the trace and number of anomalous time-bins detected

upon the total number of anomalous time-bins present in the trace).
For each number of PCs and each threshold's value we computed these two ratios, obtaining results as the ones showed in Figure 62 (with 1 PC) and Figure 63 (with 10 PC).

Figure 62

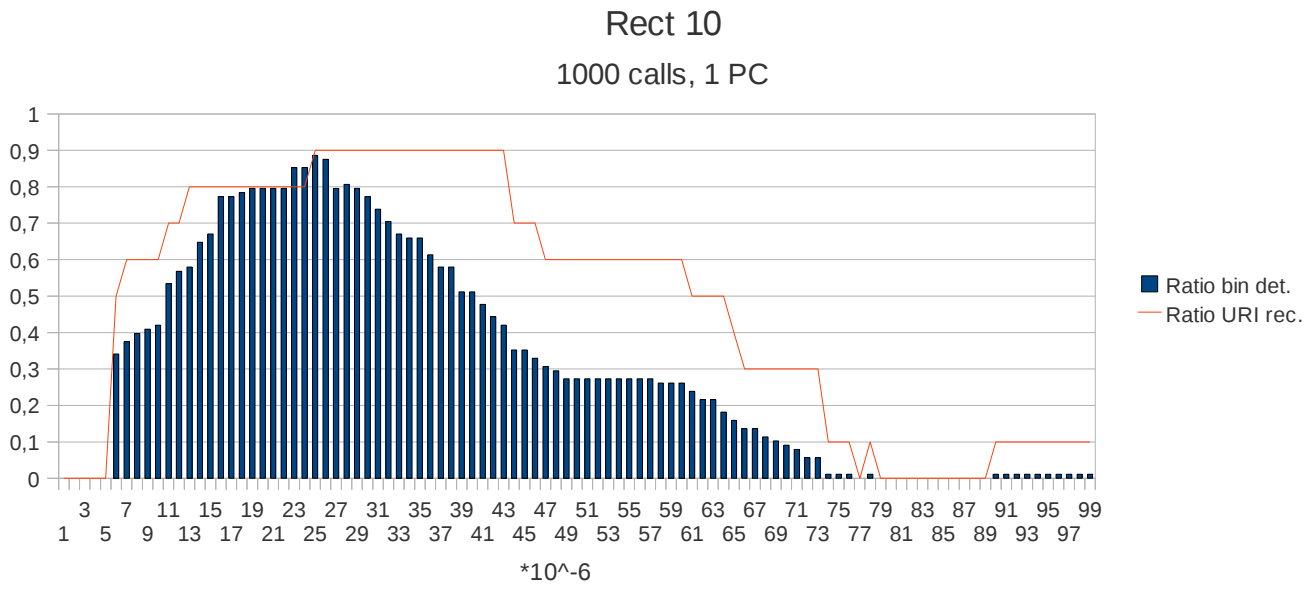
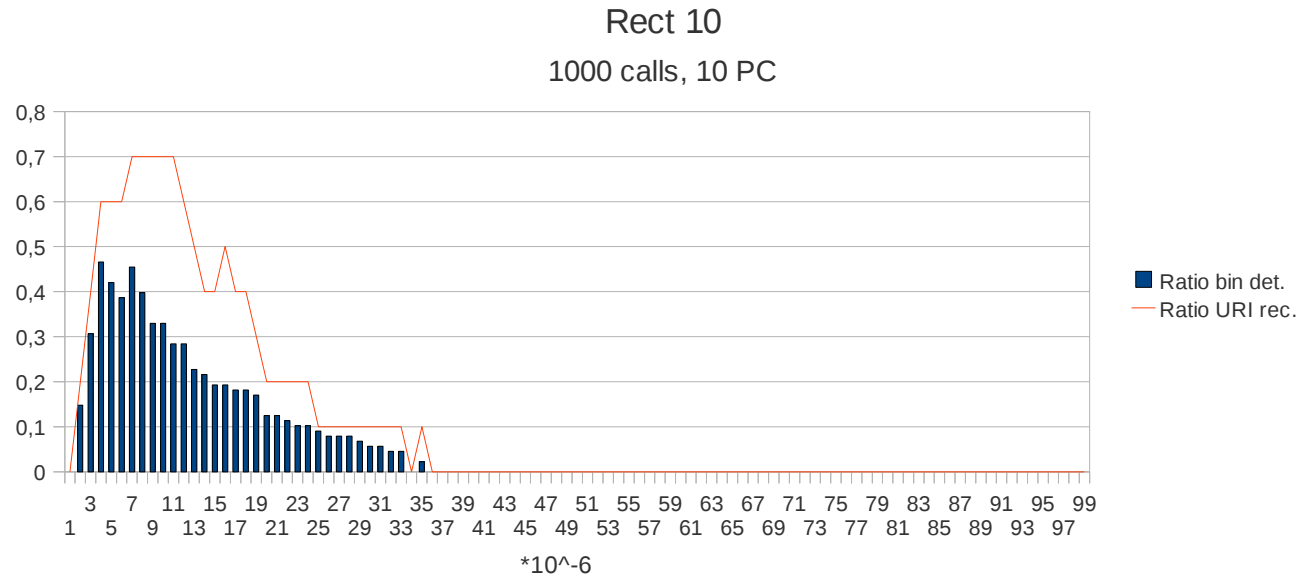


Figure 63



We can still observe how the best threshold's value and the range of thresholds that permit the best

performances decrease with the number of PCs (as we've already seen in the previous chapters). In this case we can also notice a heavier dependence of the values of the two ratios on the number of PCs: with 1 PC for example we are able to detect a maximum of 9 anomalous users on 10 and up to the 90% of anomalous time-bins, while with 10 PCs we are only able to detect a maximum of 7 anomalous users and less than 50% of anomalous time-bins. This is probably depending on the fact that anomalies are wider, so a higher anomaly's height can have a major influence on the *normal* subspace's modeling (more than in the previous cases where the anomalies where more narrow), fact that leads to a bigger performance's worsening.

6.5 Triangular anomalies, width 5

The successive type of artificial anomalies we inserted in the trace was triangular ones: we inserted 10 new URIs in the trace that remain silent for the whole duration of the trace, except for 5/10 consecutive time-bins (5/10 hours), different for each URI, in which they have a linear increase of calls, followed by a linear decrease.
The first analysis we made was with anomalies that lasted a total of 5 consecutive time-bins (with a linear increase in the first 3, a top in the third one, and a linear decrease in the last two).
We used the same starting time-bins that we used for the previous types of anomalies so, as in the case of rectangular anomalies with 5 hours of width, we obtained only one case of overlapping, with the 2 anomalies in time-bins 491-495 and 494-498.
In this case we considered as anomaly's height the top value, reached in the third anomalous time-bin by each anomalous user; we started the analysis with 5 calls and we tested the software's performances with up to 1000 calls.
In Figure 64 we plotted the ratio of anomalous calls per time-bin, with an anomaly's height of 1000 calls.

Figure 64

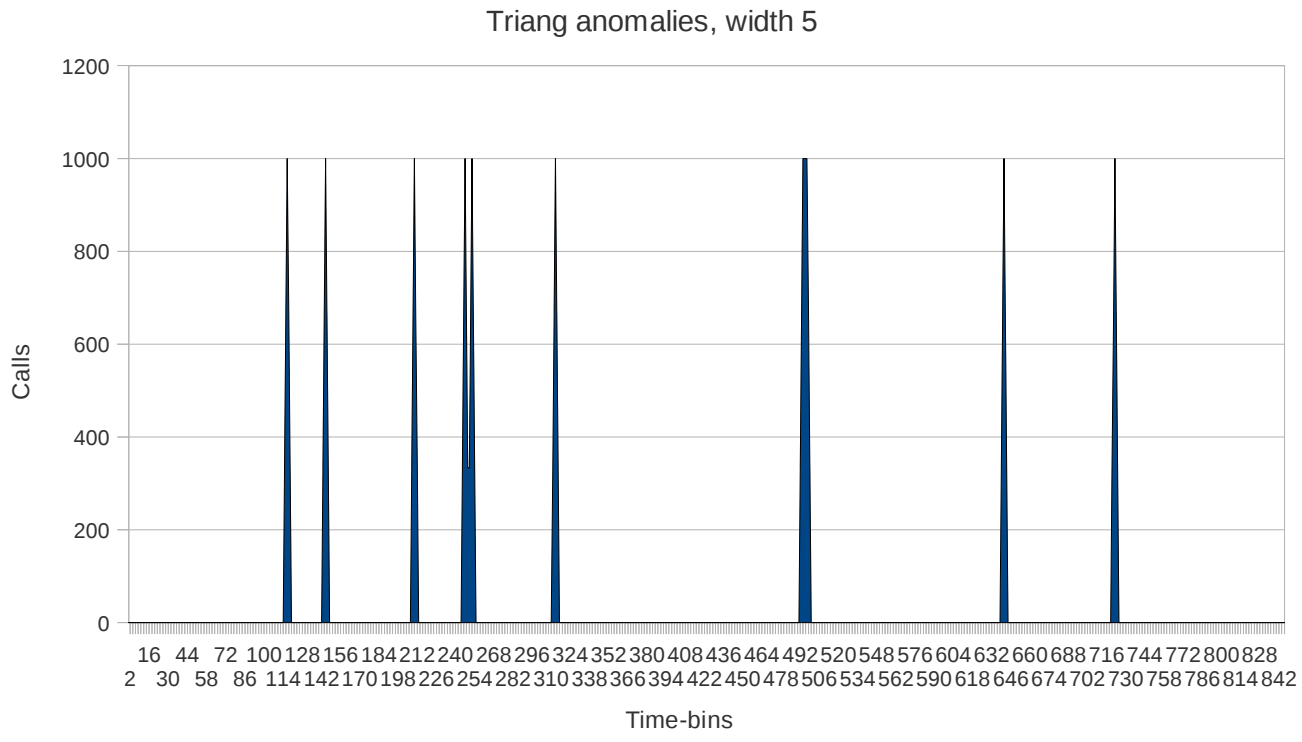
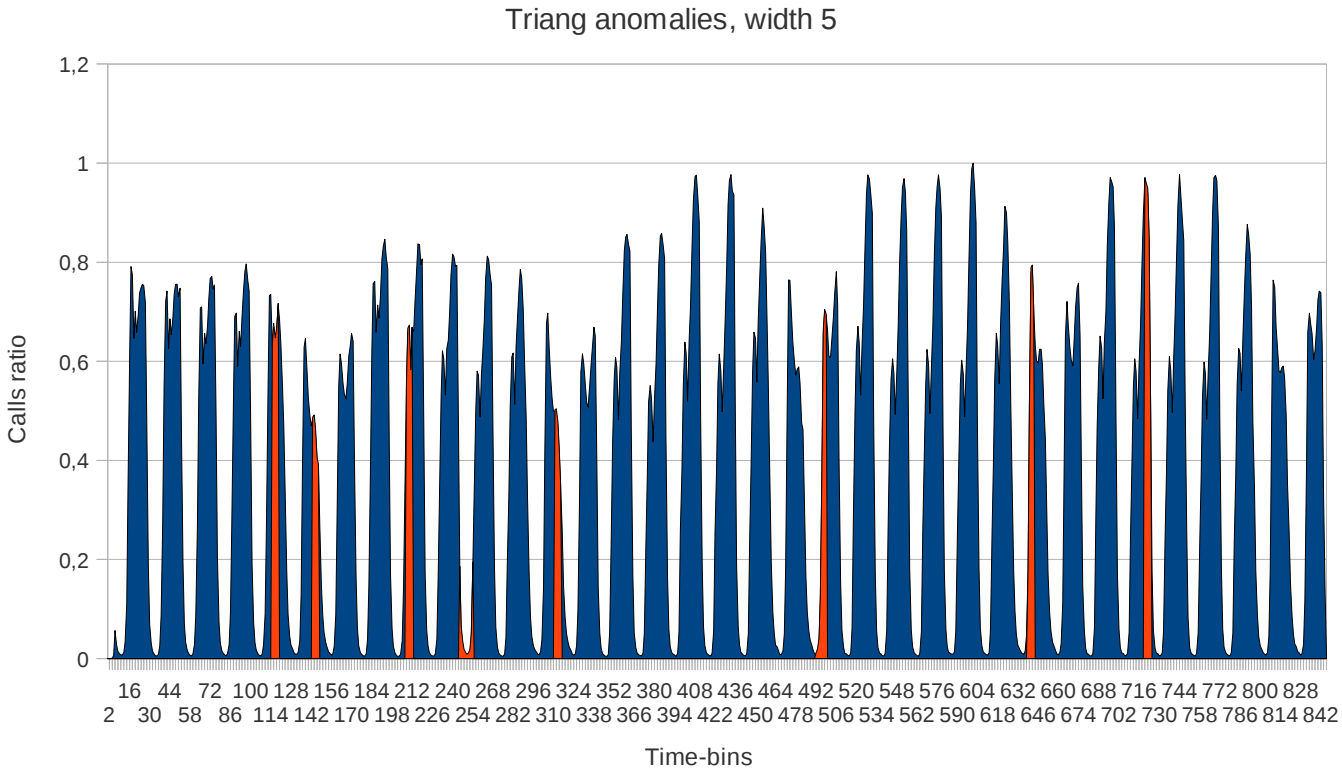


Figure 65



In Figure 65 we plotted the ratio of global calls per time-bin (on their maximum value), highlighting the positions in which we inserted the anomalies.
As with other types of anomalies, the first aspect we considered, in analyzing the results, is the order in which the anomalies were detected.
In this case we manage to identify all 10 anomalous users in the trace, with less than 1000 calls (in this aspect we have better results than with rectangular anomalies of the same size).
The order in which the anomalies were detected is the following (we use the same format used in the previous chapters, starting from anomaly's height of 100 calls, when we have the fist time-bins detected):

1. Anomaly in time-bins 491-495 (11715 calls in the first time-bin detected: time-bin 494), with the first time-bin detected that is part of the overlapping zone
2. Anomaly in time-bins 249-253 (4320 calls: time-bin 252)
3. Anomaly in time-bins 244-248 (2476 calls: time-bin 246)
4. Anomaly in time-bins 638-642 (76961 calls: time-bin 640)
5. Anomaly in time-bins 142-146 (44900 calls: time-bin 144)
6. Anomaly in time-bins 310-314 (46181 calls: time-bin 312)
7. Anomaly in time-bins 207-211 (65462 calls: time-bin 209)
8. Anomaly in time-bins 494-498 (64495 calls: time-bin 496), with the first time-bin detected that is part of the overlapping zone
9. Anomaly in time-bins 114-118 (62923 calls: time-bin 116)
10. Anomaly in time-bins 719-723 (92713 calls: time-bin 721)

From the order of detections we are not able anymore to recognize the tendency of better detecting time-bins with a lower amount of global-calls, while we can see that, except for the first anomalous user detected (the detection of which is influenced by the fact that it is part of an overlapping zone), all the anomalies were first detected in their central hour, the one with the maximum peak of calls.
In Figure 66 and 67 we can see the amount of detections per time-bin, together with the global calls' ratio per time-bin, in cases of anomaly's heights of 100 and 1000 calls.

Figure 66

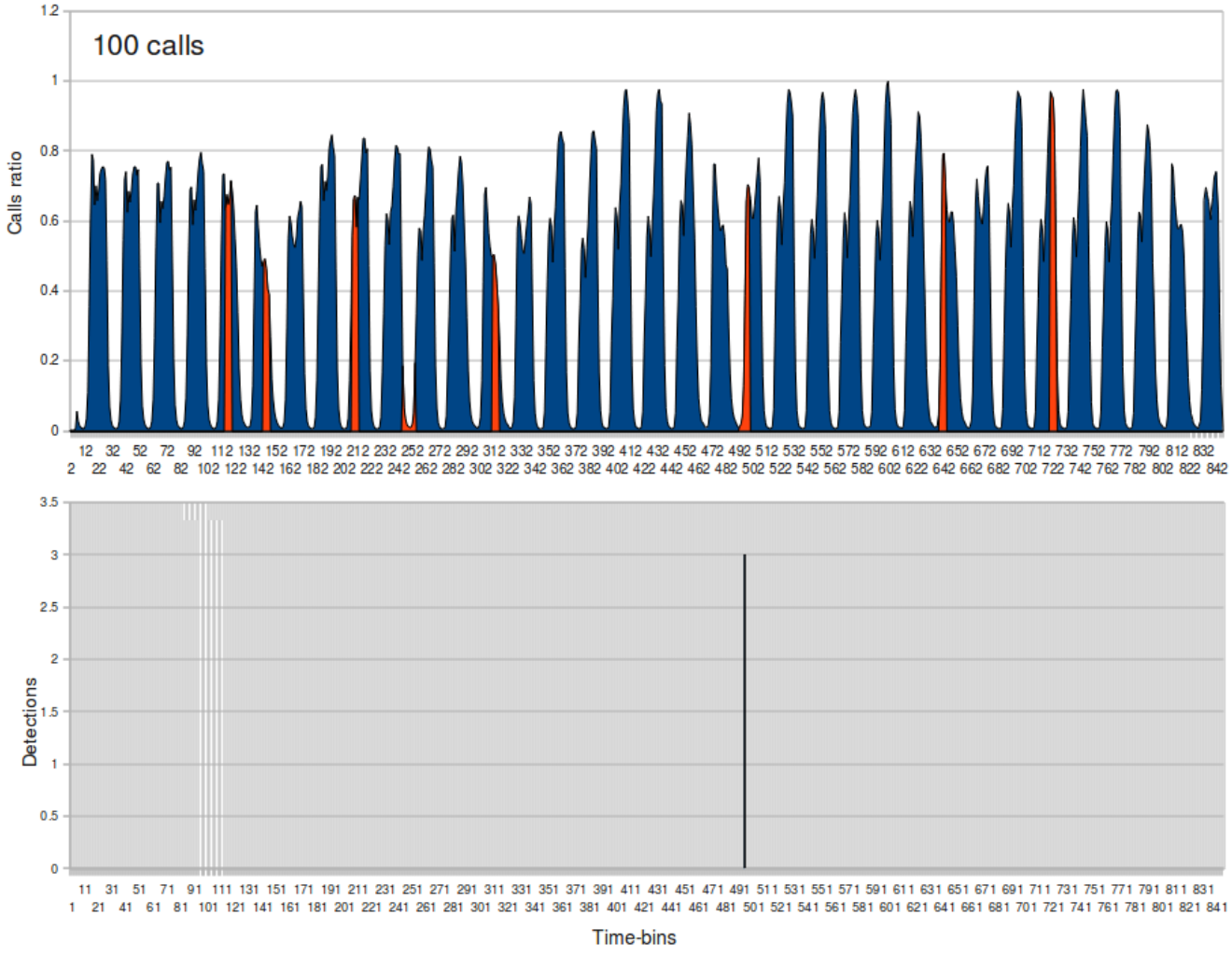
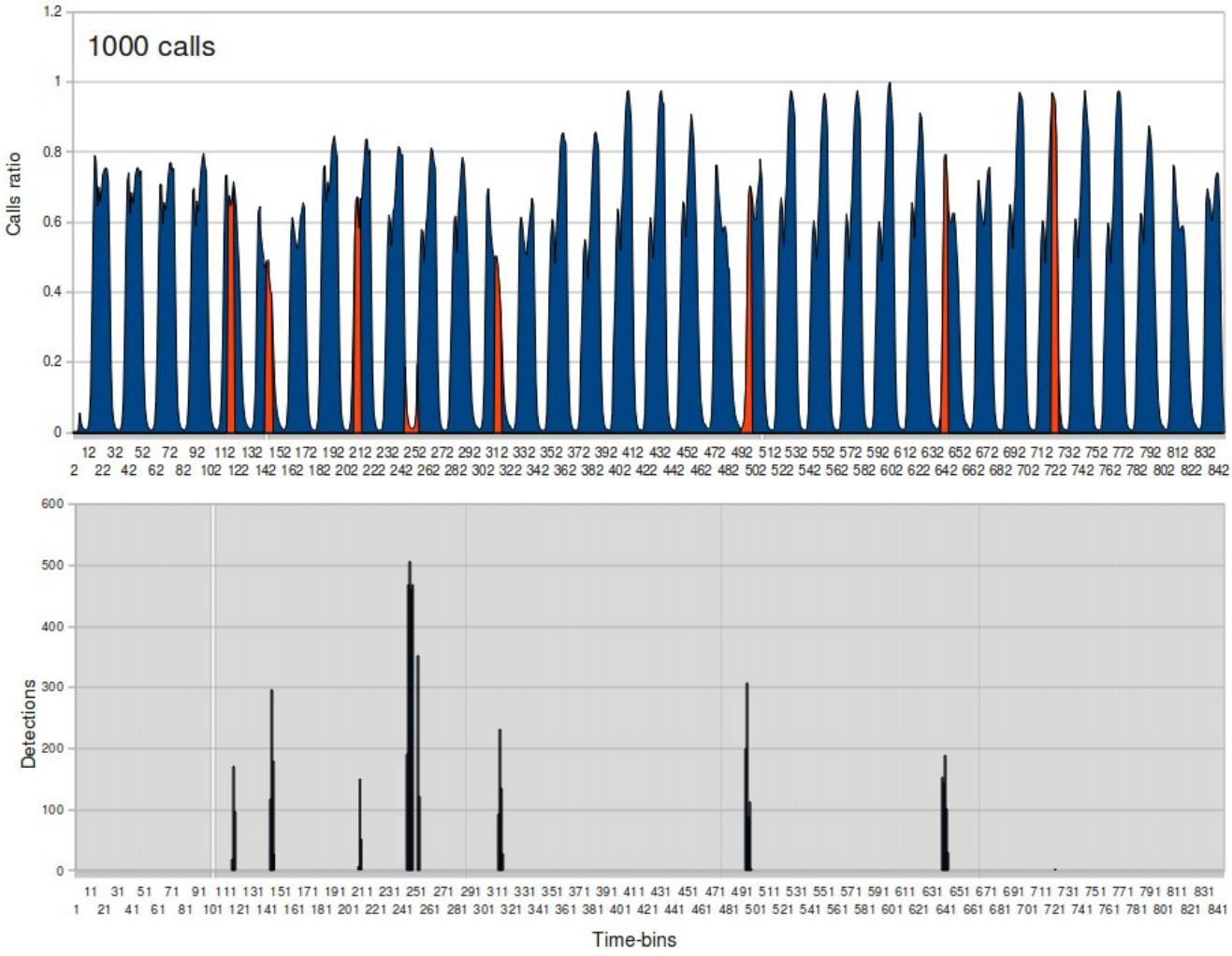
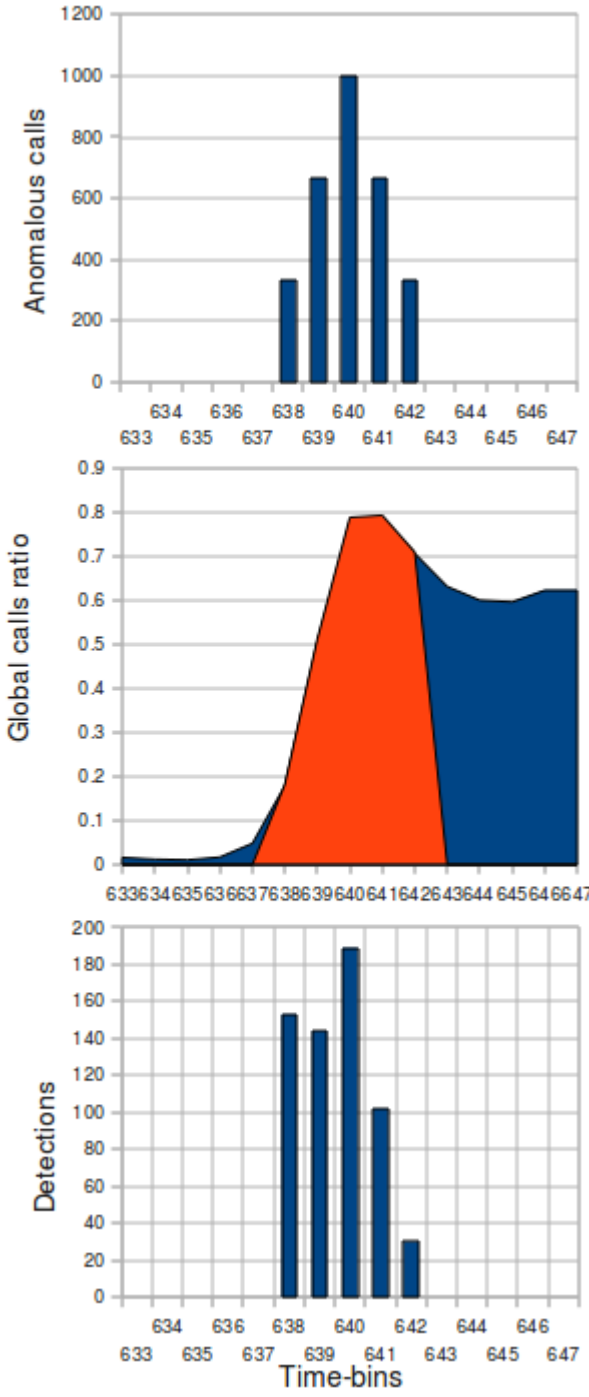


Figure 67



In order to better see this type of behavior we can focus on a single anomaly, considering the number of times each of the time-bins that compose that anomaly is detected with an anomaly's height of 1000 calls. We did this for the anomaly in time-bins 638-642, and in Figure 68 we plotted the number of anomalous calls per time-bin, the ratio of global calls per time-bin and the number of detections per time-bin. Comparing the three graphs we can understand how the software's performances in detecting each time-bin are dependent both on the amount of anomalous calls (the anomaly structure, as we have seen in the previous chapters where there was the tendency to better detect border time-bins) and on the amount of global calls (with the tendency to still better detect time-bins with less global calls).

Figure 68

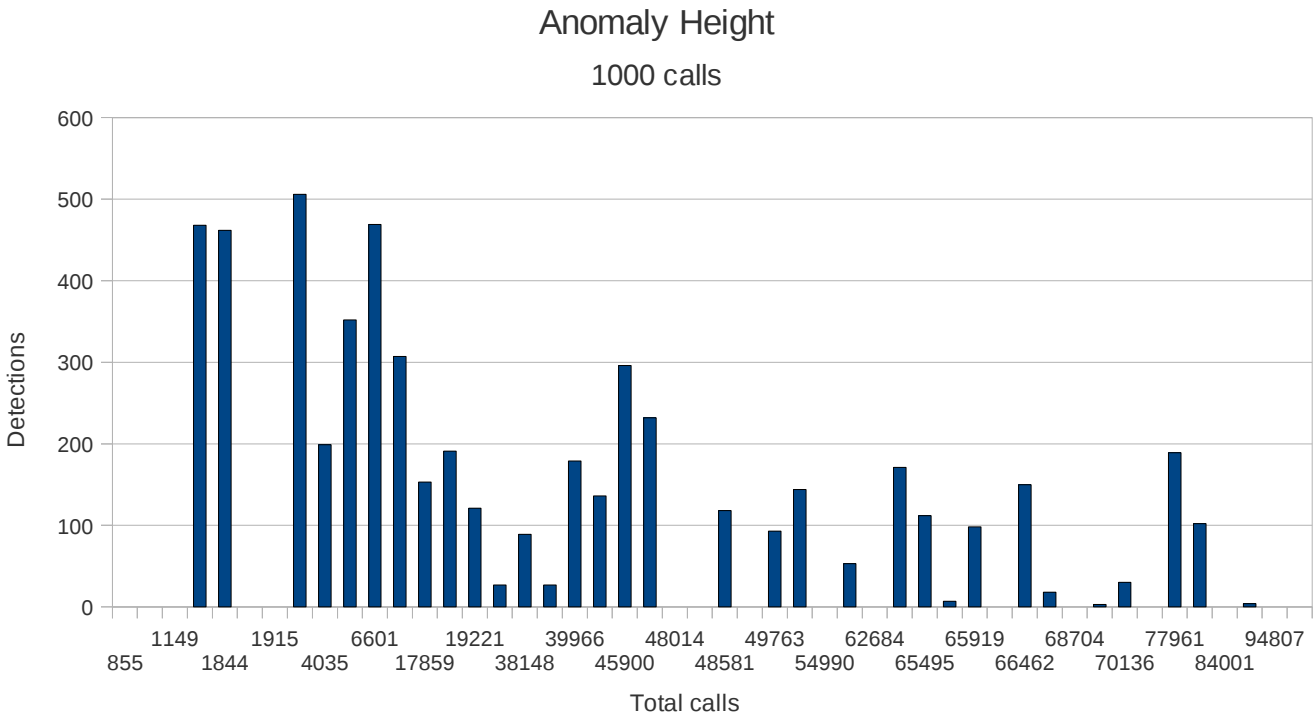


From the graphs we can see how the central time-bin (the one with the highest amount of anomalous calls) tends to be the most detected, while in other time-bins we can still observe the tendency to have a higher amount of detections in time-bins with a lower amount of global calls.
In general the software's capability of detecting a certain time-bin is a combination of these two factors.

To verify that the tendency of better detecting time-bins with less anomalous calls is still present we can plot the number of times each anomalous time-bin was detected (we didn't included not-anomalous time-bins, cause they were never been detected) versus the amount of global calls in that time-bin, ordering the results by the number of global calls. We did in Figure 69, with an anomaly's height of

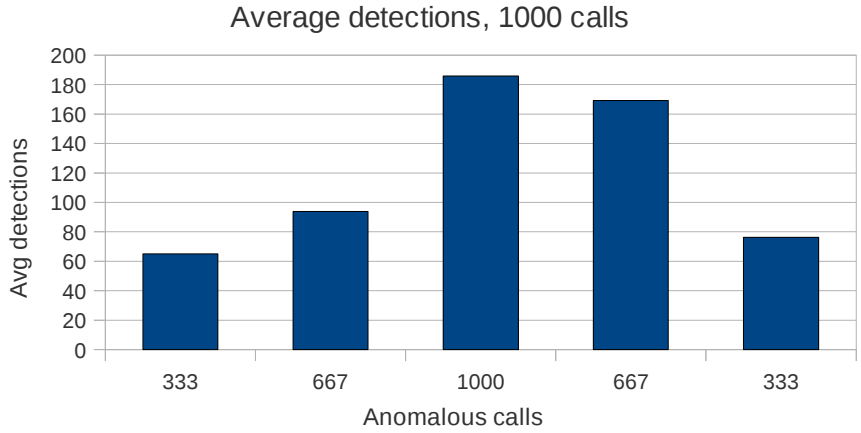
1000 calls; we can observe that this tendency is still present.

Figure 69



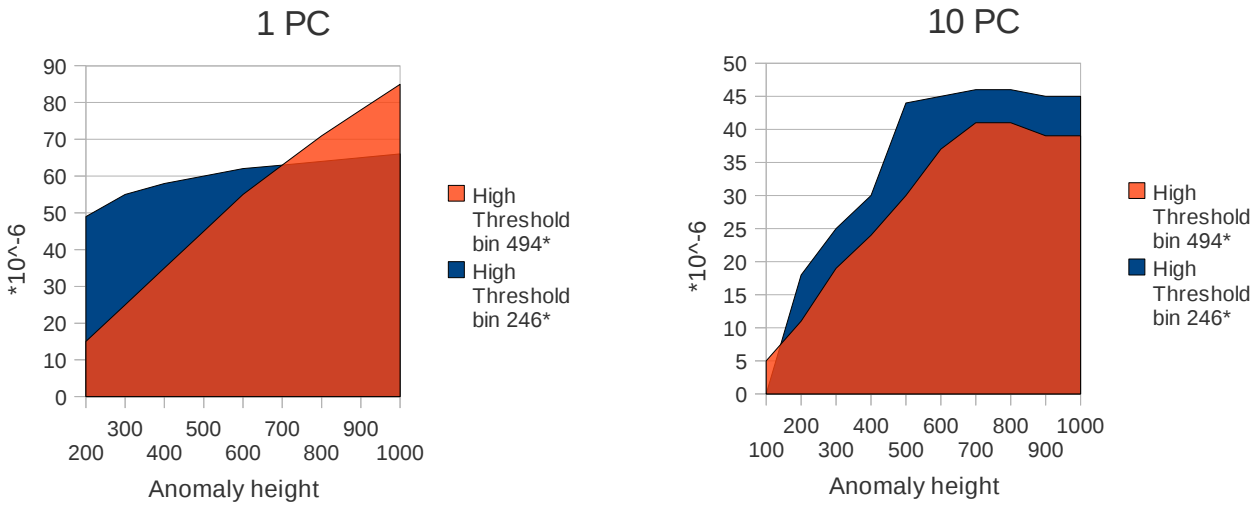
To verify the tendency to better detect central time-bins of triangular anomalies we decided to plot the average amount of detections for each type of time-bin (first time-bin, second, third, fourth and fifth), with an anomaly's height of 1000 calls (correspondent to the amount of anomalous calls in the third time-bins). We show this kind of graph in Figure 70, where on the x-axis we wrote the amount of anomalous calls in each type of time-bin; we computed the mean number of detections along the 10 anomalies inserted in the trace.
We can easily recognize the tendency to better detect time-bins with a higher amount of anomalous calls (in particular the central ones, that correspond to the highest peaks).

Figure 70



As with the other types of anomalies the second topic on which we focused is the sensitivity of the threshold to the anomaly's height and the number of Principal Components.
We first observed the thresholds range that permits the detection of a certain anomalous time-bin, as depending from the anomaly's size, varying the number of Principal Components used; we did this observation for two different time-bins: time bin 246 (the one with most detections with 1000 calls) and time bin 494 (the first one detected, with 100 calls).
In Figure 71 we plotted the maximum thresholds that permitted the identification of the anomalous users in time-bins 246 and 494 against the anomaly's height, for 1 and 10 Principal Components used.

Figure 71



We can make the same considerations we did in the previous chapters, observing how the threshold's behavior is strictly dependent on the number of PCs used. For example with 1 PC we can see that time-bin 246 is better detected with low anomaly's height and time-bin 494 is better detected with high anomaly's height, while with 10 PCs we have the opposite situation.
We can also see how a higher number of PCs leads to a faster reaching of the maximum threshold, that is reached with a lower anomaly's height than in the case of a lower number of PCs; after reaching that value thresholds start to decrease with the increasing of anomaly's height, as we observed in the previous chapter.
Further confirmations can be found observing Figure 72 and 73 where we plotted the high thresholds and the threshold's increase derivative against the anomaly's height for each number of PCs.

Figure 72

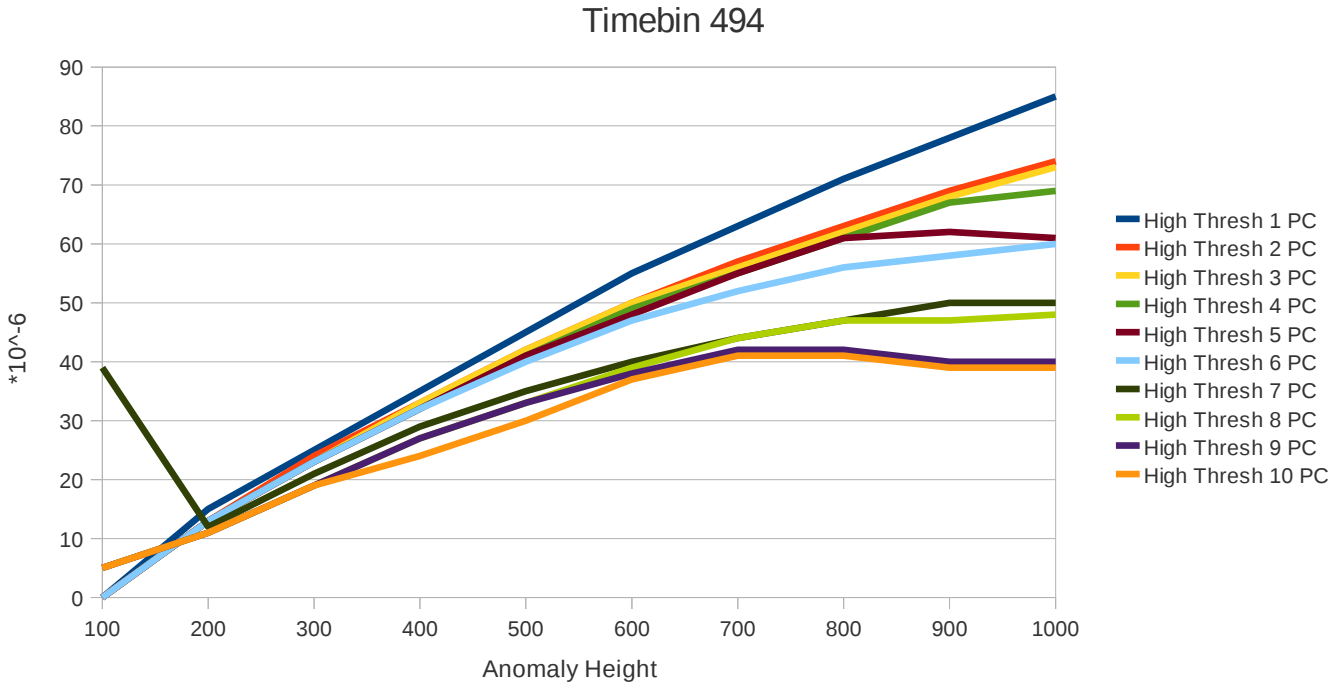
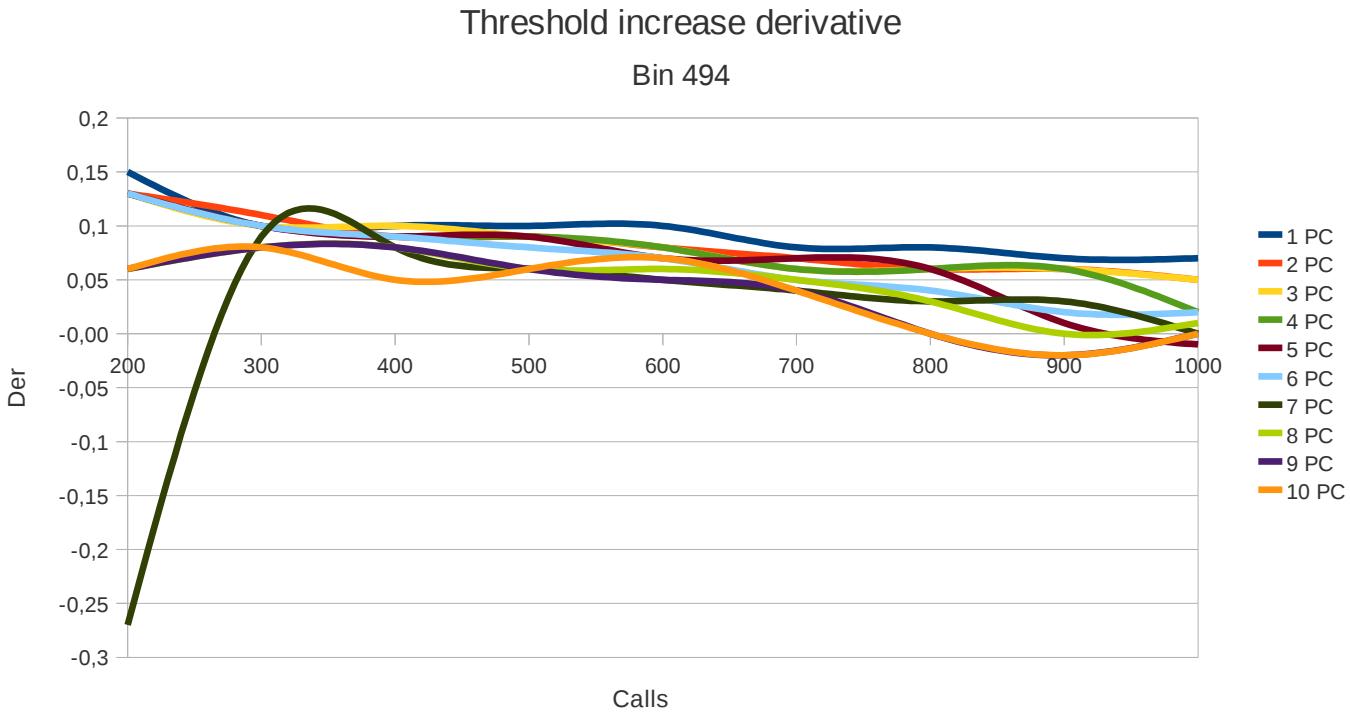


Figure 73



As in the previous chapters, the other types of considerations we can do about the thresholds regard the selection of the “best” threshold to select to obtain the best performances with a single analysis (one threshold and one number of PCs) using the software.
We used the two criteria introduced in the first chapter (number of anomalous users recognized upon the total number of anomalous users injected in the trace and number of anomalous time-bins detected upon the total number of anomalous time-bins present in the trace).
For each number of PCs and each threshold's value we computed these two ratios, obtaining results as the ones showed in Figure 74 (with 1 PC) and Figure 75 (with 10 PC).

Figure 74

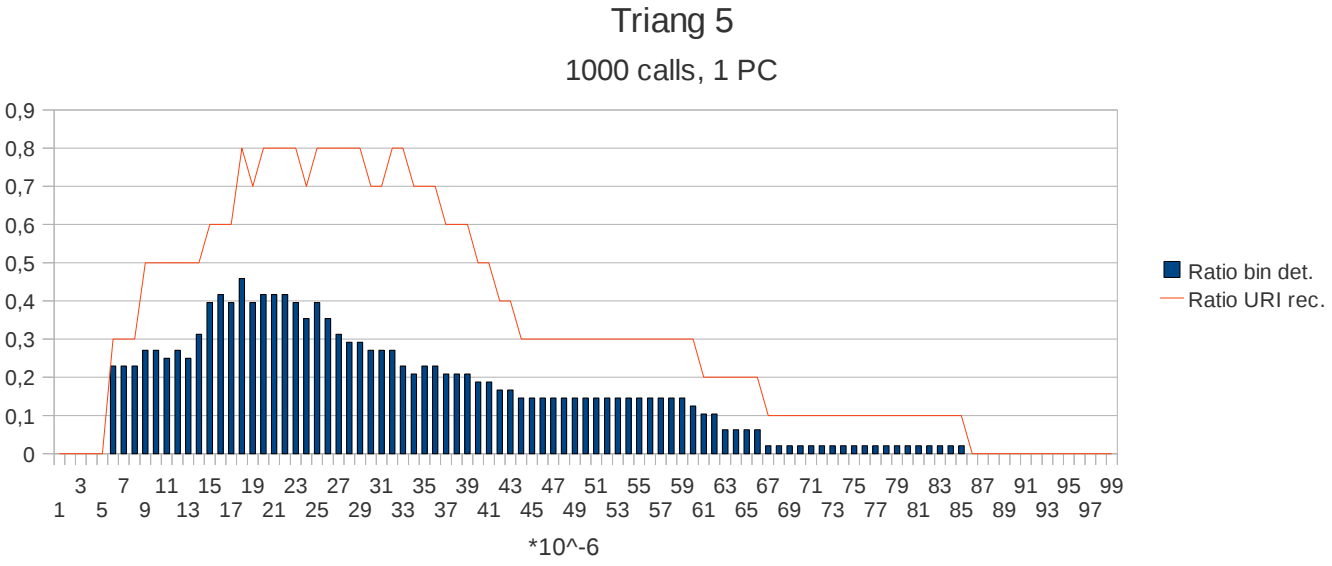
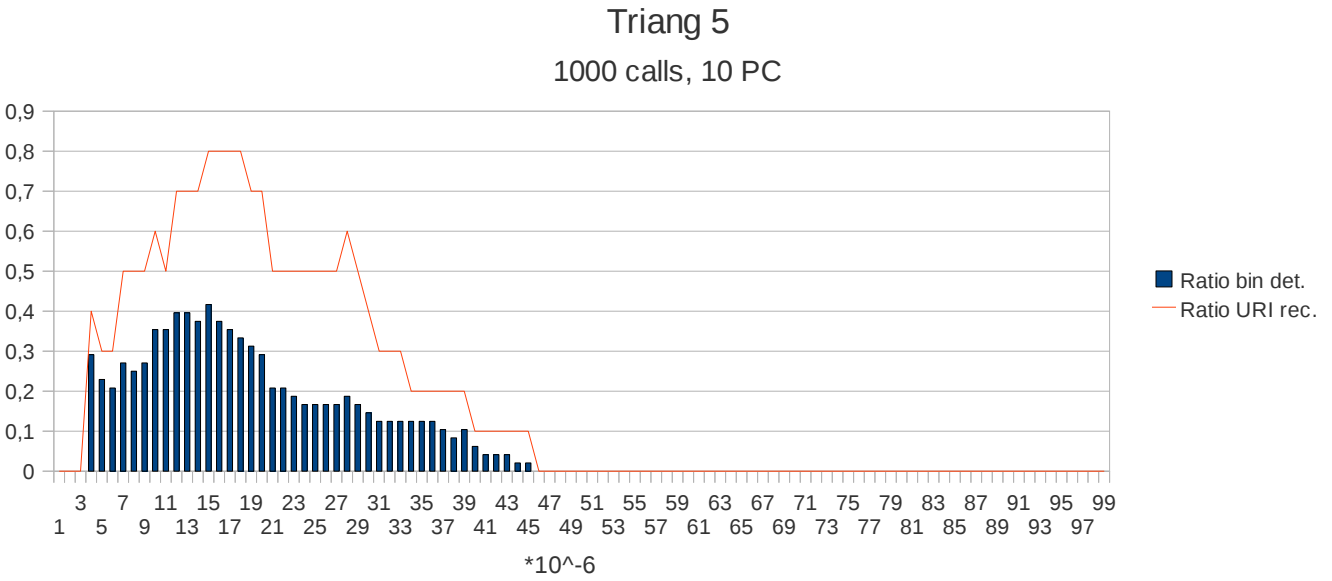


Figure 75



We can observe that there is no threshold that permits the identification of all 10 anomalous users: the maximum amount of users we can identify is 8 on 10; this is related to the fact that different users are detected with different sets of thresholds.
Compared to the results observed in the previous chapter (with rectangular anomalies of 10 time-bins), in this case we can see that the best performances are less dependent on the number of PCs (both with 1 or 10 PCs we can identify up to 8 anomalous users and around 40% of anomalous time-bins).
Like we did in the previous chapters we can see how the best threshold's values decrease with the number of PCs, as well as the range of thresholds that permits the best performances.

Triangular anomalies, width 10

The last type of anomalies injected in the trace were triangular anomalies with a duration of 10 consecutive time-bins; since the number of time-bins is even they are equivalent to triangular anomalies that are active for 9 time-bins (in the first time-bin the amount of anomalous calls is equal to zero).
Because of the longer duration (compared to the anomalies we discussed in the previous chapter) we have two cases of overlapping: anomaly in time-bins 243-252 with anomaly in time-bins 248-257, and anomaly in time-bins 490-499 with anomaly in time-bins 493-502.
We started the analysis with an anomaly's height of 5 calls and ended with 1000 calls; in Figure 76-77 we can see the amount of anomalous calls and the global calls' ratio per time-bin in the case of peaks of 1000 anomalous calls.

Figure 76

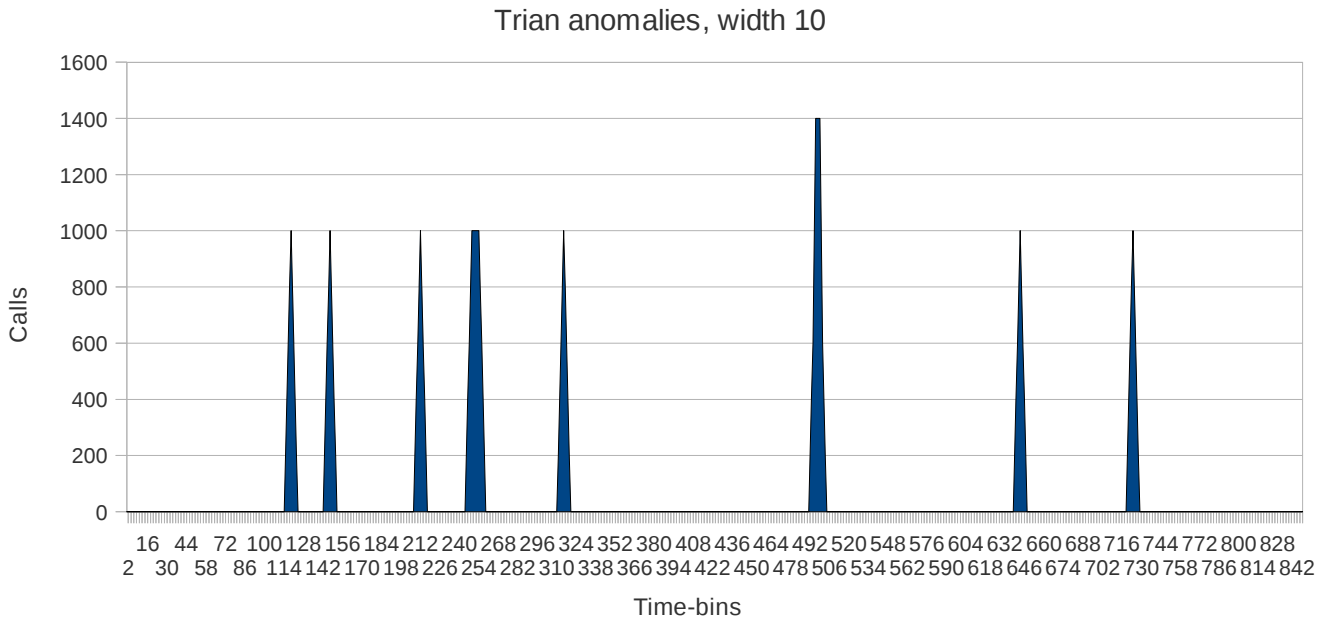
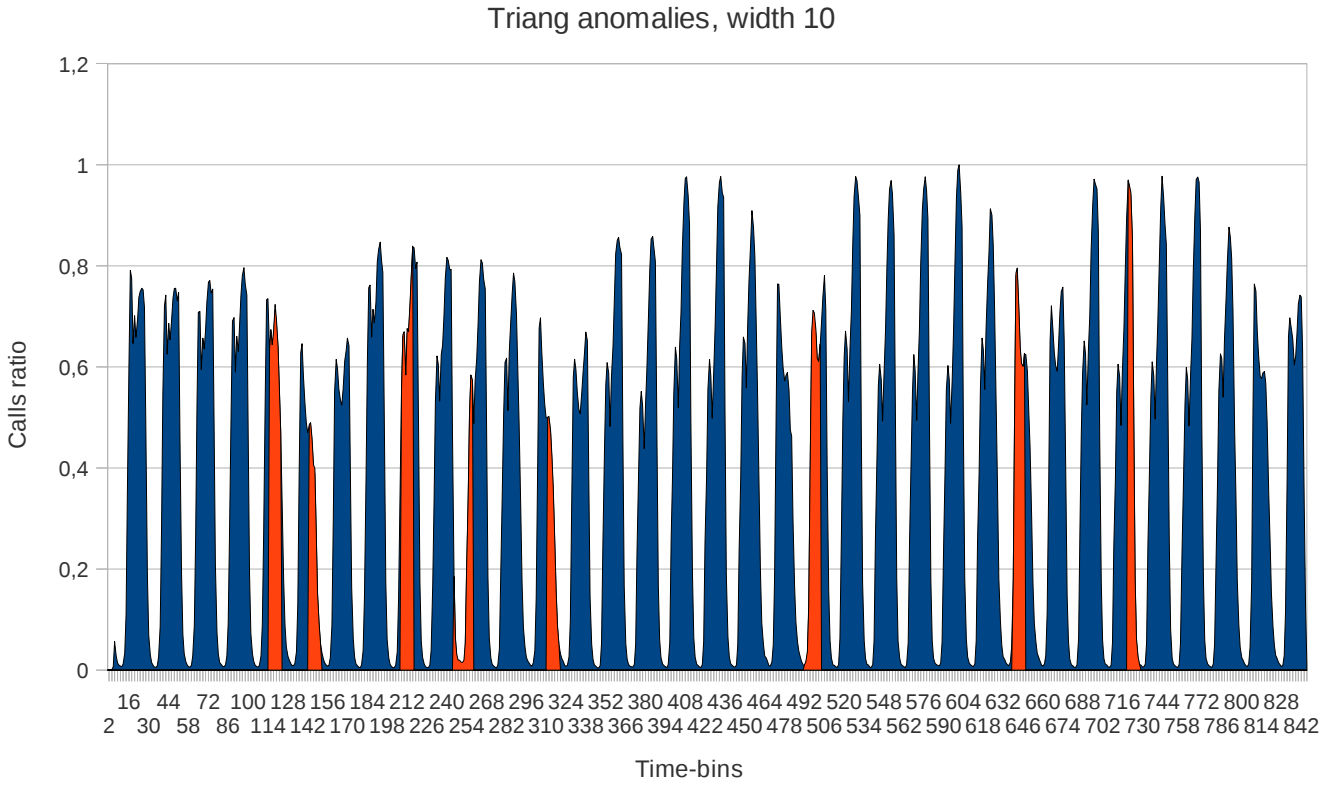


Figure 77



As usual, we start the analysis considering the order in which anomalous users are detected; the following list follows the same scheme we used in the previous chapters, and is referred to the order of detections starting from 80 calls (first time-bins detected) and ending with 700 calls (all anomalous users detected in at least one time-bin each):

1. Anomaly in time-bins 718-727 (17224 calls in the first time-bin detected: time-bin 724)
2. Anomaly in time-bins 490-499 (11715 calls: time-bin 494)
3. Anomaly in time-bins 141-150 (14575 calls: time-bin 148)
4. Anomaly in time-bin 248-257 (4320 calls: time-bin 252), with the first time-bin detected that is part of the overlapping zone
5. Anomaly in time-bins 309-318 (5029 calls: time-bin 318)
6. Anomaly in time-bins 243-252 (888 calls: time-bin 248), with the first time-bin detected that is part of the overlapping zone
7. Anomaly in time-bins 637-646 (69803 calls: time-bin 642)
8. Anomaly in time-bins 206-215 (65691 calls: time-bin 211)
9. Anomaly in time-bins 493-502 (68371 calls: time-bin 498), with the first time-bin detected that is part of the overlapping zone
10. Anomaly in time-bins 113-122 (70409 calls: time bin 118)

We can see the amount of detections per time-bin, together with the ratio of global calls per time-bin, in the cases of 80 and 1000 anomalous calls in Figure 78 and 79.

Figure 78

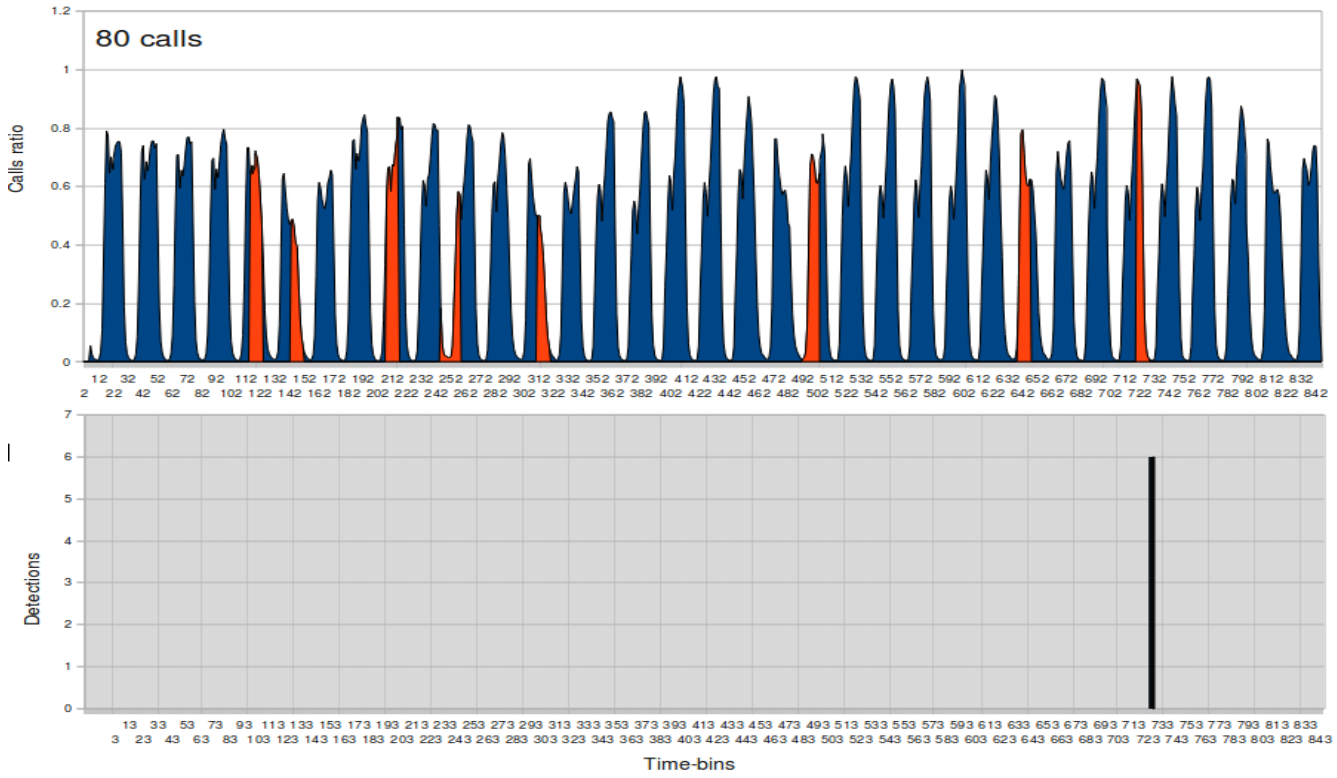
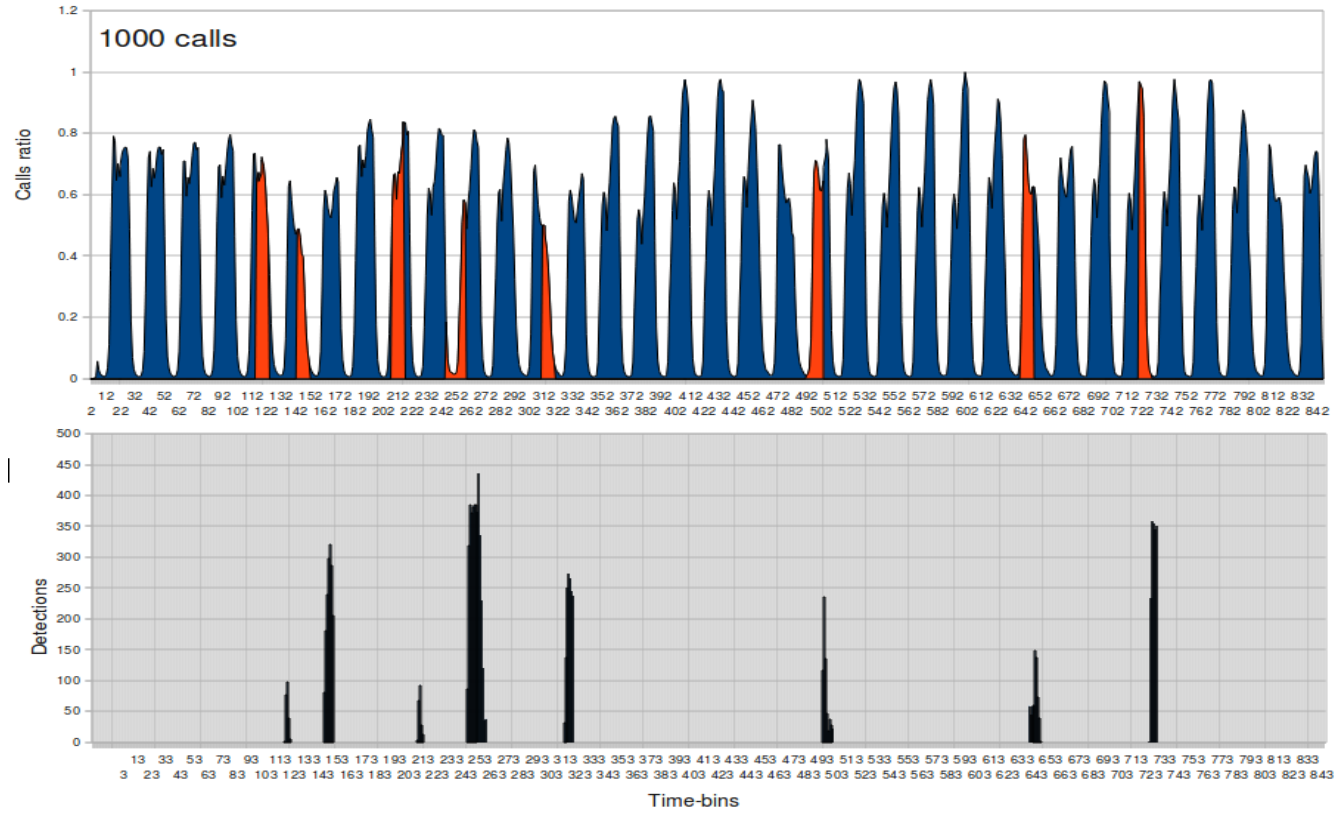
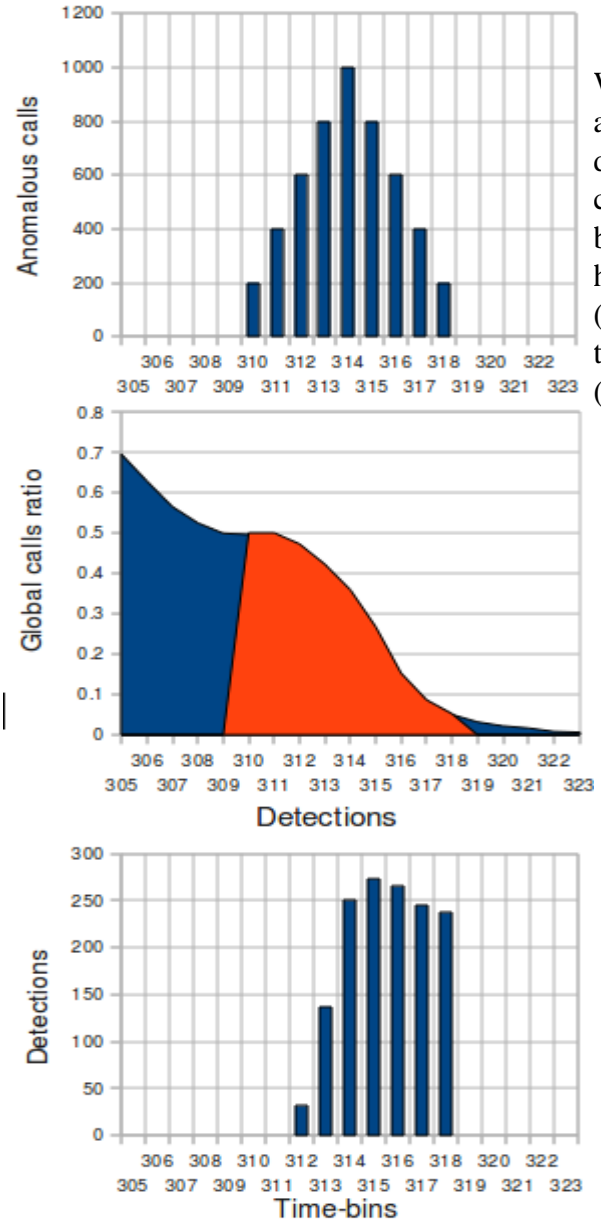


Figure 79



Even in this case we can observe how the software's capability of detecting each time-bin is a combination of two factors: the amount of anomalous calls in that time-bin (the anomaly shape, with a tendency to better detect central time-bins, where the amount of calls is higher) and the amount of global calls in the time-bin (the less the better). Compared to the previous case (triangular anomalies of 5 time-bins) the time-bin position inside the anomaly has less influence, since the higher duration of each anomaly means that there is a lower difference in the amount of calls in two consecutive time-bins, so the amount of global calls have a higher influence than in the case of narrower triangular anomalies.
We can see this by focusing on the number of detections per time-bin inside a single anomaly, with an anomaly's height of 1000 calls. We did this for the anomaly in time-bins 638-642, and in Figure 80 we plotted the number of anomalous calls per time-bin, the ratio of global calls per time-bin and the number of detections per time-bin, for that anomaly.

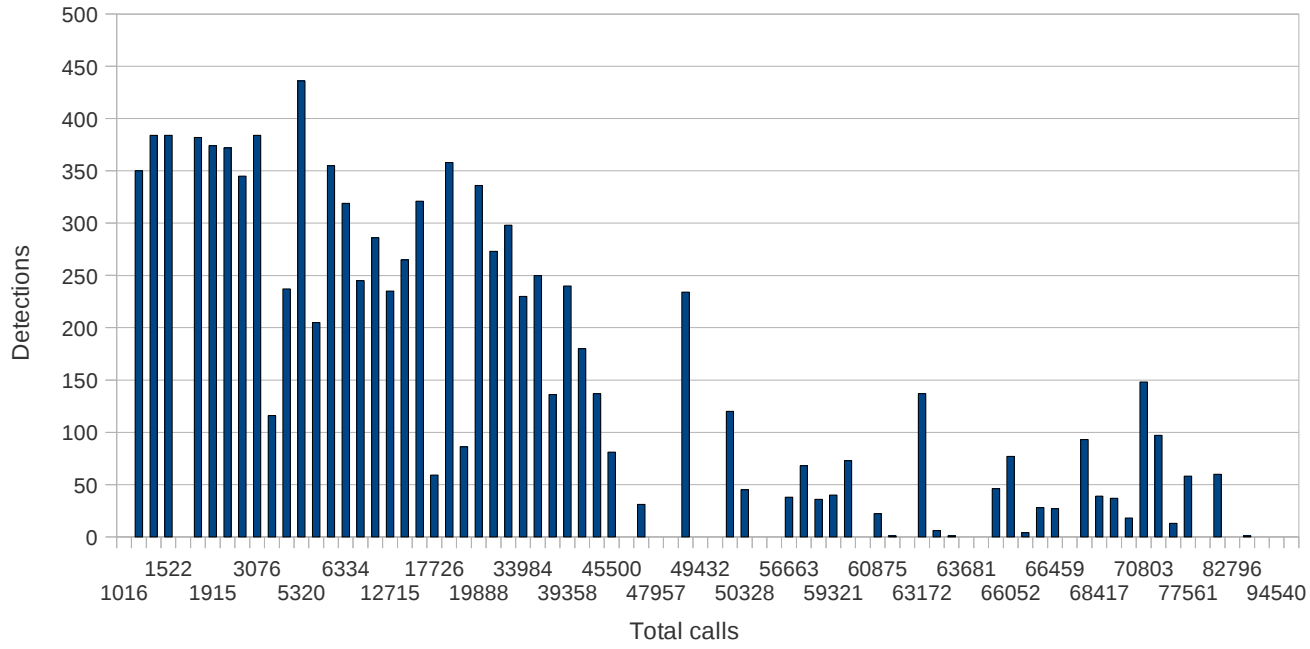
Figure 80



We can see the double dependence (from the number of anomalous calls and global calls) of the number of detections, but in this case (compared to the previous chapter) the number of global calls seems to have a bigger impact, as we can observe in Figure 80, where the higher amount of detections is not in the central time-bin (where there is the top number of anomalous calls) but in time-bin 315, where the amount of global calls is lower (even if there are 800 anomalous calls instead of 1000).

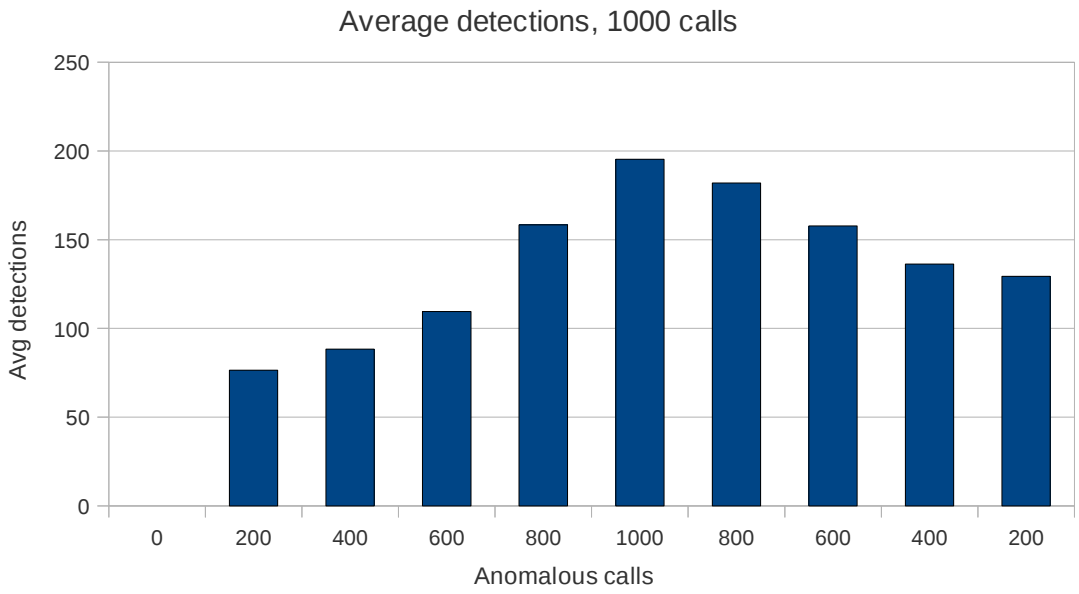
To verify the tendency of better detecting time-bins with less anomalous calls we can plot the number of times each anomalous time-bin was detected (we didn't included not-anomalous time-bins, cause they were never been detected) versus the amount of global calls in that time-bin, ordering the results by the number of global calls. We did this in Figure 81, with an anomaly's height of 1000 calls; we can observe that the tendency is still present, and is heavier than in the case of triangular anomalies on 5 time-bins.

Figure 81



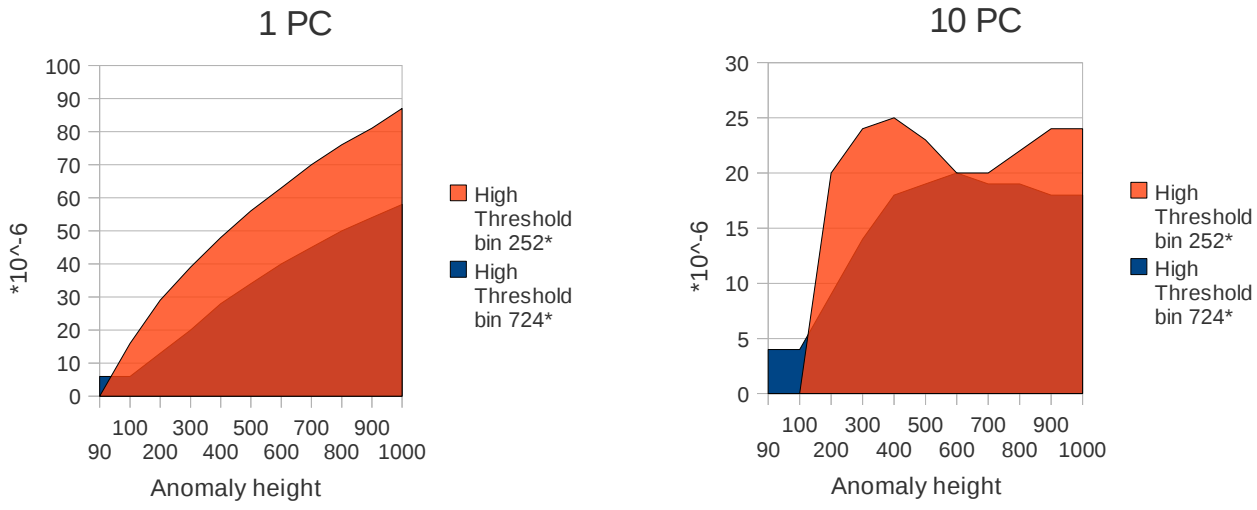
But is still present also the tendency to better detect time-bins with a higher amount of anomalous calls, and we can see this from Figure 82, where we plotted the average amount of detections for each type of time-bin (first time-bin, second, third, fourth, etc...), with an anomaly's height of 1000 calls (correspondent to the amount of anomalous calls in the sixth time-bins): on the x-axis we wrote the amount of anomalous calls in each type of time-bin and on the y-axis we have the mean numbers of detections, computed along the 10 anomalies inserted in the trace.

Figure 82



As with the other types of anomalies the second topic on which we focused is the sensitivity of the threshold to the anomaly's height and the number of Principal Components.
We first observed the thresholds range that permits the detection of a certain anomalous time-bin, as depending from the anomaly's size, varying the number of Principal Components used; we did this observation for two different time-bins: time bin 252 (the one with most detections with 1000 calls) and time bin 724 (the first one detected, with 100 calls).
In Figure 83 we plotted the maximum thresholds that permitted the identification of the anomalous users in time-bins 252 and 724 against the anomaly's height, for 1 and 10 Principal Components used.

Figure 83



The considerations we can do on these graphs are the same we did in the previous chapter, except to the fact that the behavior of the thresholds relative to time-bin 252 are different from any other type of behavior we have analyzed before: the high thresholds have two maximum points, one with an anomaly's height of 400 calls and another one (lower than the previous one) with 900. A comparison of this behavior with the ones relative to other anomalous time-bins showed that this phenomenon happened only for time-bin 252, so that could be related to the fact that this particular time-bin is part of an overlapping zone. Further investigations showed that this is the actual reason, since the same time-bin led to the detection of two different anomalies for different anomaly's heights: a lower anomaly's height's value led to the detection of the anomaly in time-bins 248-257; with an anomaly's height of 400 calls we had the peak of detections for that specific anomaly; for higher anomaly's height that anomaly wasn't detected anymore, but another anomaly (the one in time-bins 243-252) started to be pointed out; higher values of the anomaly's height led to improvements in the detections of the latter anomaly.
Except to this issue, related to the overlapping-problem, the thresholds behavior is the same discussed in the previous chapters.
The thresholds behavior related to time-bin 252 is worthy of further attention, so in Figure 84 and 85 we plotted the high thresholds and the threshold's increase derivative against the anomaly height for each number of PCs, for that particular time-bin.
We can observe how the “moment” in which we switch from the detection of one anomaly to the other is different using different numbers of PCs (for example with 1 PCs we continue to detect the anomaly in time-bins 248-257 even with an anomaly's height of 1000 calls); from further analysis we can observe that it could also happen that, with an analysis conducted with a fixed number of PCs, some thresholds lead to the detection of one anomaly while others lead to the detection of the other.

Figure 84

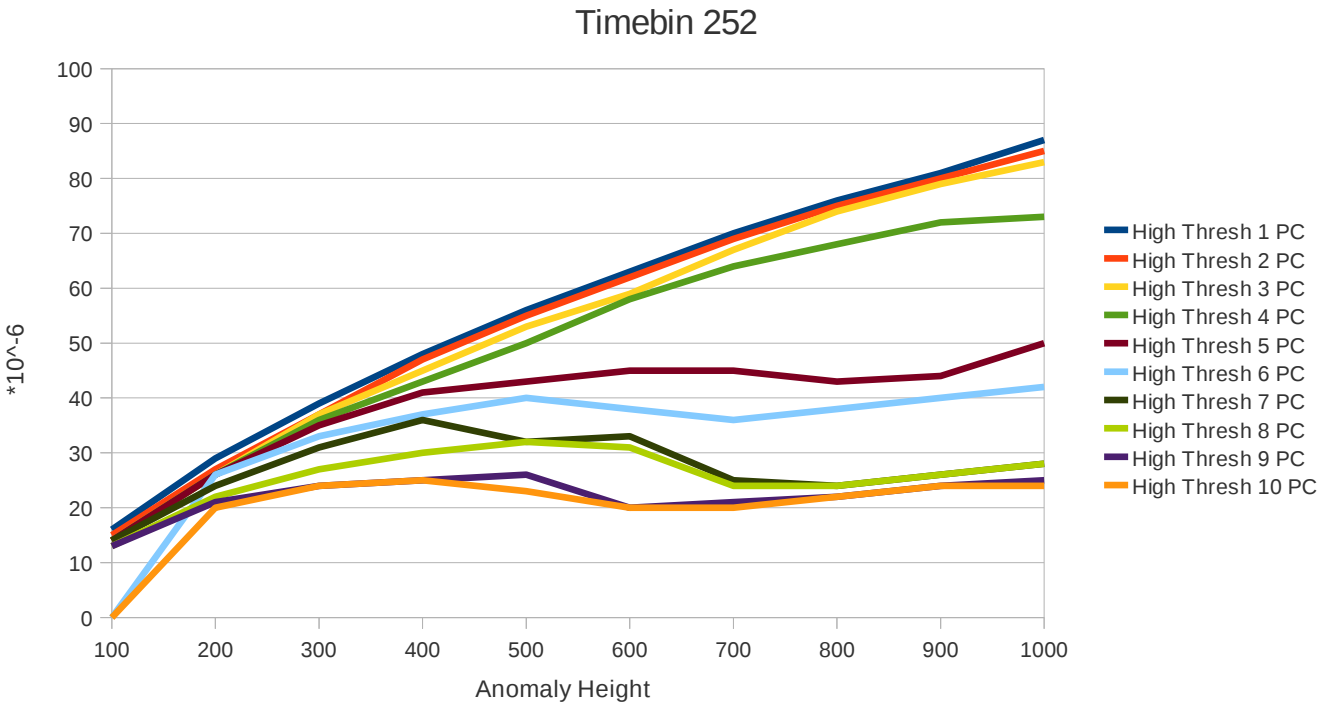
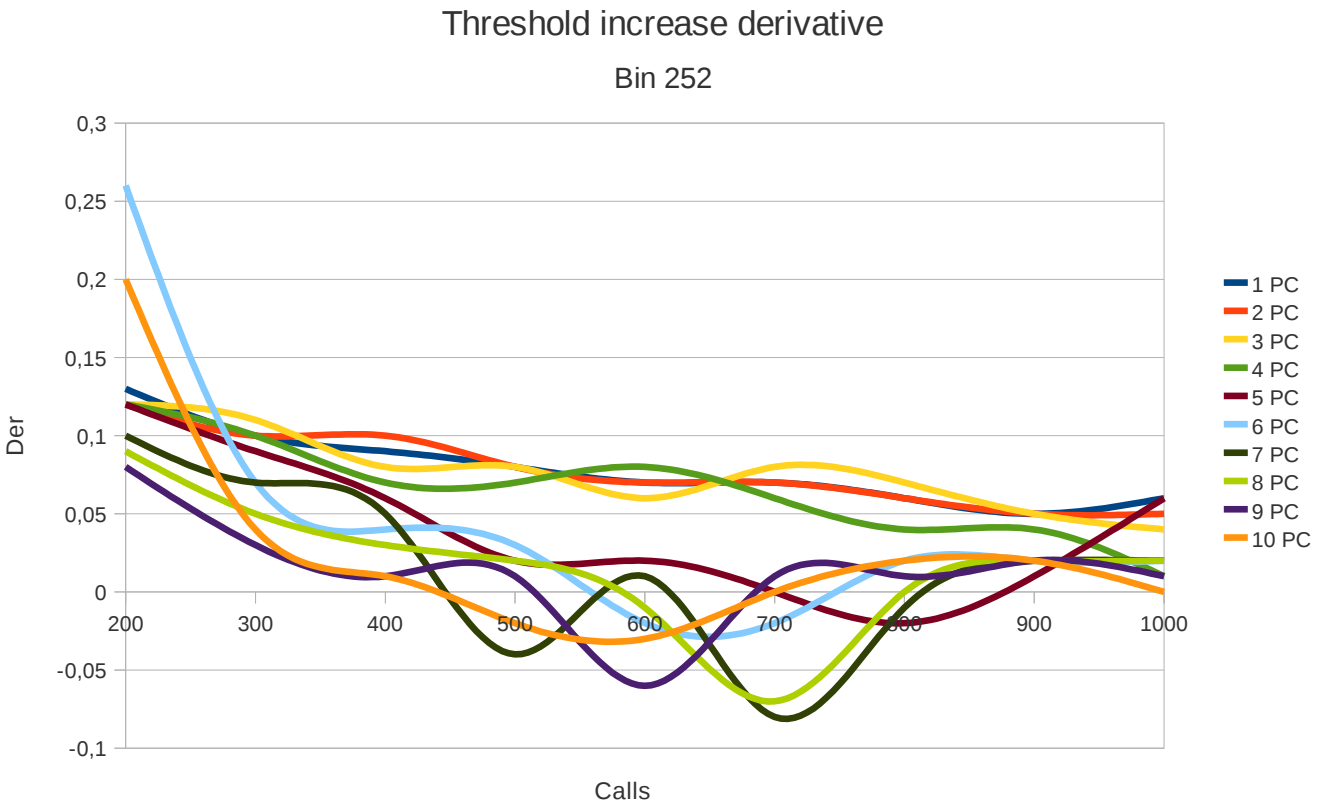


Figure 85



As in the previous chapters, the last considerations we can do about the thresholds regard the selection of the "best" threshold to be selected to obtain the best performances with a single analysis (one threshold and one number of PCs) using the software.
We used the two criteria introduced in the first chapter (number of anomalous users recognized upon the total number of anomalous users injected in the trace and number of anomalous time-bins detected upon the total number of anomalous time-bins present in the trace).
For each number of PCs and each threshold's value we computed these two ratios, obtaining results as the ones showed in Figure 86 (with 1 PC) and Figure 87 (with 10 PC).

Figure 86

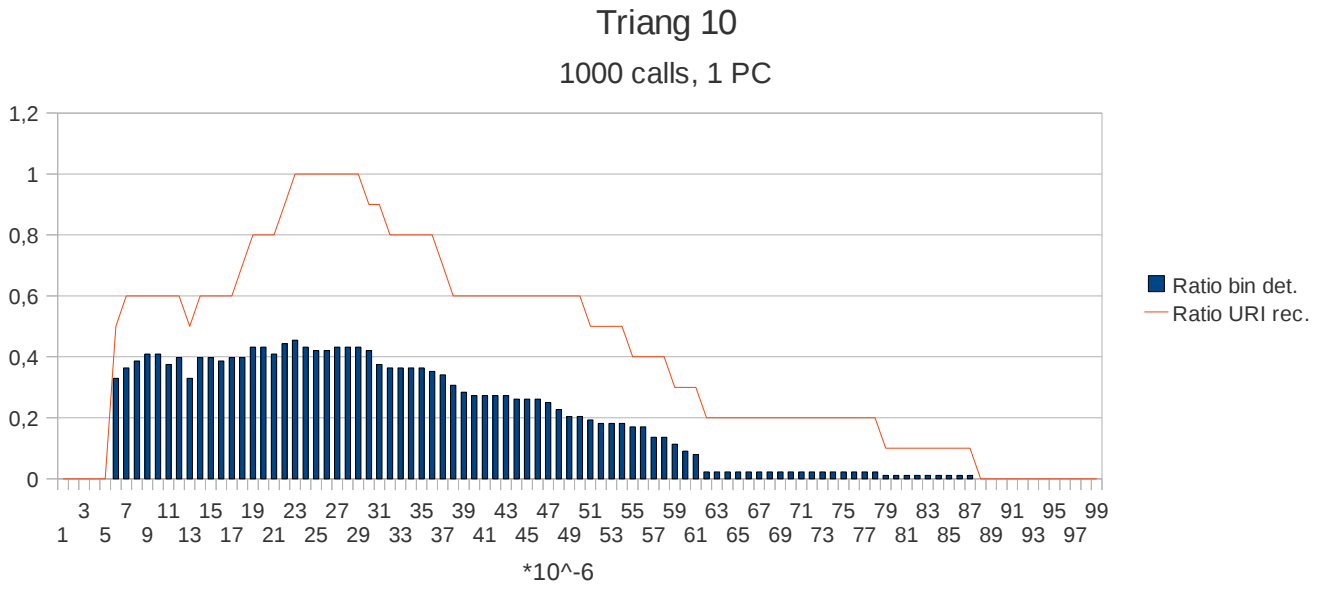
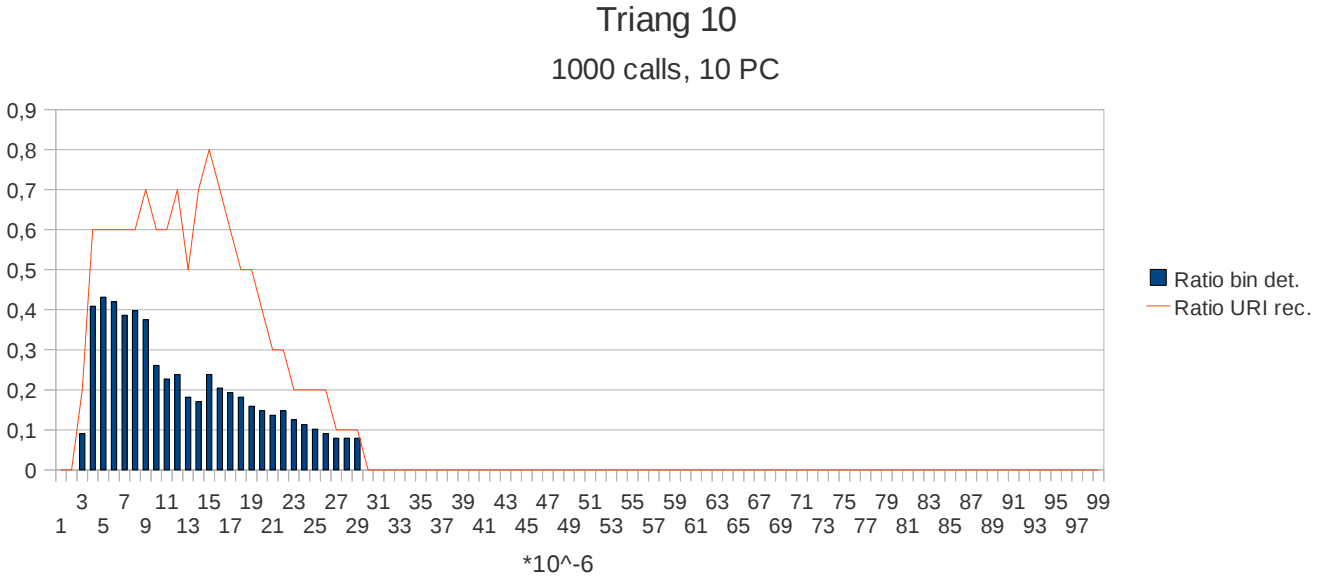


Figure 87



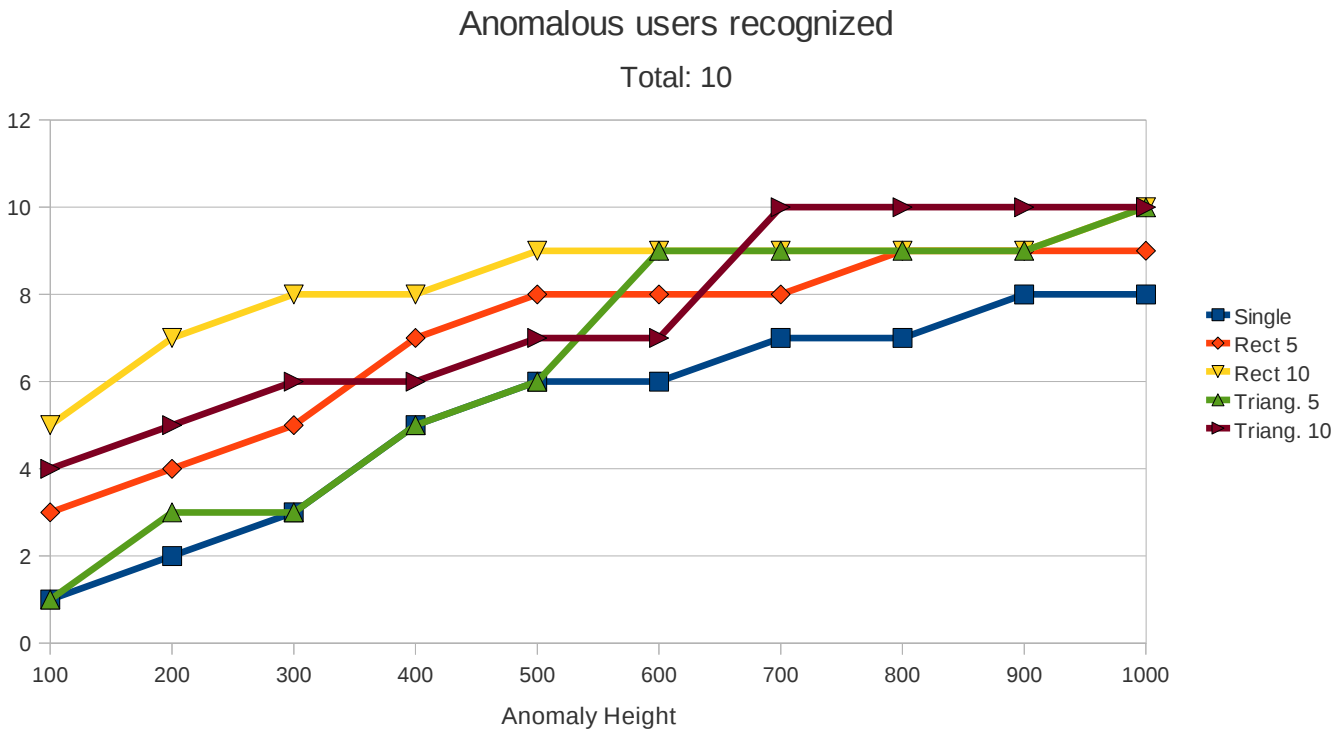
Except for the same considerations made in the previous chapters it is worthy of notice that only in this case we are able to perform a “single threshold – single number of PCs” analysis that can lead to the identification of all 10 anomalous users, since there is a range of thresholds (using 1 to 6 PCs) that leads to the detection of all anomalous users.
We can also notice that the software's best performances in detecting anomalous users are more dependent to the number of PCs (with 10 PCs we can only identify up to 8 users on 10) than the performances in detecting anomalous time-bins, that are around 40% both with 1 or 10 PCs.

6.7 *Results comparison*

In this chapter we compared the results obtained by analyzing the different types of artificial anomalies we took into consideration, highlighting common trends and differences in the software's capability of detecting each type of anomaly.

The first type of comparison we did is about the software's capability of detecting as many anomalous users as possible, as depending from the anomaly's height.
In Figure 88 we plotted the amount of anomalous users identified (at least in one time-bin with one threshold and one number of PCs) in a full single-round analysis (all thresholds and all number of PCs), for each value of anomaly's height (from 100 to 1000), for each type of anomaly considered.

Figure 88



We can make several observations on this graph:

- Anomalies with a lower width tend to be detected in a lower number, for each value of the anomaly's height; we can see this comparing the number of anomalous users recognized with rectangular anomalies of 5 and 10 time-bins (this second type is always detected in a higher number), triangular anomalies of 5 and 10 time-bins (there is only an exception, with an anomaly's height of 600 calls) or single spike anomalies with any other type.
- Rectangular anomalies tend to be detected in a higher number with low anomaly's height values, as we can see comparing the results from rectangular and triangular anomalies with the same

width.

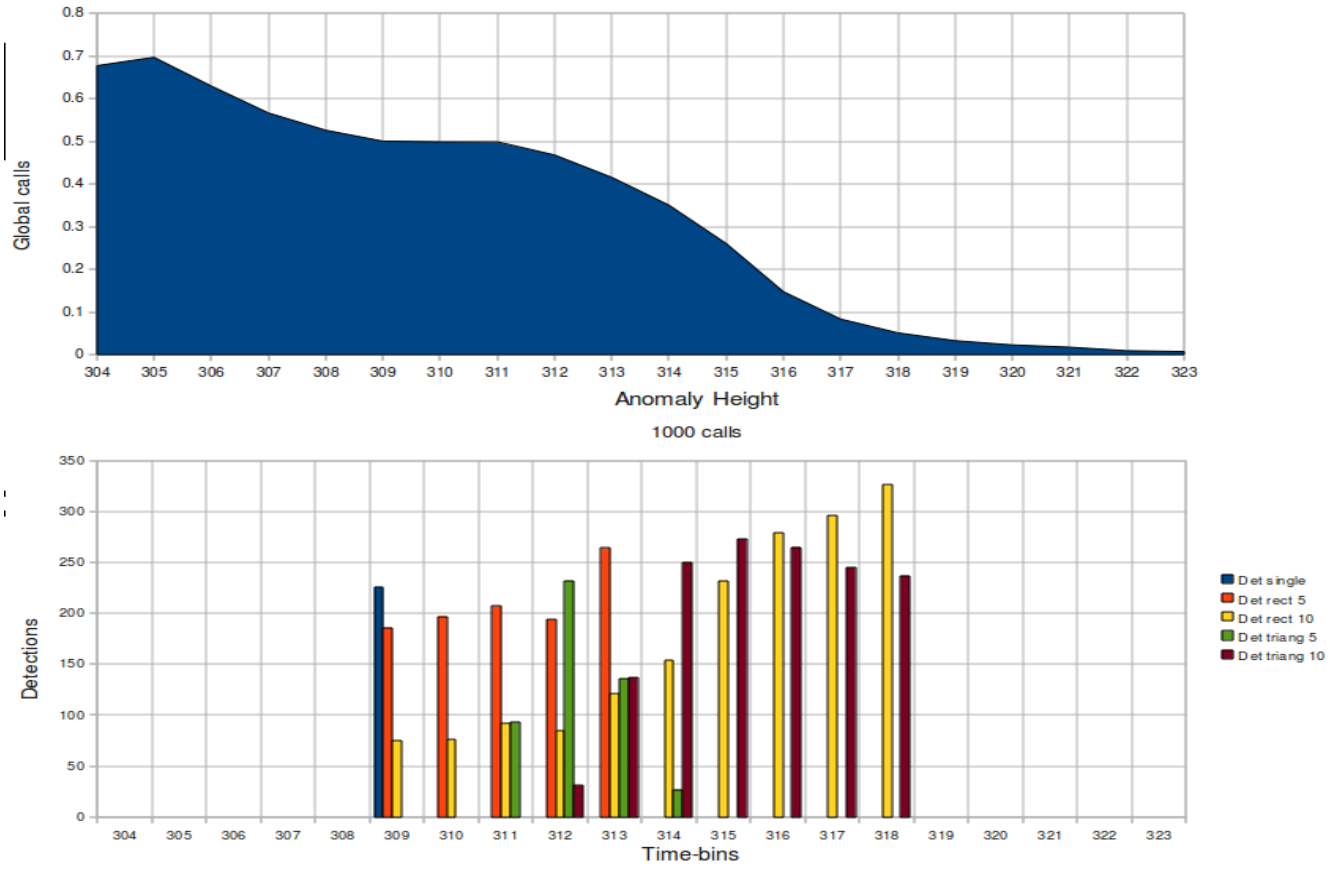
- On the other side the amount of anomalous users the software is able to detect in the case of triangular anomalies grows faster, than in the case of rectangular anomalies, with the anomaly's height (for example we can observe how triangular anomalies of 10 time-bins are the only one that permit the detection of all of them with an anomaly's height of 700 calls).

To analyze in better detail the software's performances in detecting each anomalous time-bin of an anomaly, we decided to focus on a single anomalous user, taking into consideration the number of times each anomalous time-bin was detected in a full analysis (100 thresholds and 10 number of PCs), comparing these results for the different types of artificial anomalies. In Figure 89 we plotted the amount of detections per time-bin, in a full analysis, compared to the ratio of global calls per time-bin, for each type of anomaly we analyzed.
From the graphs we can observe the two kind of tendencies we discovered in the previous chapters:

- The tendency to better detect time-bins with a lower amount of global calls.
- The tendency to follow the anomaly's shape, that means more detections for the central time-bins for triangular anomalies (more anomalous calls) and more detections in border time-bins for rectangular anomalies (the software detects the discontinuity).

The number of detections for each time-bin is a trade-off of these two tendencies, with more focus on the first one for wider anomalies and more on the second one for narrower anomalies (in particular we observed this tendency for triangular anomalies).

Figure 89



We then started to focus on “single threshold – single number of PCs” analysis, comparing the software's performances for this type of analysis, for each type of anomalies inserted in the trace.
We started considering the same two parameters we introduced in the previous chapters: the ratio of anomalous time-bins correctly detected (upon the total of them) and the ratio of anomalous users recognized, in at least one time-bin (upon the total of them). To compare the results obtained with different kinds of anomalies we took into consideration the top values of this ratios (that we can obtain selecting the appropriate threshold) for each number of Principal Components used, obtained using an anomaly's height of 1000 calls. In Figure 90 we can see these results for the ratio of anomalous time-bin detected and in Figure 91 for the ratio of anomalous users detected.

Figure 90

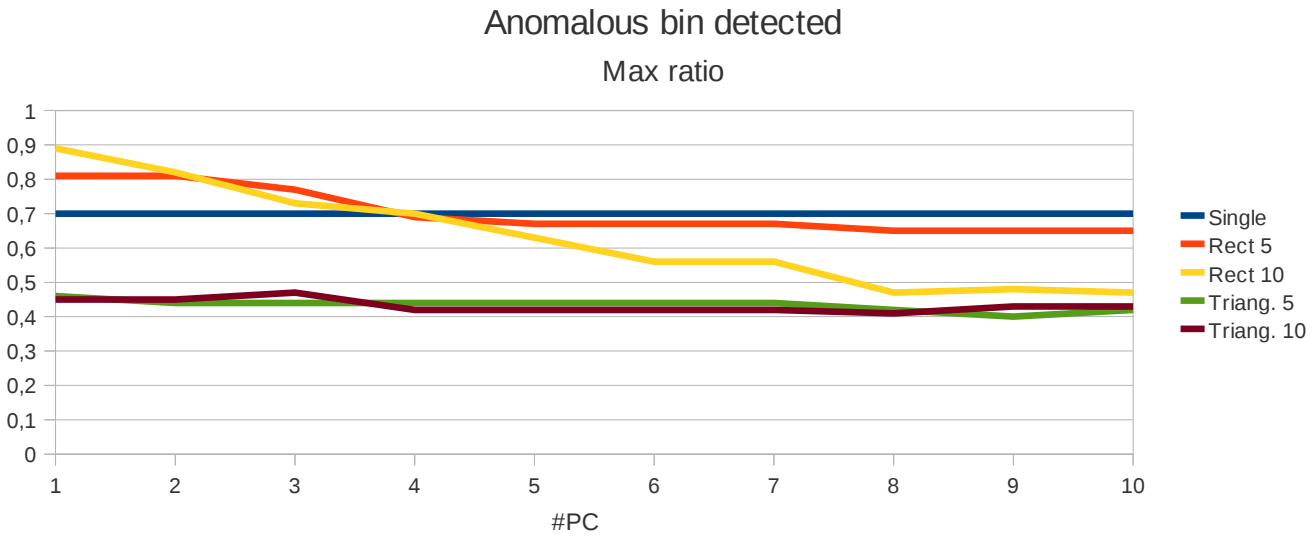
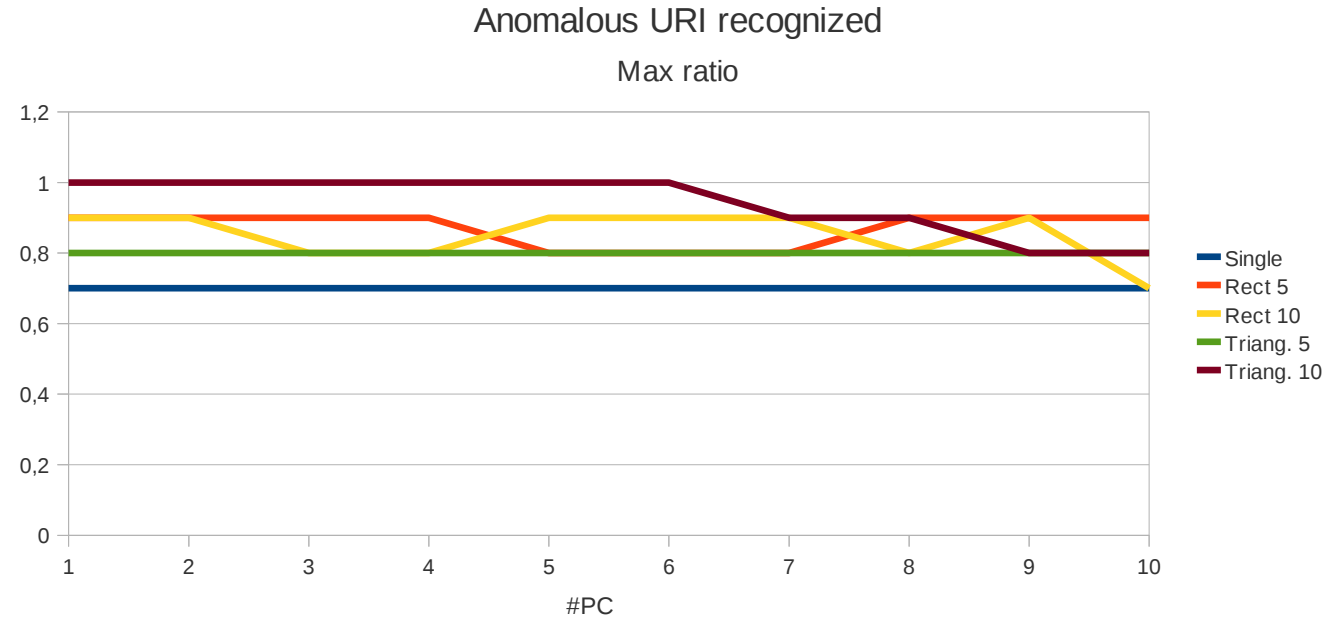


Figure 91



About the ratio of anomalous time-bins correctly detected we can do the following considerations:

- Rectangular anomalies permit better results, since are the only one that permit the detection of up to 90% of them.
- The software's performances with rectangular anomalies are more dependent on the number of PCs used, than triangular or single spike anomalies (the performances of which seem to be stable with the number of PCs); that is because rectangular anomalies inject a higher total amount of anomalous calls in the trace (they have a bigger volume than triangular or single spike), so they are more likely to be included in the *normal* subspace with higher numbers of PCs (we have a confirmation of this theory observing that the dependence on the number of PCs is less accentuated for rectangular anomalies of 5 time-bins, compared to the same type o anomalies of 10 time-bins, since their total volume of calls is smaller).
- In general having wider anomalies seems not to lead to an improvement in the software's capability of detecting a higher amount of anomalous time-bins (but we should also consider that wider anomalies have a higher amount of anomalous time-bins, so with the same ratio we have more detections).

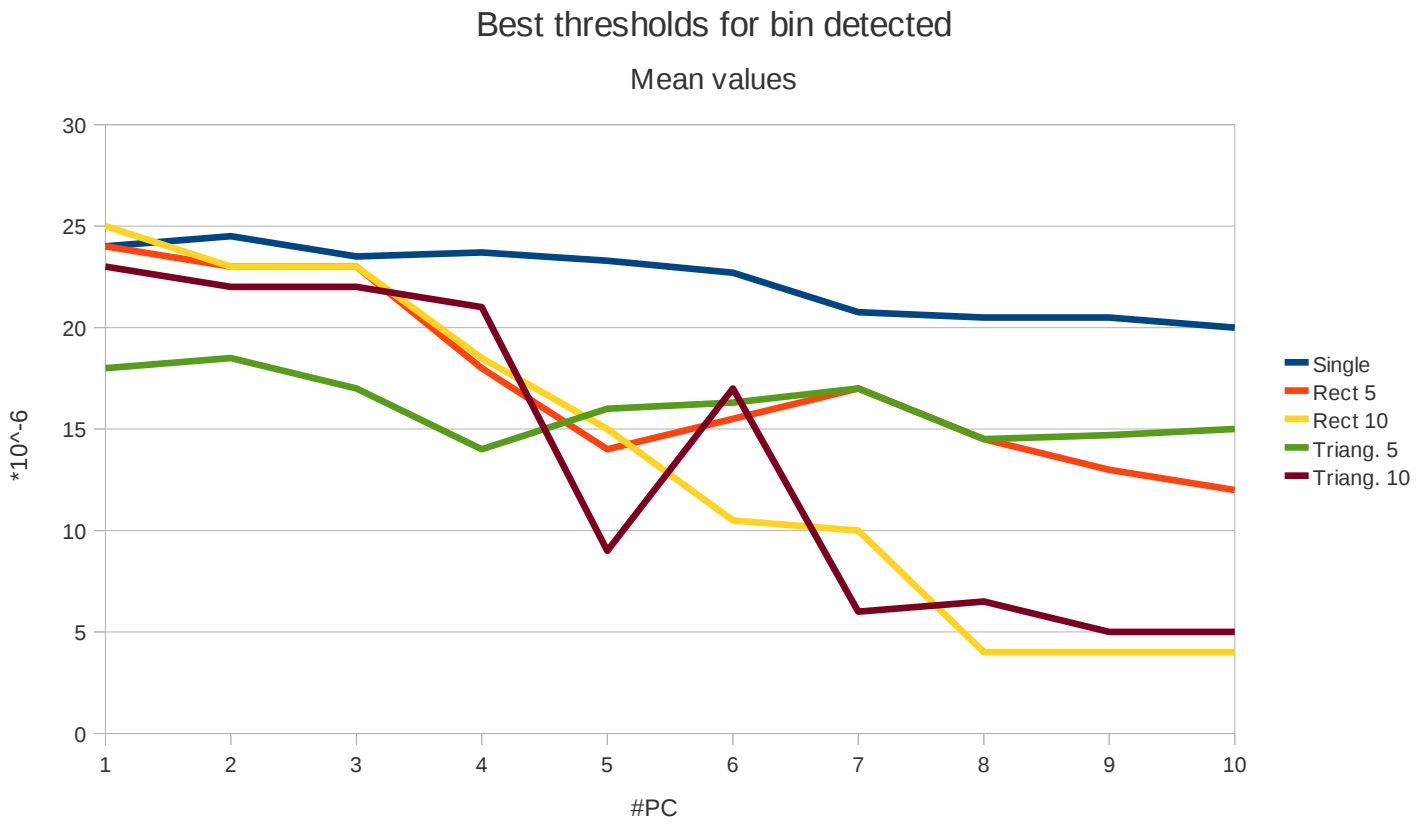
About the ratio of anomalous users correctly recognized we can observe, as we already did in the previous chapters, that the only type of anomalies that permits the identification of all 10 anomalous users (with an anomaly's height of 1000 calls) is triangular anomalies of 10 time-bins. There is a small improvement in the software's capability of detecting different users in at least one time-bin when increasing the anomaly's width, but this is normal because of the fact that wider anomalies give more chances of having at least one time-bin in which each anomaly is detected.

We then focused on the threshold's values that permitted to obtain those maximum ratio values we just analyzed, as depending from the number of PCs used and the type of anomaly.
In Table 4 we inserted the “best threshold” values, in the meaning of maximum ratio of anomalous time-bins detected, for each kind of anomaly we used and each number of PCs (the values should be multiplied by 10^-6).
In Figure 92 we took the mean values of these ranges of thresholds and we plotted them against the number of PCs for each type of anomaly.

Table 4

PC number	Single	Rect 5	Rect 10	Triang. 5	Triang. 10
1	22-26	22-26	25	18	23
2	23-26	23	23	16,21	22
3	22-25	23	23	14,20	22
4	22,24-25	18	18-19	14	21
5	21,24-25	14	15	14,18	9
6	21,23-24	14,17	10-11	14,17-18	17
7	19-21,23	17	10	17	6
8	19-22	13-16	4	13,16	5-8
9	19-22	13	4	12,15,17	5
10	19-21	12	4	15	5

Figure 92

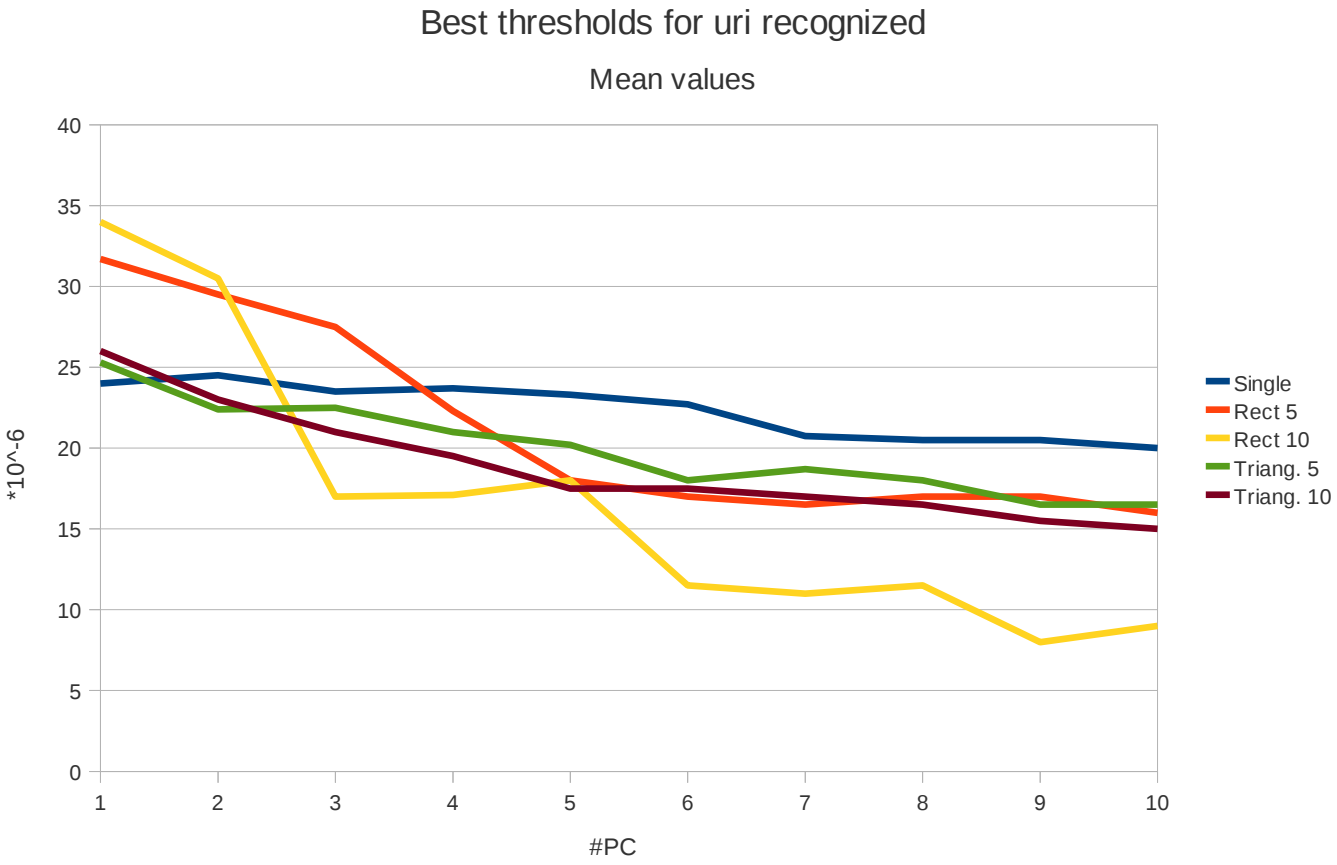


We can see that the best threshold's values tend to decrease with the number of PCs (we already noted this phenomenon) and that they are different for different kinds of anomalies. That means it's difficult to choose an appropriate single threshold's value, when launching an analysis without knowing the type of anomalies we want to detect; as we have already said it's always suggested to use a range of thresholds (as wider as possible) when using this software in AD applications.
To confirm this tendency we performed the same comparison for the other type of "best" thresholds: the ones that permit the identification of the highest ratio of anomalous users possible. In Table 5 we inserted these values and in Figure 93 we plotted the mean values against the number of PCs, for each type of anomaly.

Table 5

PC number	*Single*	*Rect 5*	*Rect 10*	*Triang. 5*	*Triang. 10*
1	22-26	22-26,28-41	25-43	18,20-23,25-29,32-33	23-29
2	23-26	23-36	26-35	20-22,24-25	22-24
3	22-25	22-33	12-22	20-21,24-25	20-22
4	22,24-25	21-22,24	11-22,24	20,22	18-21
5	21,24-25	14-22	18	17-20,23-24	17-18
6	21,23-24	13-21	11-12	17-19	17-18
7	19-21,23	13-20	11	17-18,21	16-18
8	19-22	17	10-13	16-20	16-17
9	19-22	17	8	15-18	15-16
10	19-21	16	7-11	15-18	15

Figure 93



Upon this results we can do the same observations we did before, noting how the “best” threshold's values tend to decrease with the increasing of the number of PCs and how there isn't an a-priori value that we can choose when launching an analysis on a VoIP trace. In this case, for example, we should use a range of a least 20 thresholds (for an analysis with a single number of PC) or at least 30 when launching an analysis with multiple numbers of PCs, to be sure to use the “best” threshold's values for each kind of anomaly.

7. Conclusions

In this work we analyzed the possibility of applying the AD software based on Principal Components Analysis developed in Pisa on a real VoIP trace, collecting records of the number of attempted calls in every hour of a month, by each user of a telephonic operator.
We first performed a single round analysis, trying to tune each value of the parameters needed by the software: the number of hash functions, the number of PCs and the threshold's value.
We decided to set the number of hashing functions required to consider the identification procedure successful equal to 5 (on a total of 8 hash functions used), because using a lower value led to a too high percentage of false alarms in the list of anomalous users identified.
We then decided to launch the analysis using threshold's values varying from 0.000001 (the minimum value accepted by the software) to 0.0001 (because using higher values led to no other users detected) and number of PCs varying from 1 to 10.
From the single-round analysis we discovered that the type of anomalous users the software is able to detect varies using different numbers of PCs, focusing more on volume anomalies with a lower number of PCs and more on behavioral anomalies using a higher number of PCs.
We also noted that using a single threshold's value is not the best option when using this software, since different anomalies can be detected with different ranges of thresholds, so if we are interested in finding the higher amount of anomalous users possible, we should use wide ranges of thresholds, instead than one single values.
A big limitation in the possibility of detecting high numbers of anomalous users is caused by the fact that this type of software is only able to identify up to one single anomalous user per time-bin, since it first searches for anomalous time-bins in the trace, and then, for each of them, attempts to identify the user responsible for the anomaly detected.
To avoid this problem we decided to perform a multiple-round analysis, in which we removed, after every round, the most anomalous user detected. As we thought this analysis led to better performances in the number of anomalous users identified; however we have to consider that this type of analysis has just theoretical importance, since it's computational complexity is too higher (than the single round analysis) to use it in AD applications.
The results obtained with multiple-round analysis led us to some more observations, regarding the modifications in software's performances when the trace starts to be cleaned up from anomalies. We discovered that the number of Principal Components needed to correctly detect and identify behavioral anomalies tends to become smaller when the trace is cleaned up from the anomalous users; the software's performances are boosted up by the removal of highest volume anomalies in the trace (since they are the most likely to influence the algorithm's ability to model the *normal* behavior from the trace), and that tends to happen more with low numbers of PCs than with high numbers of them (as we said a low number of PCs is better in identifying volume anomalies). Because of this progressive performances improvement (each round the most anomalous users are removed, so, if they are high volume anomalies, the software's ability in modeling the *normal* behavior increases) we obtain the best results for low numbers of PCs, managing to detect a bigger number of anomalous users than with higher numbers of PCs.
In order to better inspect the software's performances in detecting anomalies on a clean trace, the last

type of analysis performed regarded artificial anomalies inserted on the clean trace obtained at the end of the multiple-round analysis. We analyzed the software's performances using different types of anomalies (single spike, rectangular and triangular), with different volumes (different height, that means number of anomalous calls per time-bin, and different width, that means number of time-bins that composed an anomaly).
From this type of analysis we observed how the software's performances in detecting this type of anomalies are dependent both on the anomaly shape and to the positions in the trace where we insert these anomalies. In general the software is most likely to detect anomalies that happen in time-bins with a low number of global calls (this is expected, since the anomalies tend to be more “visible” in those types of zones); it's also most likely to detect anomalies in their peak points (for anomalies in which the number of calls varies among the time-bins) or in their discontinuity points (for anomalies that have the same amount of calls over consecutive time-bins).
The software's capability of detecting each anomalous user is dependent on the type of anomalies present in the trace and on the anomaly volume; we are able to identify all of them only in some cases. Even if we are able to identify all the anomalous users it's of course more difficult to detect every time-bin that composes each anomaly. It's important to note that the best performances (in these senses) are obtained with very specific values of the threshold, that vary with the type and the volume of the anomalies inserted, and often with low number of Principal Components. That leads us to the consideration that is always better, when using this software for AD applications, using a range of thresholds instead than a single value.
The number (and volume) of the anomalies present in the trace is a key-factor in the software's performances, since anomalies with a high volume can contaminate the *normal* subspace created with the Principal Components used in the analysis, leading to a worsening of software's performances.
Even if we didn't analyze this phenomenon in detail we must also notice that the parameter's choice is sensible to the trace's size: before focusing on the trace we discussed we tried to apply the software for analyzing traces correspondent to a lower number of call hours, noticing that the threshold's values that permitted the best performances in detecting anomalous users were considerably different.

For concluding, we can answer to the question “Is this software a valuable method for Anomaly Detection on VoIP traces?” highlighting the following considerations:

- The threshold's sensibility to the size and the type of the trace is an unavoidable issue; the only solution for obtaining results, on a previously not analyzed trace, is using a range of thresholds as wide as possible (increasing the software's time requirements).
- The choice of the appropriate number of PCs is dependent to the number and size of anomalies in our trace:
  - In a “highly polluted” trace we can use low numbers of PCs (1-3) to detect volume anomalies and high numbers of PCs (more than 4) to detect behavioral anomalies.
  - In an “almost-clean” trace we should always use a low number of PCs, since it's sufficient for modeling the *normal* subspace and leaving the anomalies in the *anomalous* subspace.
- The impossibility of detecting more than one anomalous user per-time bin remains a big limitation (the only way of avoiding it is with a multiple round analysis, but it leads to a big rise of the execution time).
- The amount of users we are able to identify is quite low (in our results it was never bigger than

11,12 users, with a single threshold analysis; that's another reason for using a multiple thresholds analysis, since different sets of thresholds point out different sets of users).

Even if we have to pay attention on all these issues this AD method is valuable because of his possibility of detecting behavioral anomalies, even if they represent a very small volume of traffic, feature that makes this software useful, in particular if used in combination with other kinds of AD techniques (as we said when we compared the results with the ones obtained with VoIP Seal). In particular we think that the best solution would be using first an Anomaly Detector, with better performances in detecting volume anomalies, for deleting top-users from the trace, then using this software on the cleaned trace to detect behavioral anomalies.

Bibliography

[1]

Diagnosing network-wide traffic anomalies
A. Lakhina, M. Crovella and C. Diot
In ACM SIGCOMM, pp. 219-230, 2004.

[2]

Characterization of network-wide anomalies in traffic flows
A. Lakhina, M. Crovella and C. Diot
In ACM Internet Measurement Conference, pp. 201-206, 2004.

[3]

Mining anomalies using traffic feature distributions
A. Lakhina, M. Crovella and C. Diot
Tech. Rep., 2005

[4]

Detection and Identification of network anomalies using sketch subspaces
X. Li, F. Bian, M. Crovella, C. Diot, R. Govidan, G. Iannaccone and A. Lakhina
IMC '06: Proceedings of 4th Internationa Workshop on Internet Performance, Simulation, Monitoring and Measurements, IPS-MOME, 2006

[5]

Improving the Performance of PCA-based Anomaly Detection Systems by means of Multiple Time-Scales Analysis and K-L Divergence
Christian Callegari, Loris Gazzarrini, Stefano Giordano, Michele Pagano, and Teresa Pepe
Dept. of Information Engineering, University of Pisa, ITALY.

[6]

An overview of anomaly detection techniques: Existing solutions and latest technological trends
A. Patcha, J. Park
Computer Networks 51 (2007), 3448-3470

[7]

Computer security threat monitoring and surveillance
J.P. Anderson
James P. Anderson Co., Fort, Washington, PA, USA, Technical Report 98-17, April 1980.

[8]

An intrusion-detection model
D.E. Denning
IEEE Transactions in Software Engineering 13 (1987) 222-232.

[9]

Research in intrusion-detection systems: A survey
S. Axelsson
Department of Computer Engineering, Chalmers University of Technology, Goteborg, Sweden, Technical Report 98--17, December 1998.

[10]

An Application of Pattern Matching in Intrusion Detection
S. Kumar and E. H. Spafford
The COAST Project, Department of Computer Sciences, Purdue University, West Lafayette, IN, USA., Technical Report CSD-TR-94-013, June 17 1994

[11]

Haystack: An Intrusion Detection System
S. E. Smaha
IEEE Fourth Aerospace Computer Security Applications Conference, Orlando, FL, 1988, pp. 37 – 44

[12]

Requirements and model for IDES---A real-time intrusion detection system
D. E. Denning and P. G. Neumann
Menlo Park, CA 94025-3493, Technical report # 83F83-01-00, 1985

[13]

A Real- time Intrusion Detection Expert System (IDES)
T. F. Lunt, A. Tamaru, F. Gilham, R. Jagannathm, C. Jalali, P. G. Neumann, H. S. Javitz, A. Valdes, and T. D. Garvey
Computer Science Laboratory, SRI International, Menlo Park, CA, USA, Final Technical Report February 1992

[14]

Next-generation Intrusion Detection Expert System (NIDES): A Summary
D. Anderson, T. Frivold, and A. Valdes
Computer Science Laboratory, SRI International, Menlo Park, CA 94025, Technical Report SRI-CSL-95-07, May 1995

[15]

Next-generation intrusion detection expert system (NIDES), Software Users Manual, Beta-Update release
D. Anderson, T. Frivold, A. Tamaru, and A. Valdes
Computer Science Laboratory, SRI International, Menlo Park, CA, USA, Technical Report SRI--CSL--95--0, May 1994

[16]

Detecting Unusual Program Behavior Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES)
D. Anderson, T. F. Lunt, H. Javitz, A. Tamaru, and A. Valdes
Computer Science Laboratory, SRI International, Menlo Park, CA, USA SRI-CSL-95-06, May 1995

[17]

Practical automated detection of stealthy portscans
S. Staniford, J. A. Hoagland, and J. M. McAlerney
Journal of Computer Security, vol. 10, pp. 105-136, 2002

[18]

Snort - Lightweight Intrusion Detection for Networks
M. Roesch
Proceedings of the 13th USENIX conference on System administration Seattle, Washington 1999 pp. 229-238

[19]

Multivariate Statistical Analysis of Audit Trails for Host-Based Intrusion Detection
N. Ye, S. M. Emran, Q. Chen, and S. Vilbert
IEEE Transactions on Computers, vol. 51, pp. 810-820, July 2002

[20]

Service specific anomaly detection for network intrusion detection
C. Krügel, T. Toth, and E. Kirda
2002 ACM symposium on Applied computing Madrid, Spain 2002, pp. 201 – 208

[21]

A case study of Ethernet anomalies in a distributed computing environment
R. A. Maxion and F. E. Feather
IEEE Transactions on Reliability, vol. 39, pp. 433--443, October 1990 1990

[22]

Information Theoretic Measures for Anomaly Detection
W. Lee and D. Xiang
2001 IEEE Symposium on Security and Privacy, Washington, DC, USA, 2001, pp. 130 – 143

[23]

A Sense of Self for Unix Processes
S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff
IEEE Symposium on Research in Security and Privacy, Oakland, CA, USA, 1996, pp. 120—128

[24]

Intrusion Detection Using Sequences of System Calls
S. A. Hofmeyr, S. Forrest, and A. Somayaji
Journal of Computer Security, vol. 6, pp. 151-180, 1998.

[25]

Fast Effective Rule Induction
W. W. Cohen
2th International Conference on Machine Learning, Tahoe City, CA, 1995, pp. 115—123.

[26]

Modeling System Calls for Intrusion Detection with Dynamic Window Sizes
E. Eskin, S. J. Stolfo, and W. Lee
DARPA Information Survivability Conference & Exposition II, Anaheim, CA 2001, pp. 165 – 175.

[27]

Detecting Intrusions Using System Calls: Alternative Data Models
C. Warrender, S. Forrest, and B. Pearlmutter
IEEE Symposium on Security and Privacy, Oakland, CA, USA, 1999, pp. 133-145.

[28]

A Tutorial on Learning With Bayesian Networks
D. Heckerman
Microsoft Research, Technical Report MSR-TR- 95-06, March 1995

[29]

Bayesian Event Classification for Intrusion Detection
C. Kruegel, D. Mutz, W. Robertson, and F. Valeur
19th Annual Computer Security Applications Conference, Las Vegas, NV, 2003

[30]

Adaptive Model-based Monitoring for Cyber Attack Detection
A. Valdes and K. Skinner
Recent Advances in Intrusion Detection Toulouse, France, 2000, pp. 80—92

[31]

Probabilistic Networks with Undirected Links for Anomaly Detection
N. Ye, M. Xu, and S. M. Emran
IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop, West Point, New York, 2000

[32]

EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances
P. A. Porras and P. G. Neumann
20th NIST-NCSC National Information Systems Security Conference, Baltimore, MD, USA, 1997, pp. 353—365

[33]

A Comparative Study of Principal Component Analysis Techniques
R. A. Calvo, M. Partridge, and M. A. Jabri
Ninth Australian Conference on Neural Networks, Brisbane, Queensland, Australia, 1998.

[34]

Analysis of a Complex of Statistical Variables into Principal Components
H. Hotelling
Journal of Educational Psychology, vol. 24, pp. 417-441, 498-520, 1933.

[35]

Identifying Intrusions in Computer Networks with Principal Component Analysis
W. Wang and R. Battiti
The First International Conference on Availability, Reliability and Security, Vienna, Austria, 2006, pp. 270 – 279

[36]

A Novel Anomaly Detection Scheme Based on Principal Component Classifier
M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang
IEEE Foundations and New Directions of Data Mining Workshop, Melbourne, Florida, USA, 2003, pp. 172-179

[37]

Efficient Intrusion Detection Using Principal Component Analysis
Y. Bouzida, F. e. e. Cuppens, N. Cuppens-Boulahia, and S. Gombault
3ème Conférence sur la Sécurité et Architectures Réseaux (SAR), Orlando, FL, USA, 2004.

[38]

A Novel Intrusion Detection Method Based on Principle Component Analysis in Computer Security
W. Wang, X. Guan, and X. Zhang
International Symposium on Neural Networks, Dalian, China, 2004, pp. 657 – 662

[39]

Robustness of the Markov-Chain Model for Cyber-Attack Detection
N. Ye and Y. Z. C. M. Borror
IEEE Transactions on Reliability, vol. 53, pp. 116-123, March 2004

[40]

Host-Based Intrusion Detection Using Dynamic and Static Behavioral Models
D.-Y. Yeung and Y. Ding
Pattern Recognition, vol. 36, pp. 229--243, 2003

[41]

PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic
M. V. Mahoney and P. K. Chan
Department of Computer Sciences, Florida Institute of Technology, Melbourne, FL, USA, Technical Report CS-2001- 4, April 2001

[42]

Learning Models of Network Traffic for Detecting Novel Attacks
M. V. Mahoney and P. K. Chan
Computer Science Department, Florida Institute of Technology CS-2002-8, August 2002 2002

[43]

Learning Nonstationary Models of Normal Network Traffic for Detecting Novel Attacks
M. V. Mahoney and P. K. Chan,
Eighth ACM SIGKDD international conference on Knowledge discovery and data mining, Edmonton, Canada, 2002, pp. 376--385

[44]

The 1999 DARPA off-line Intrusion Detection Evaluation
R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das
Computer Networks,, vol. 34, pp. 579-595, 2000

[45]

A Data Mining and CIDF Based Approach for Detecting Novel and Distributed Intrusions
W. Lee, R. A. Nimbalkar, K. K. Yee, S. B. Patil, P. H. Desai, T. T. Tran, and S. J. Stolfo
3rd International Workshop on Recent Advances in Intrusion Detection (RAID 2000), Toulouse, France, 2000, pp. 49-65

[46]

3rd International Workshop on Recent Advances in Intrusion Detection (RAID 2000), Toulouse, France, 2000, pp. 49-65
W. Lee and S. J. Stolfo
7th USENIX Security Symposium (SECURITY-98), Berkeley, CA, USA, 1998, pp. 79--94

[47]

Adaptive Intrusion Detection: a Data Mining Approach
W. Lee, S. J. Stolfo, and K. W. Mok
Artificial Intelligence Review, vol. 14, pp. 533--567, 2000

[48]

Data Mining: Challenges and Opportunities for Data Mining During the Next Decade
R. Grossman
1997

[49]

C4.5: Programs for Machine Learning
J. R. Quinlan
Los Altos, CA: Morgan Kaufmann, 1993.

[50]

A Data Mining Framework for Building Intrusion Detection Models
W. Lee, S. J. Stolfo, and K. W. Mok
IEEE Symposium on Security and Privacy, Oakland, CA, 1999, pp. 120-132

[51]

Security is fuzzy!: Applying the Fuzzy Logic Paradigm to the Multipolicy Paradigm
H. H. Hosmer
1992-1993 workshop on New security paradigms Little Compton, Rhode Island, United States 1993

[52]

Fuzzy Data Mining and Genetic Algorithms Applied to Intrusion Detection
S. M. Bridges and R. B. Vaughn
National Information Systems Security Conference, Baltimore, MD, 2000

[53]

Fuzzy network profiling for intrusion detection
J. E. Dickerson and J. A. Dickerson
19th International Conference of the North American Fuzzy Information Processing Society (NAFIPS), Atlanta, GA 2000, pp. 301 - 306

[54]

Using Genetic Algorithm for Network Intrusion Detection
W. Li
C. S. G. Department of Energy, Ed., 2004, pp. 1-8.

[55]

An Approach to Implement a Network Intrusion Detection System using Genetic Algorithms
M. M. Pillai, J. H. P. Eloff, and H. S. Venter
2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries, Stellenbosch, Western Cape, South Africa, 2004, pp. 221 - 228

[56]

Evolving Fuzzy Classifiers for Intrusion Detection
J. Gomez and D. Dasgupta
IEEE Workshop on Information Assurance, United States Military Academy, NY, 2001

[57]

Applying Genetic Programming to Intrusion Detection
M. Crosbie and G. Spafford
Working Notes for the AAAI Symposium on Genetic Programming, Cambridge, MA, 1995, pp. 1--8

[58]

A Real-Time Intrusion Detection System Based on Learning Program Behavior
A. K. Ghosh, C. Michael, and M. Schatz
Third International Workshop on Recent Advances in Intrusion Detection Toulouse, France, 2000, pp. 93-109

[59]

A study in using neural networks for anomaly and misuse detection
A. K. Ghosh and A. Schwartzbard
Eighth USENIX Security Symposium, Washington, DC, 1999, pp. 141—151

[60]

Learning Program Behavior Profiles for Intrusion Detection
A. K. Ghosh, A. Schwartzbart, and M. Schatz
1st USENIX Workshop on Intrusion Detection and Network Monitoring, Santa Clara, CA, USA, 1999

[61]

Finding Structure in Time
J. L. Elman
Cognitive Science, vol. 14, pp. 179-211, 1990

[62]

Detecting Anomalous Network Traffic with Self-organizing Maps
M. Ramadas and S. O. B. Tjaden
6th International Symposium on Recent Advances in Intrusion Detection, Pittsburgh, PA, USA, 2003, pp. 36 – 54

[63]

Real Time Data Mining- based Intrusion Detection
W. Lee, S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, and J. Zhang
Second DARPA Information Survivability Conference and Exposition, Anaheim, CA, 2001, pp. 85--100

[64]

Determining the operational limits of an anomaly-based intrusion detector
K. M. C. Tan and R. A. Maxion
IEEE Journal on Selected Areas in Communication, vol. 2, pp. 96--110, 2003

[65]

Hierarchical Kohonenen Net for Anomaly Detection in Network Security
S. T. Sarasamma, Q. A. Zhu, and J. Huff
IEEE Transactions on Systems, Man and Cybernetics—PART B: Cybernetics, vol. 35, pp. 302-312, April 2005.

[66]

Identifying Important Features for Intrusion Detection Using Support Vector Machines and Neural Networks
A. H. Sung and S. Mukkamala
2003 Symposium on Applications and the Internet 2003, pp. 209-216

[67]

Intrusion Detection with Unlabeled Data Using Clustering
L. Portnoy, E. Eskin, and S. J. Stolfo
ACM Workshop on Data Mining Applied to Security, Philadelphia, PA, 2001

[68]

Efficient Algorithms for Mining Outliers from Large Data Sets
S. Ramaswamy, R. Rastogi, and K. Shim
ACM SIGMOD international conference on Management of data, Dallas, TX, USA, 2000, pp. 427 – 438

[69]

ADMIT: Anomaly-based Data Mining for Intrusions
K. Sequeira and M. Zaki
8th ACM SIGKDD international conference on Knowledge discovery and data mining, Edmonton, Alberta, Canada, 2002, pp. 386—395

[70]

Outliers in Statistical Data
V. Barnett and T. Lewis
John Wiley & Sons, 1994

[71]

Outlier detection for high dimensional data
C. C. Aggarwal and P. S. Yu
ACM SIGMOD International conference on management of data, 2001 pp. 37 – 46

[72]

LOF: Identifying Density-Based Local Outliers
M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander
ACM SIGMOD International Conference on Management of Data, Dallas, TX, , 2000, pp. 93-104

[73]

Algorithms for Mining Distance-Based Outliers in Large Datasets
E. M. Knorr and R. T. Ng
24th International Conference on Very Large Data Bases, New York, NY, USA, 1998, pp. 392—403

[74]

On tests and measures of groups divergence
P. C. Mahalanobis
Journal of the Asiatic Society of Bengal, vol. 26, p. 541, 1930

[75]

Mahalanobis Distance
Wikipedia
vol. 2006, 2006

[76]

Outlier Detection Using k-Nearest Neighbour Graph
V. Hautamaki, I. Karkkainen, and P. Franti
In Proceedings of the 17th International Conference on Pattern Recognition Los Alamitos, CA, USA, 2004, pp. 430-433

[77]

Use of K-Nearest Neighbor Classifier for Intrusion Detection
Y. Liao and V. R. Vemuri
Computers & Security, vol. 21, pp. 439-448, October 2002

[78]

The MINDS – Minnesota Intrusion Detection System
L. Ertöz, E. Eilertson, A. Lazarevic, P.-N. Tan, V. Kumar, J. Srivastava, and P. Dokas
Next Generation Data Mining Boston: MIT Press, 2004

[79]

Mining Association Rules between Sets of Items in Large Databases
R. Agrawal, T. Imielinski, and A. Swami
ACM SIGMOD Conference on Management of Data, Washington, D.C., 1993, pp. 207—216

[80]

Algorithms for Association Rule Mining - A General Survey and Comparison
J. Hipp, U. Güntzer, and G. Nakhaeizadeh
ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, USA, 2000, pp. 58—64

[81]

ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection
D. Barbará, J. Couto, S. Jajodia, and N. Wu
ACM SIGMOD Record: SPECIAL ISSUE: Special section on data mining for intrusion detection and threat analysis, vol. 30, pp. 15 - 24 2001

[82]

The 1999 DARPA off-line intrusion detection evaluation
R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das
Computer Networks: The International Journal of Computer and Telecommunications Networking, vol. 34, pp. 579 - 595 October 2000

[83]

A Serial Combination of Anomaly and Misuse IDSes Applied to HTTP Traffic
E. Tombini, H. Debar, L. Mé, and M. Ducassé
20th Annual Computer Security Applications Conference, Tucson, AZ, USA, 2004.

[84]

A Hybrid Network Intrusion Detection Technique Using Random Forests
J. Zhang and M. Zulkernine
The First International Conference on Availability, Reliability and Security, Vieena University of Technology, 2006, pp. 262 - 269

[85]

Random Forests
L. Breiman
Machine Learning, vol. 45, pp. 5-32, 2001

[86]

Real Time Data Mining-based Intrusion Detection
W. L. S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop, and J. Zhang
Second DARPA Information Survivability Conference and Exposition, Anaheim, CA, USA, 2001, pp. 85--100

[87]

An improved data stream summary: the count-min sketch and its applications
G. Cormode and S. Muthukrishnan
Journal of Algorithms, vol. 55, no. 1, pp. 58 – 75, 2005.

[88]

Tabulation based 4-universal hashing with applications to second moment estimation
M. Thorup and Y. Zhang
SODA '04: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms, (Philadelphia, PA, USA), pp. 615–624, Society for Industrial and Applied Mathematics, 2004.

[89]

Analyzing telemarketer behavior in massive telecom data records
Nico d'Heureuse, Sandra Tartarelli, Saverio Niccolini
NEC Europe Ltd., NEC Laboratories Europe, Germany

Appendix A

(Single round analysis)

Number of anomalous time-bins detected with “single threshold-single number of PCs” analysis (5 hash functions)

Figure A.1

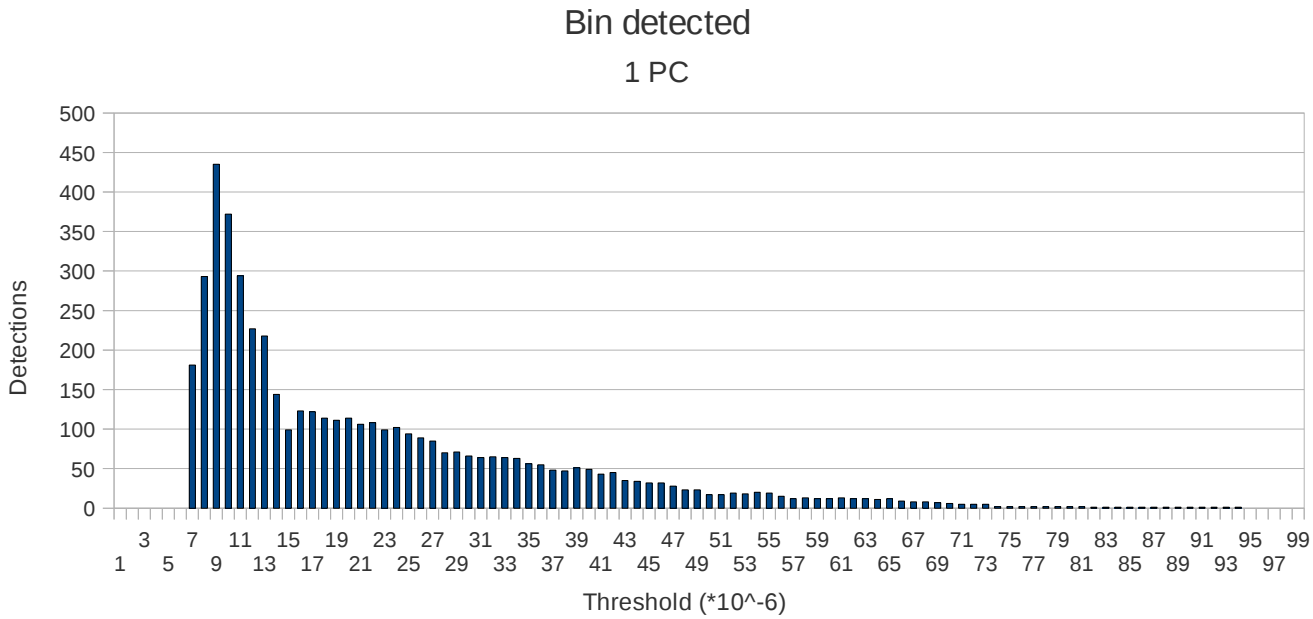


Figure A.2

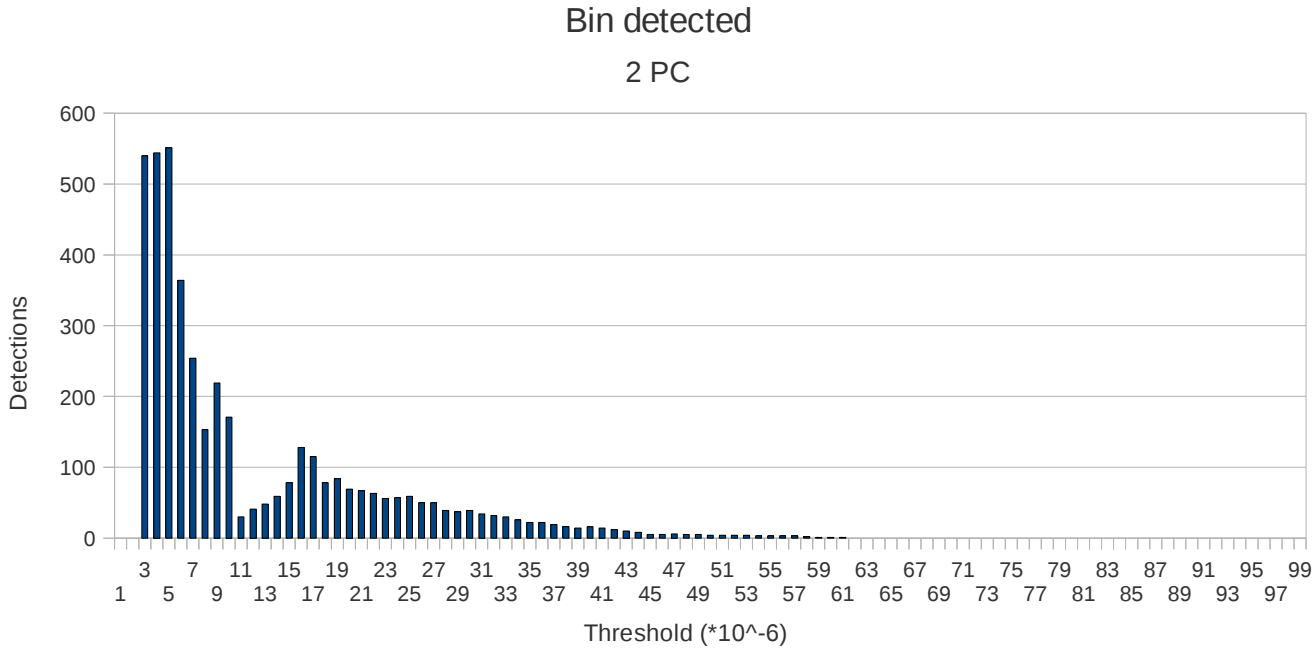


Figure A.3

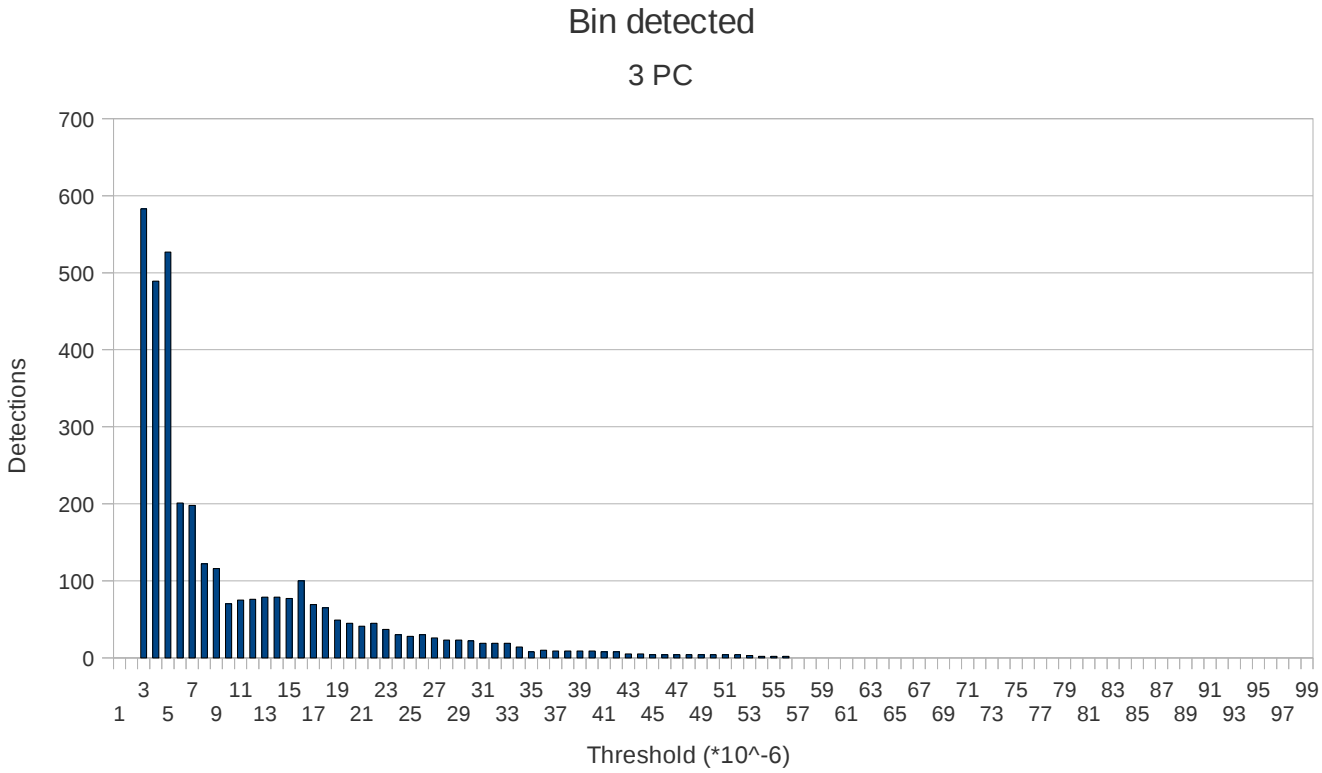


Figure A.4

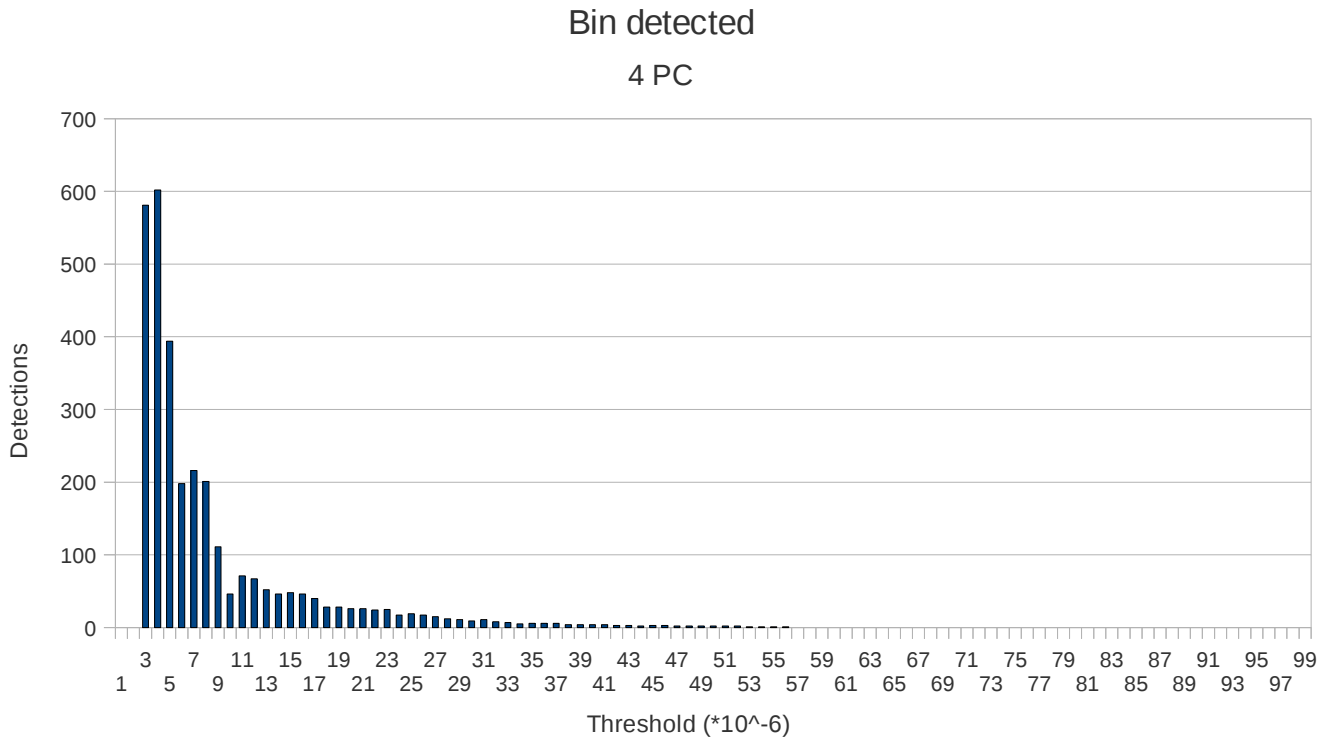


Figure A.5

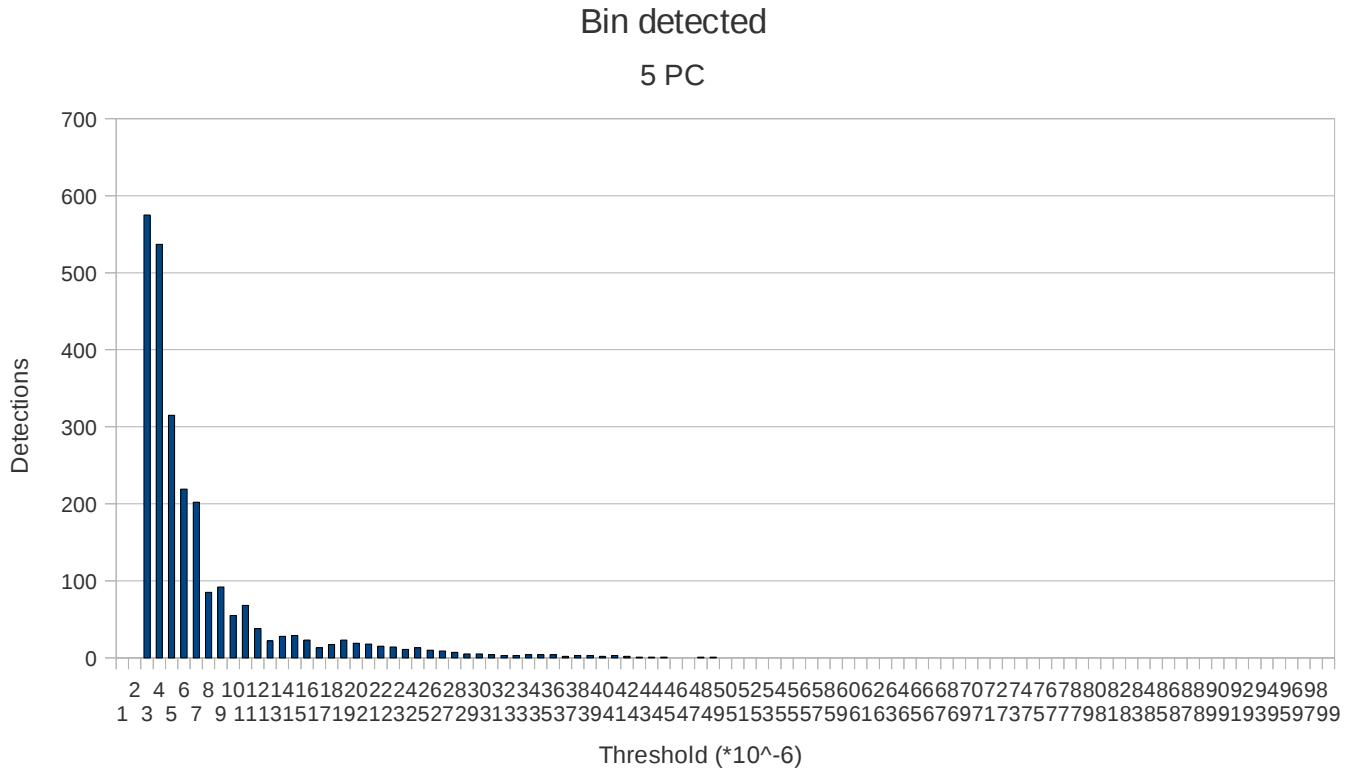


Figure A.6

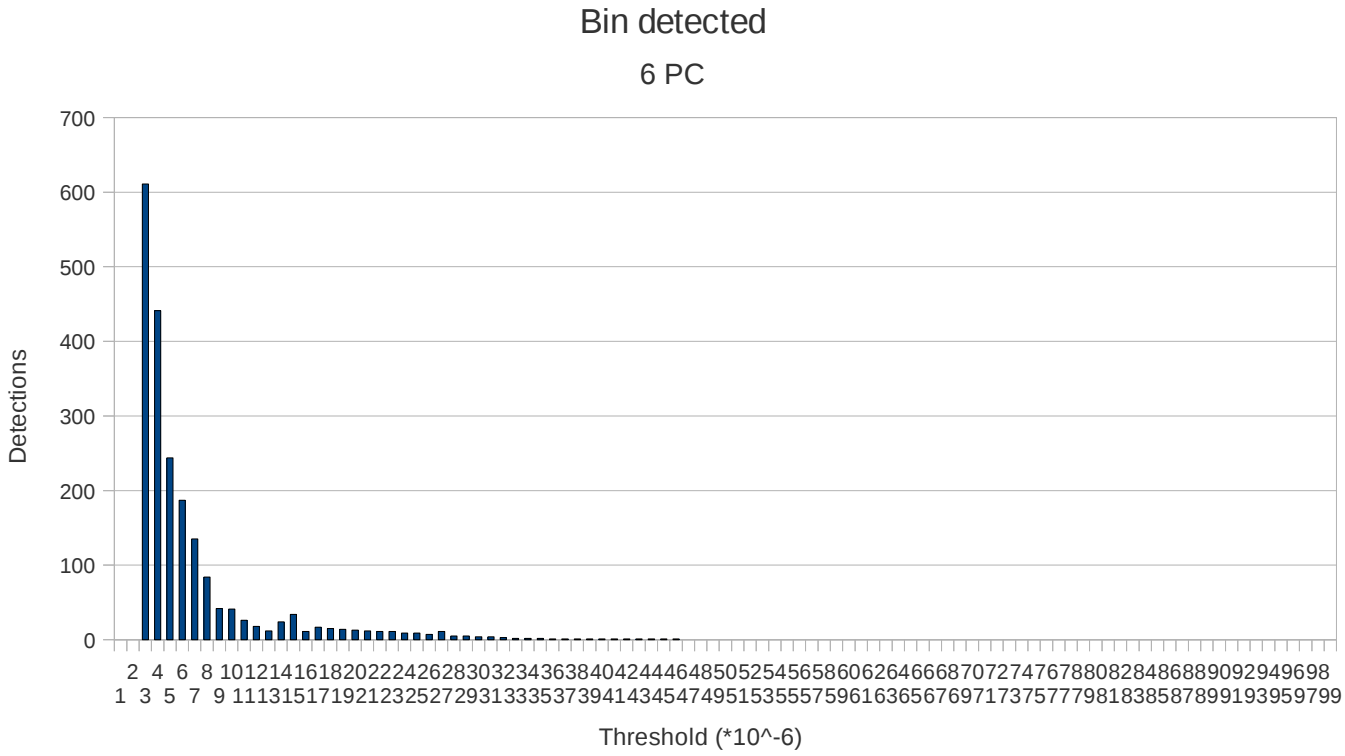


Figure A.7

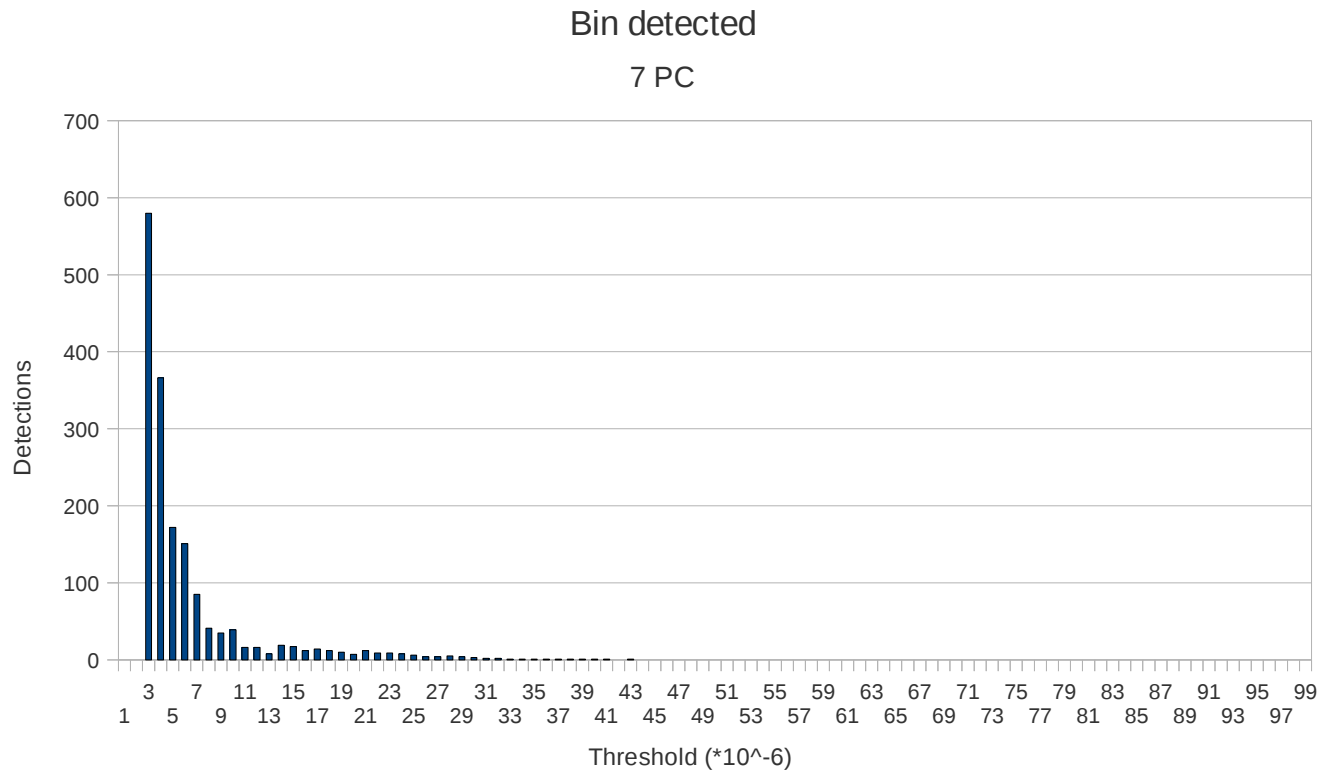


Figure A.8

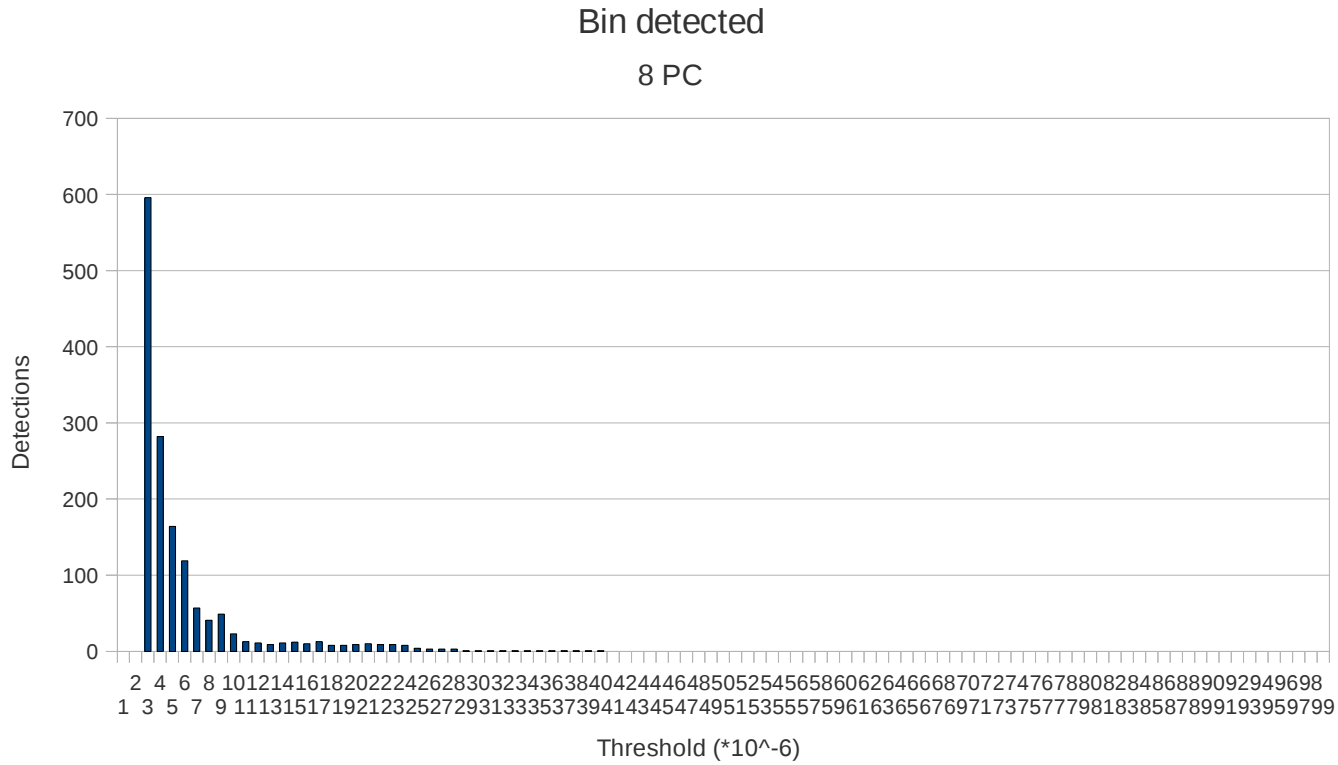


Figure A.9

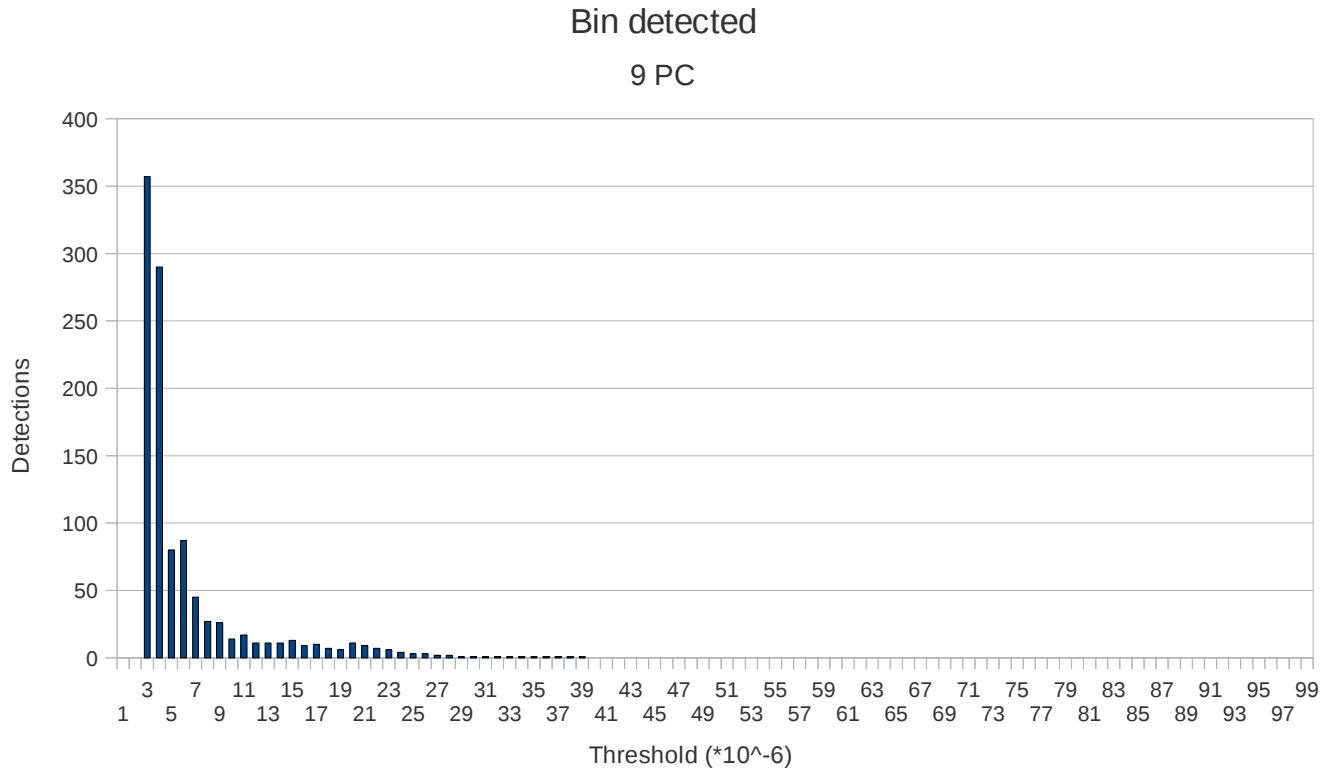
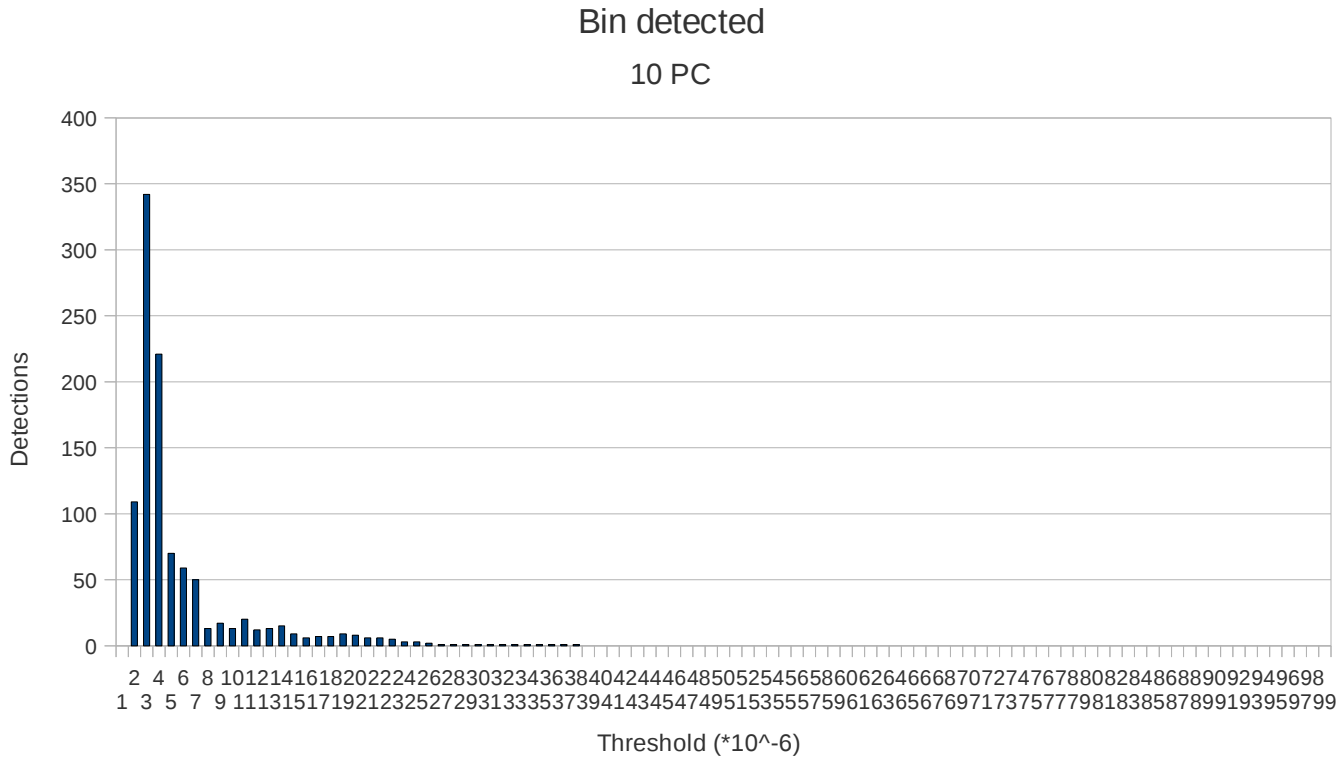


Figure A.10



Number of anomalous users recognized with “single threshold-single number of PCs” analysis (5 hash functions)

Figure A.11

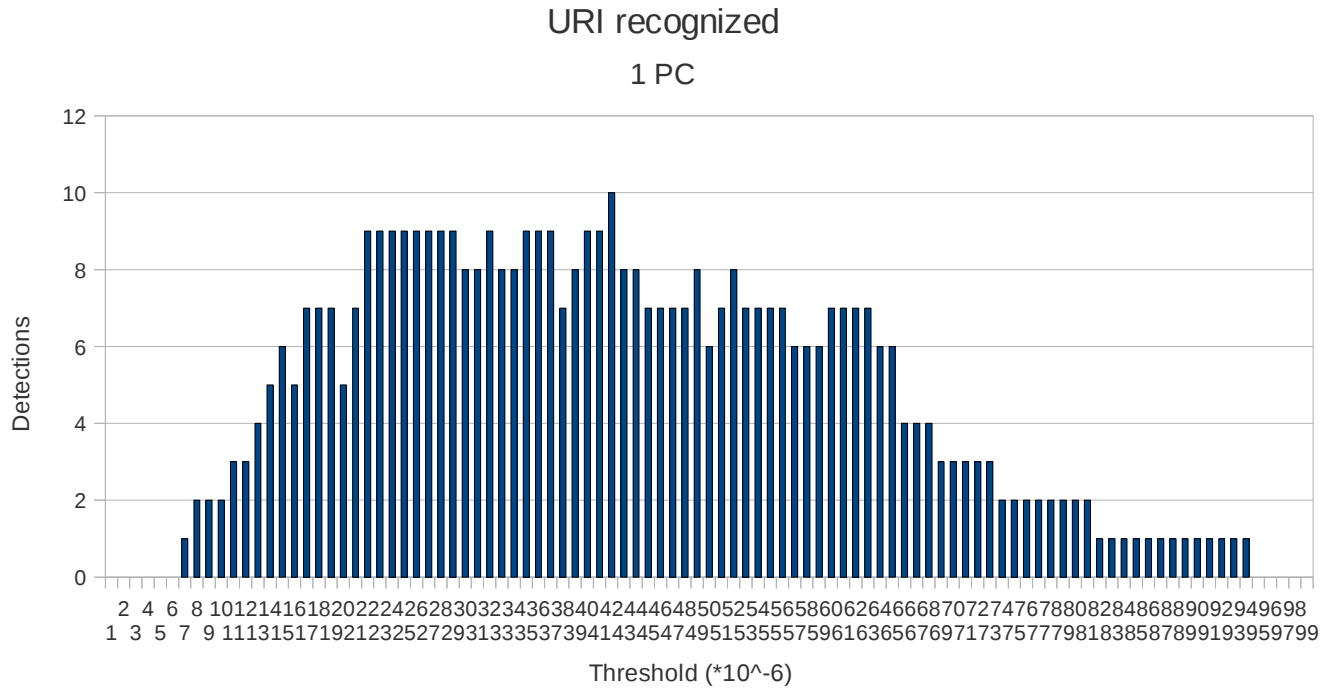


Figure A.12

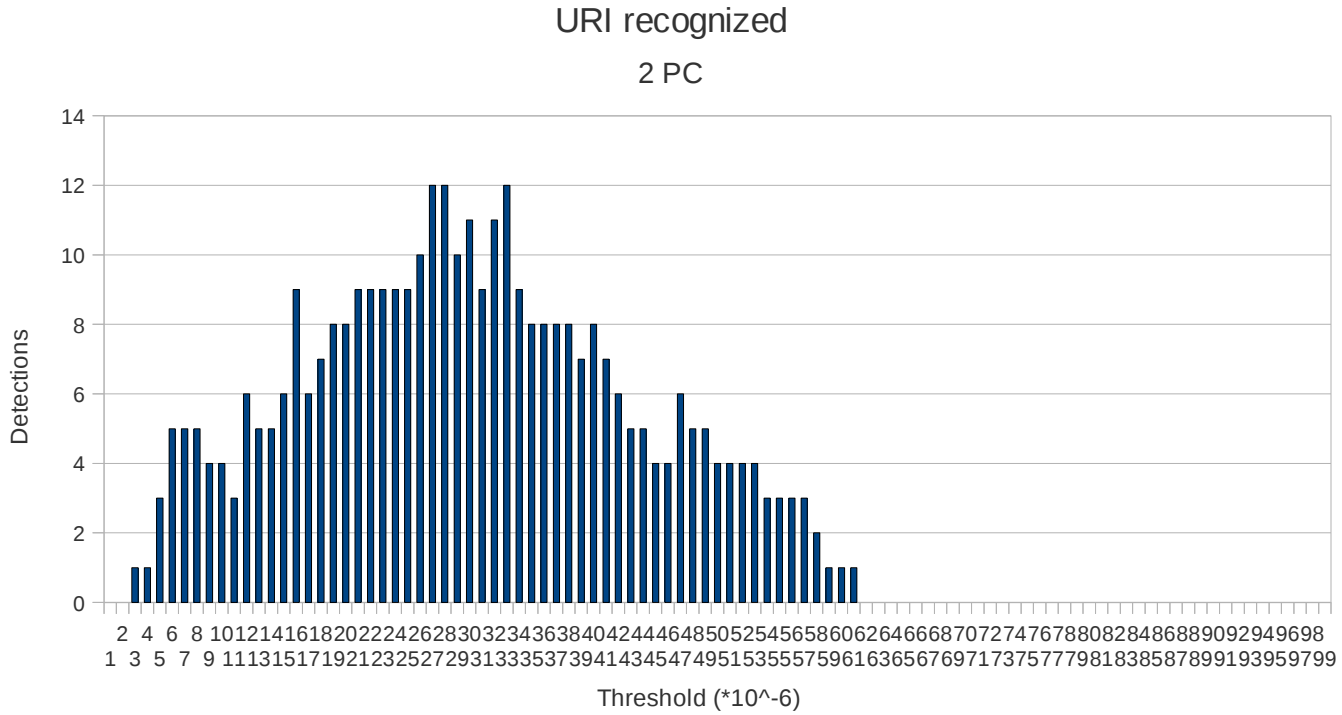


Figure A.13

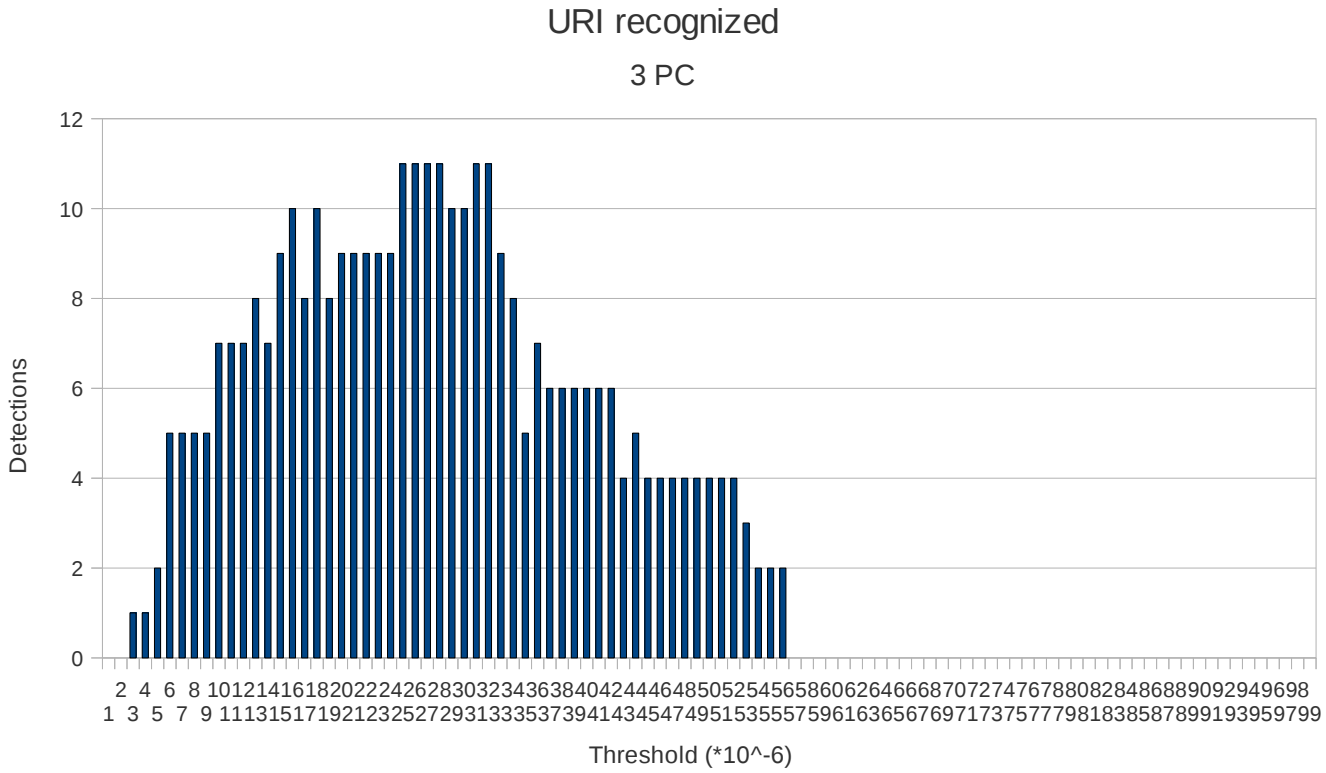


Figure A.14

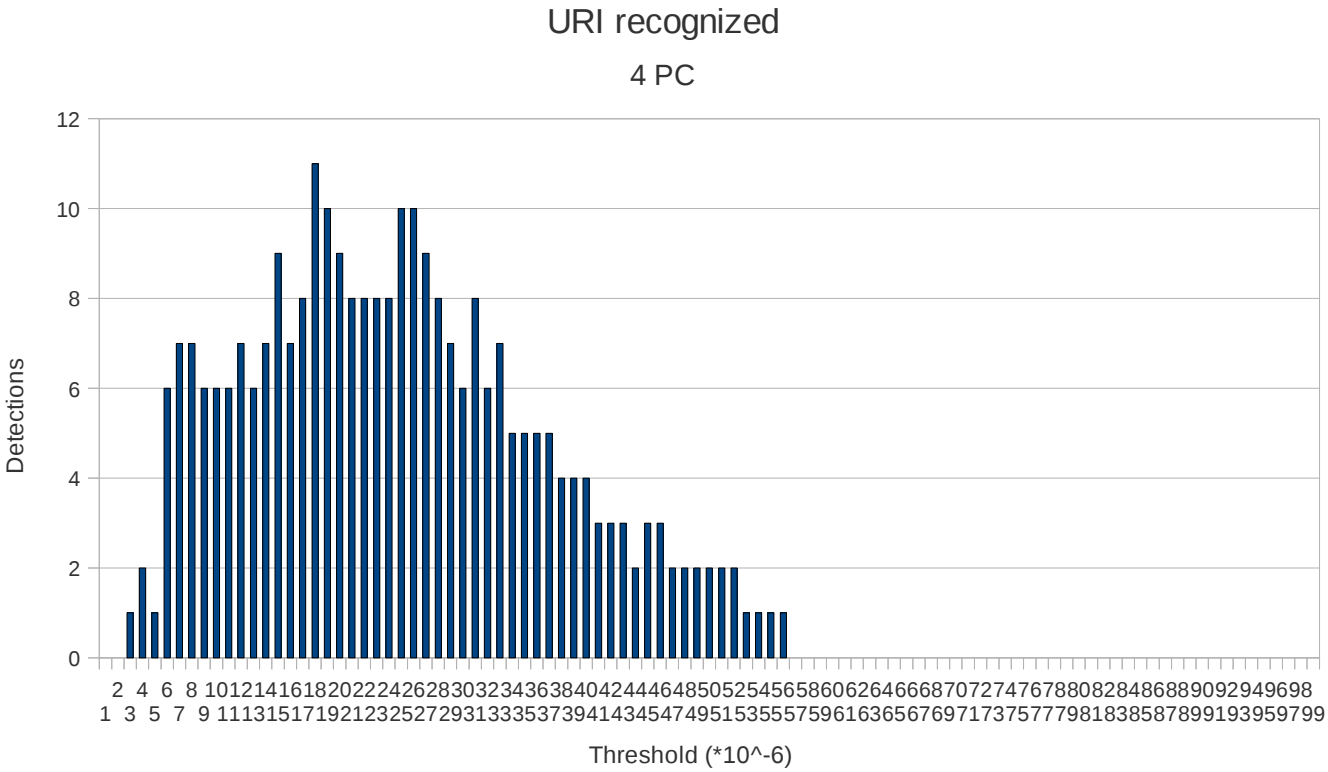


Figure A.15

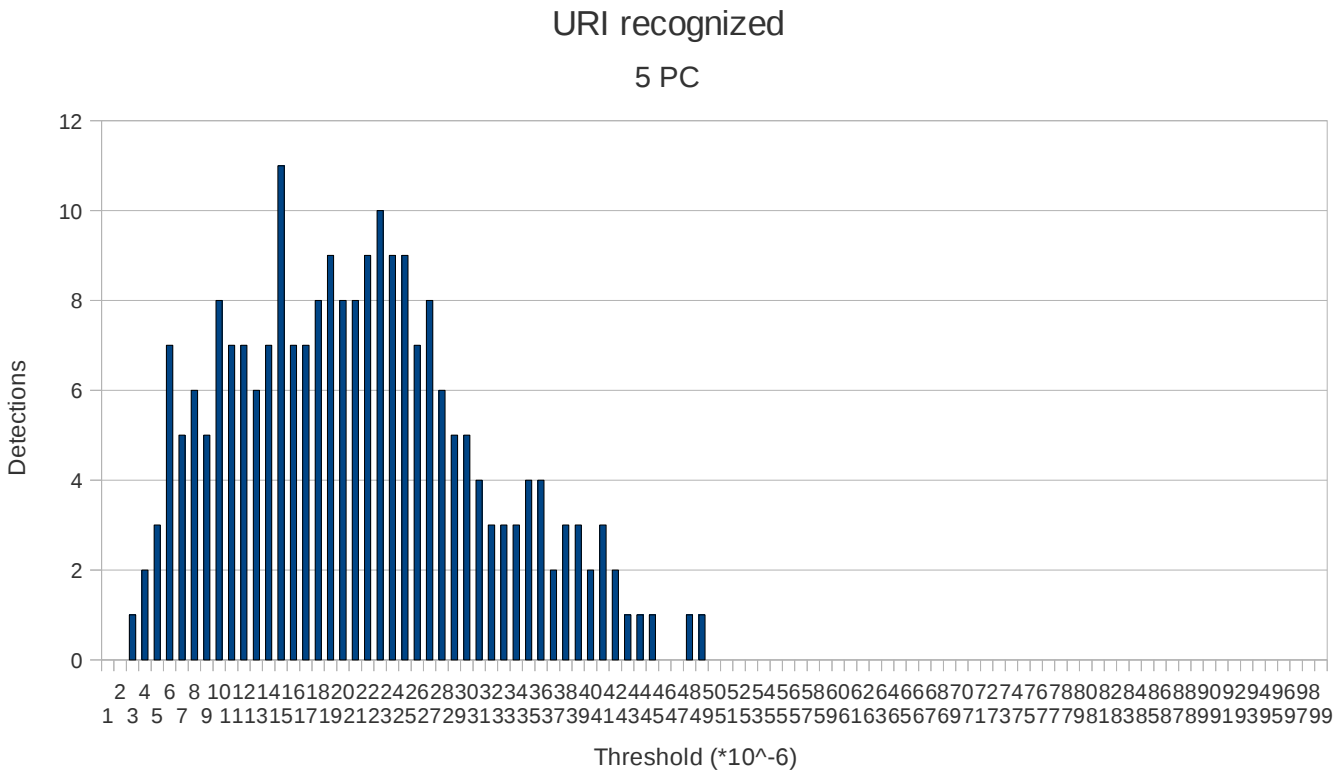


Figure A.16

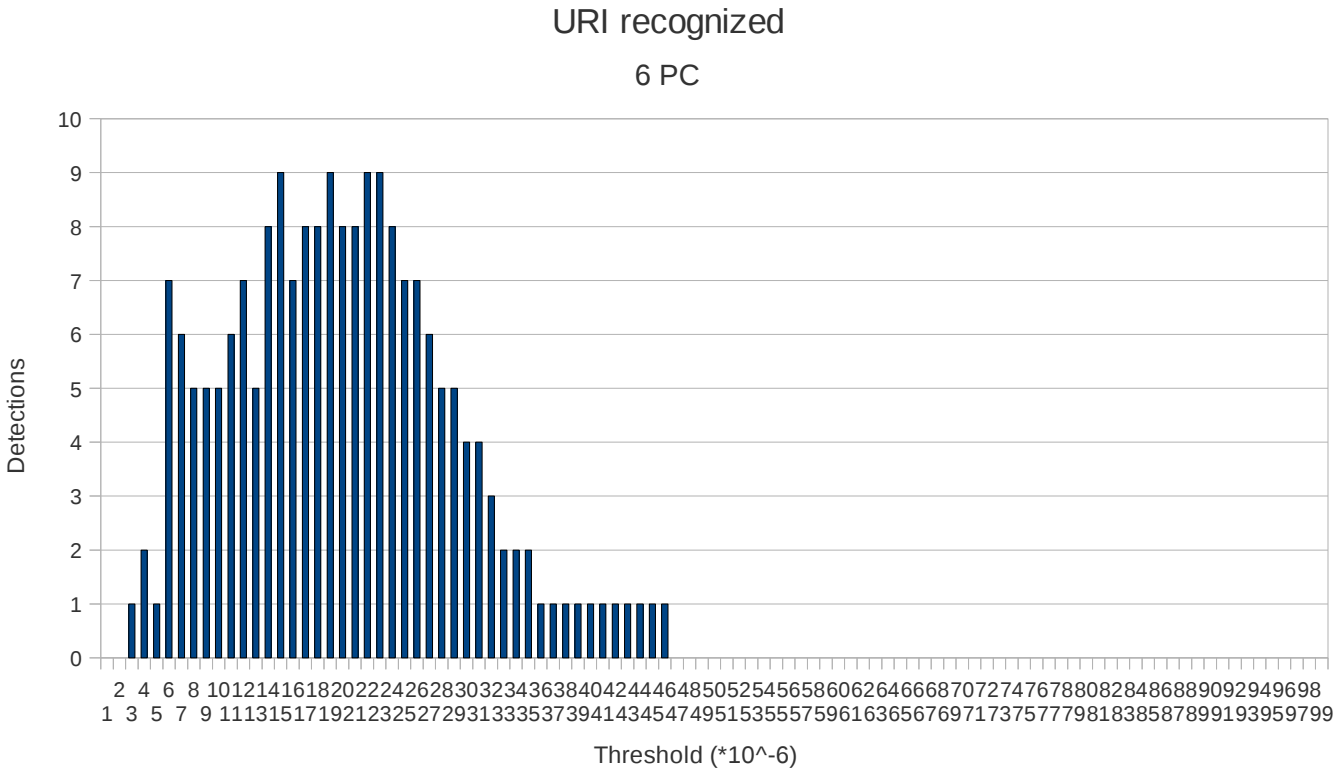


Figure A.17

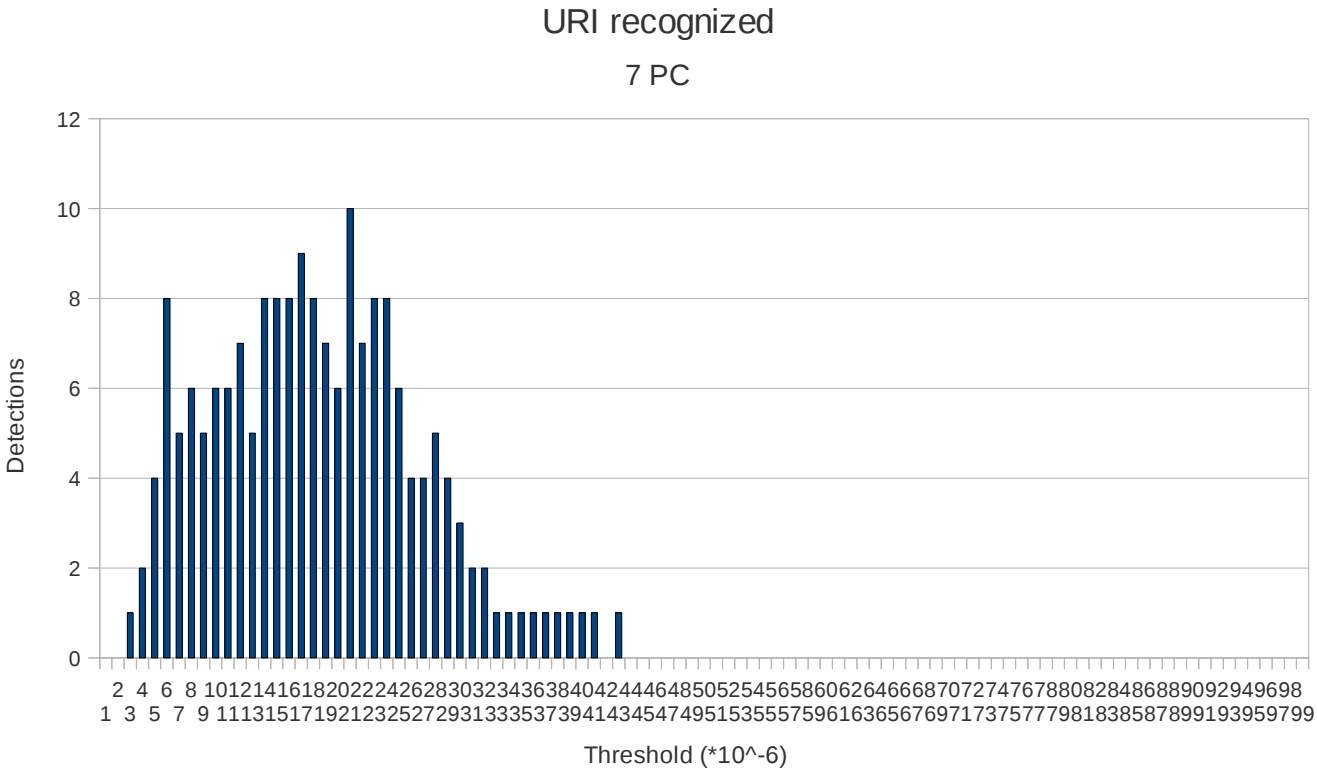


Figure A.18

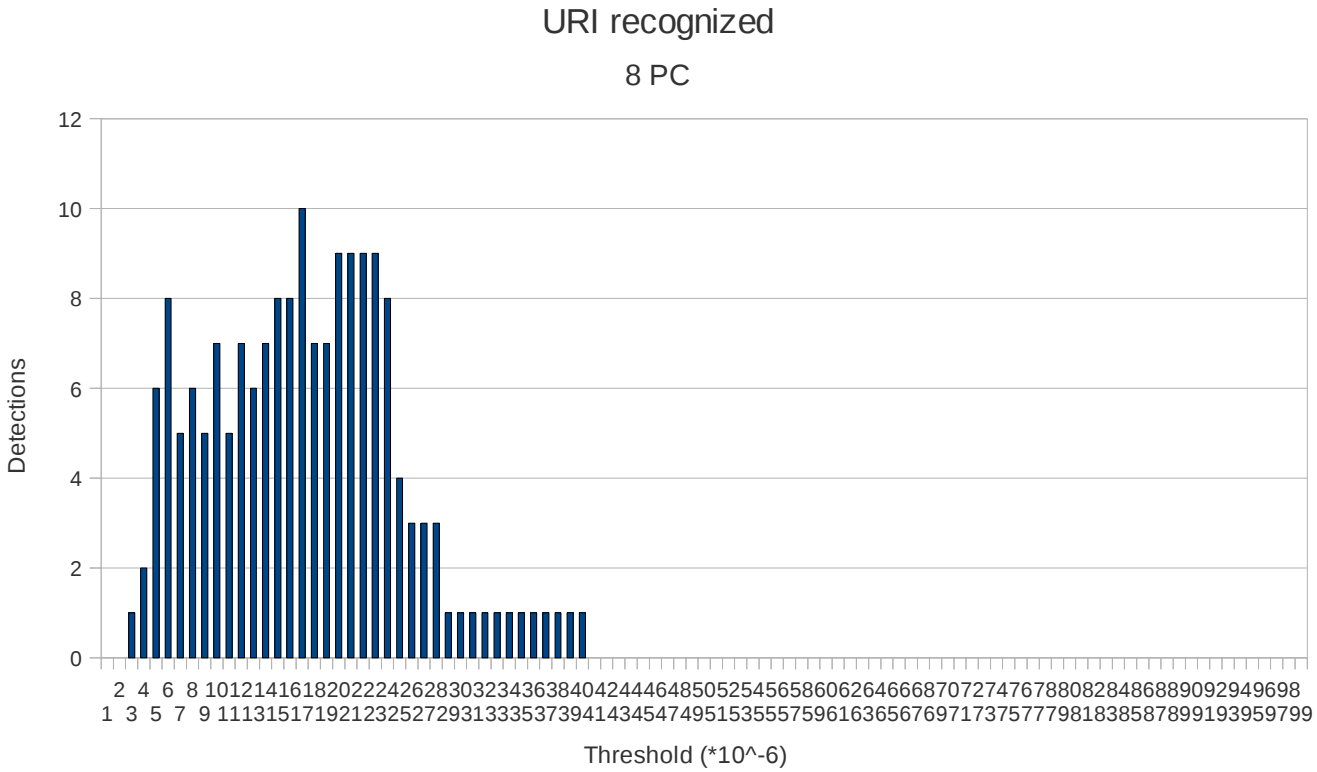


Figure A.19

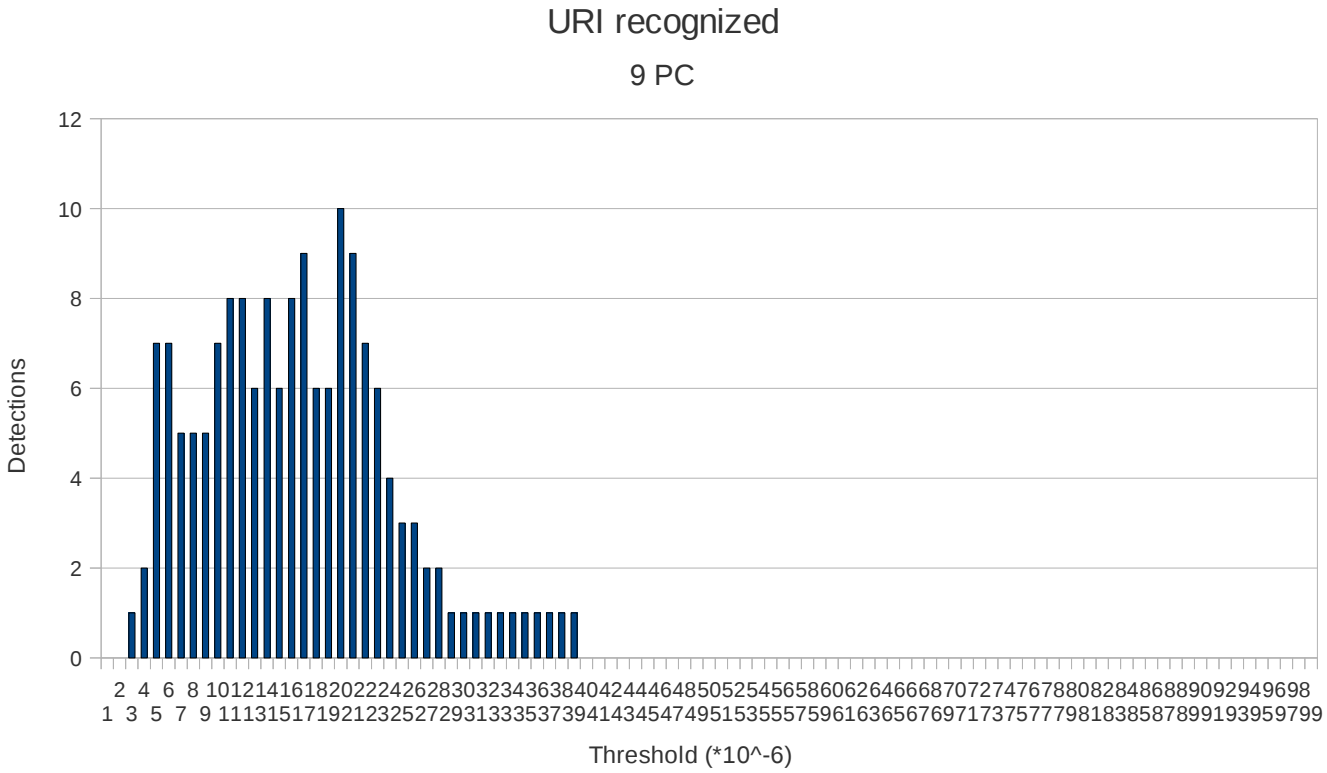
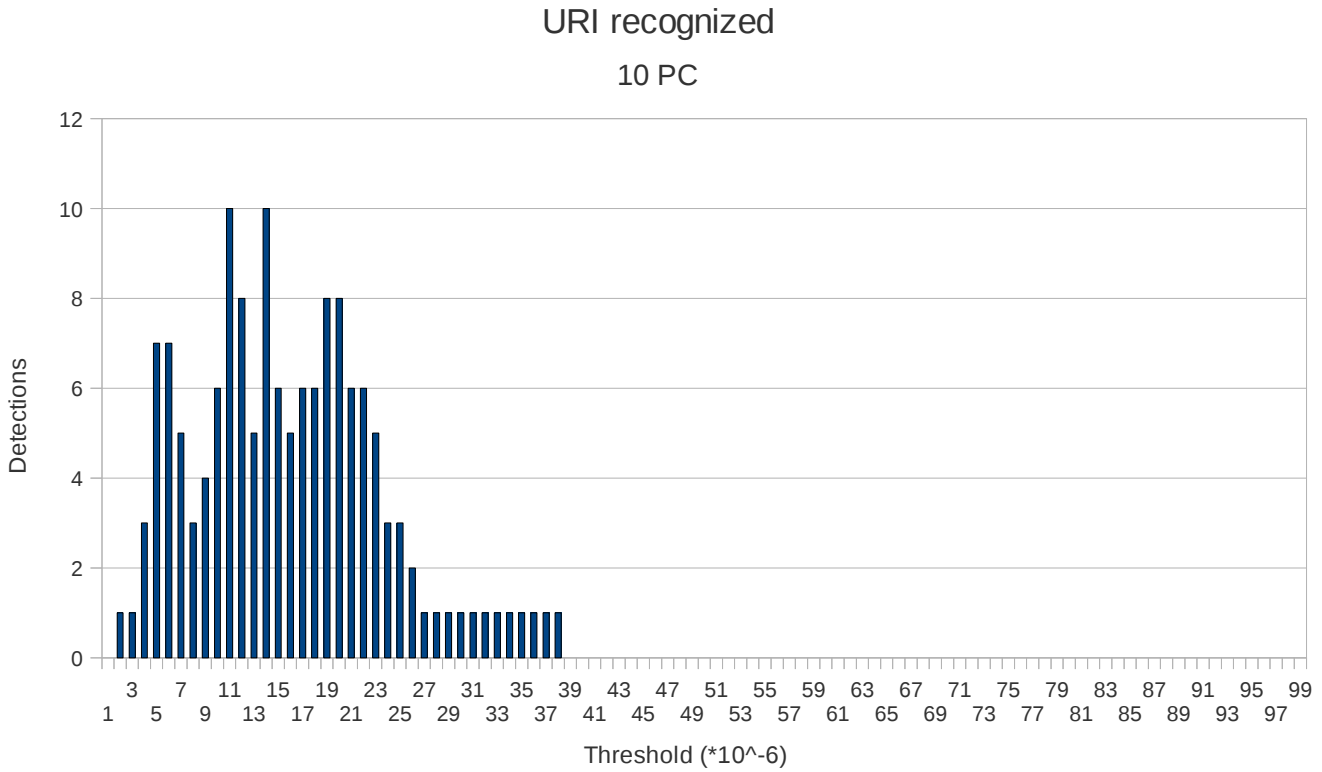


Figure A.20



Appendix B

(Single spike artificial anomalies)

Number of detections for each time-bin in a "100 thresholds-10 numbers of PCs" analysis VS ratio of global calls per time-bin on their maximum value

Figure B.1

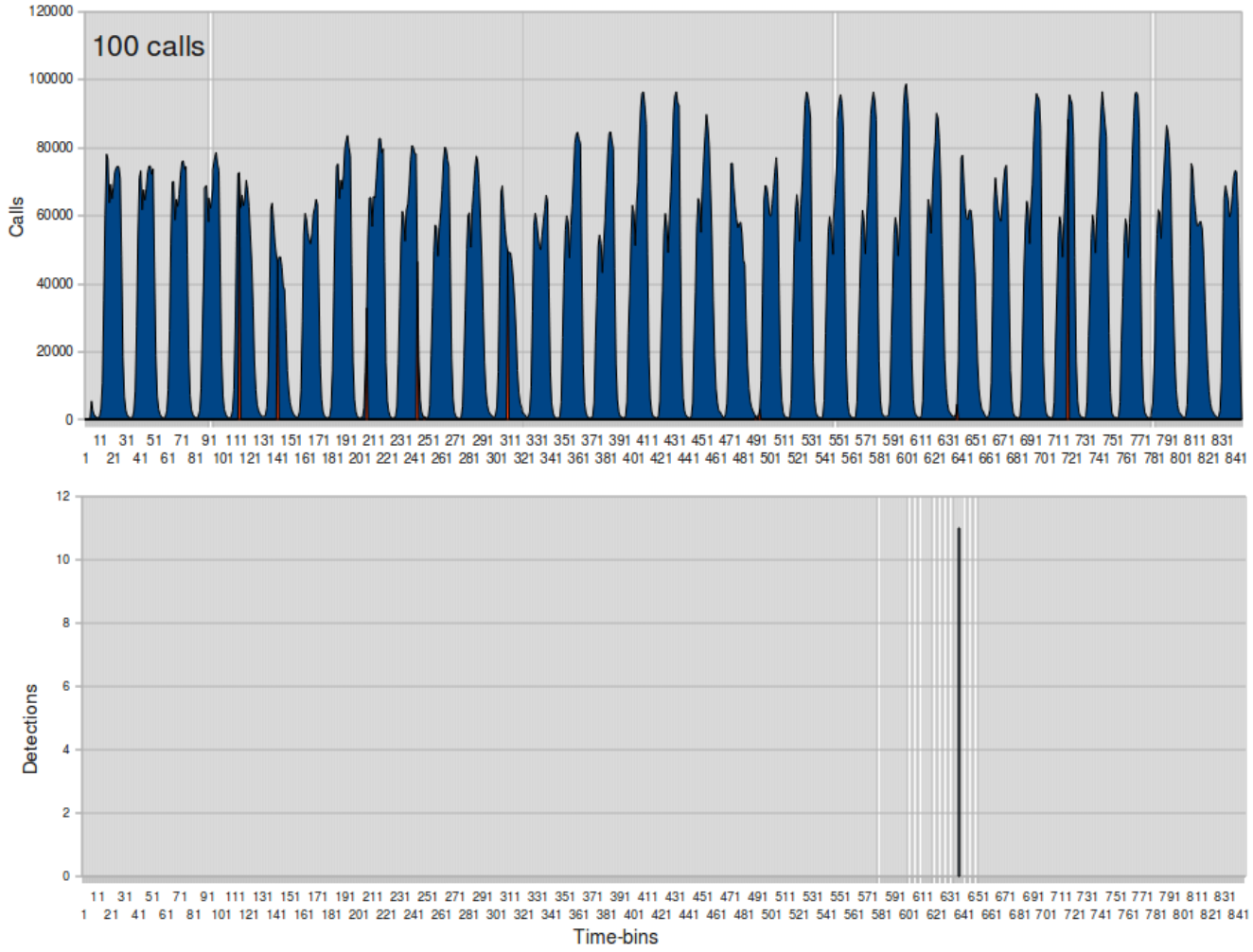


Figure B.2

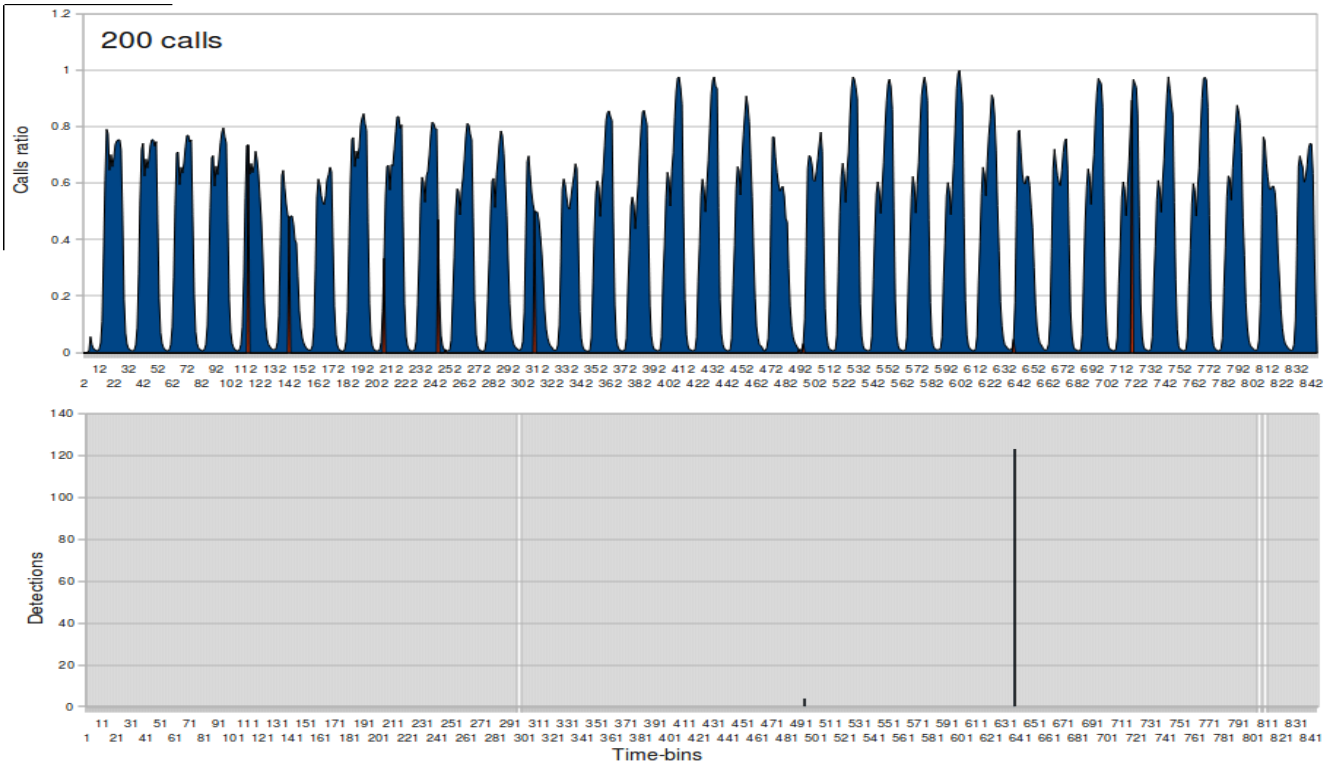


Figure B.3

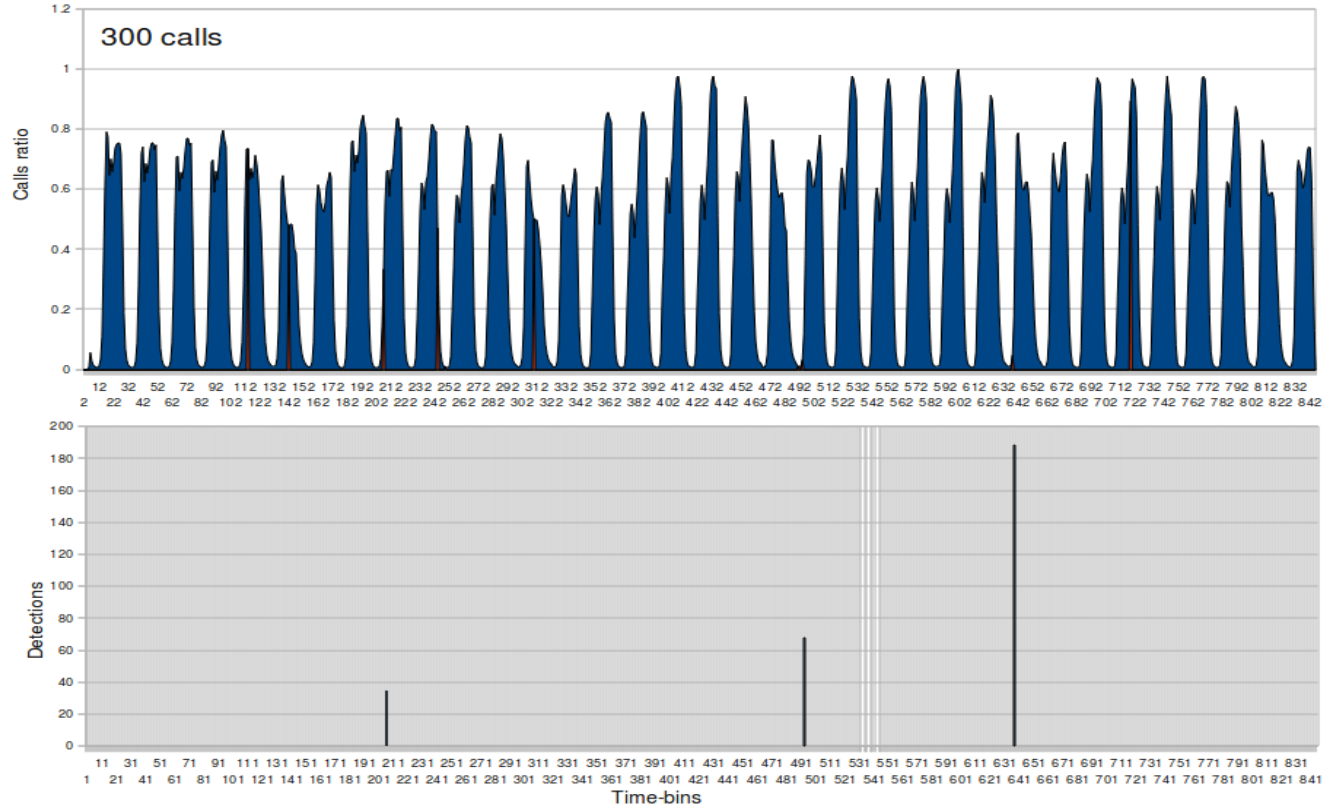


Figure B.4

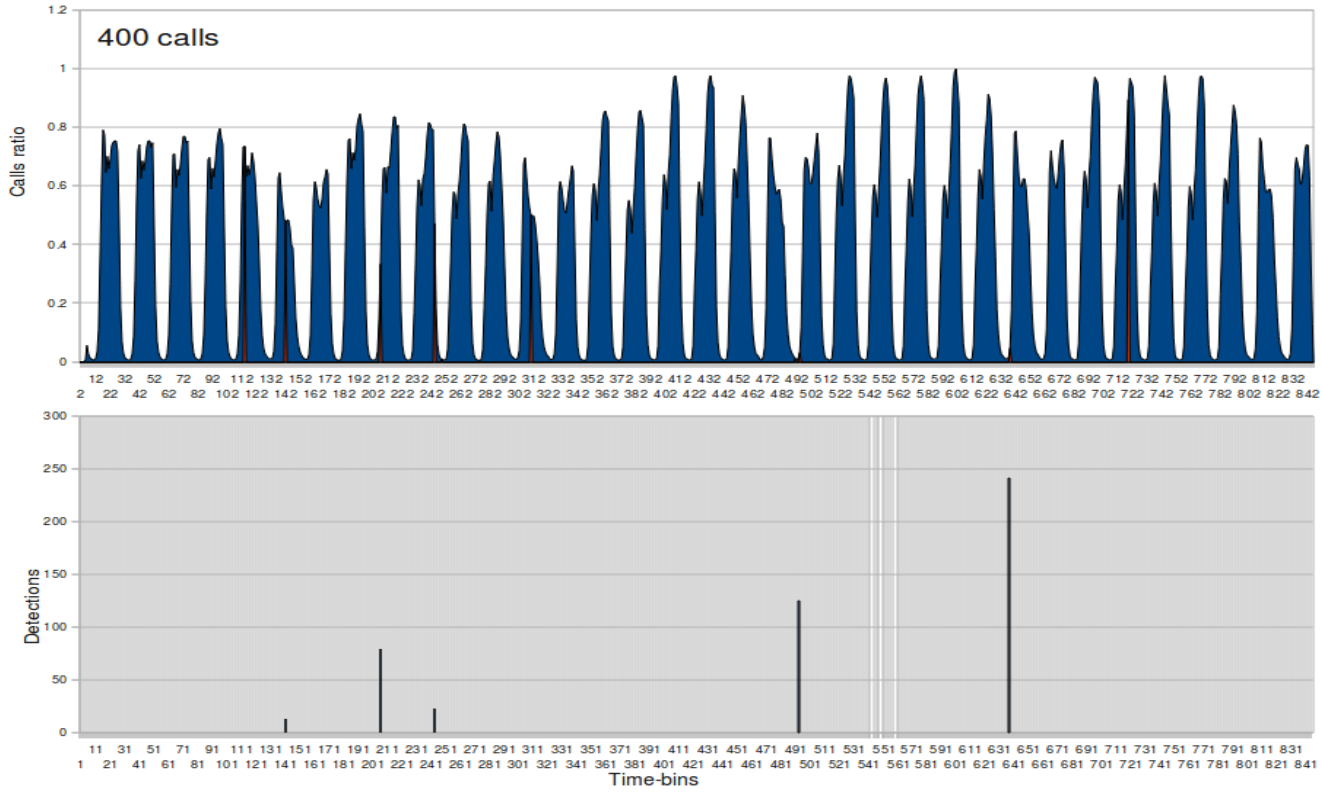


Figure B.5

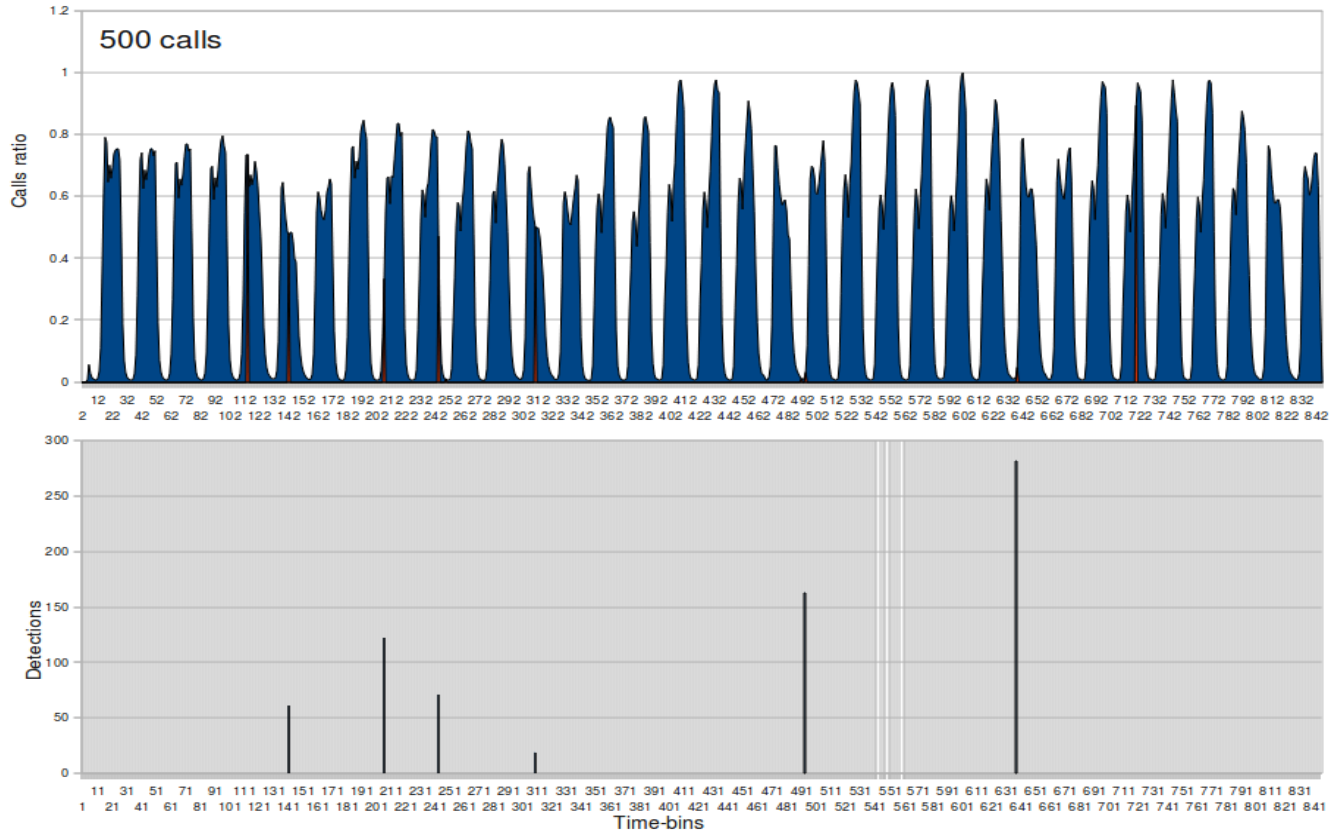


Figure B.6

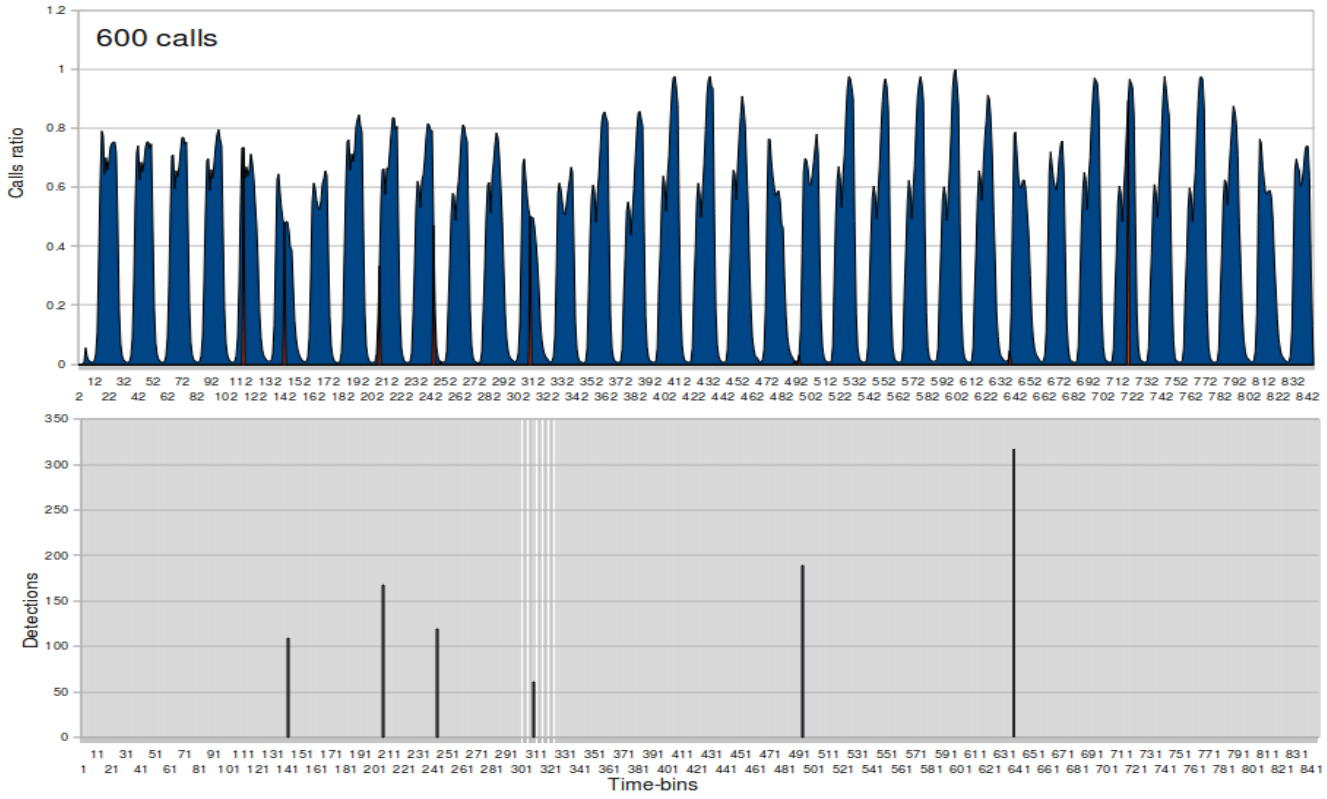


Figure B.7

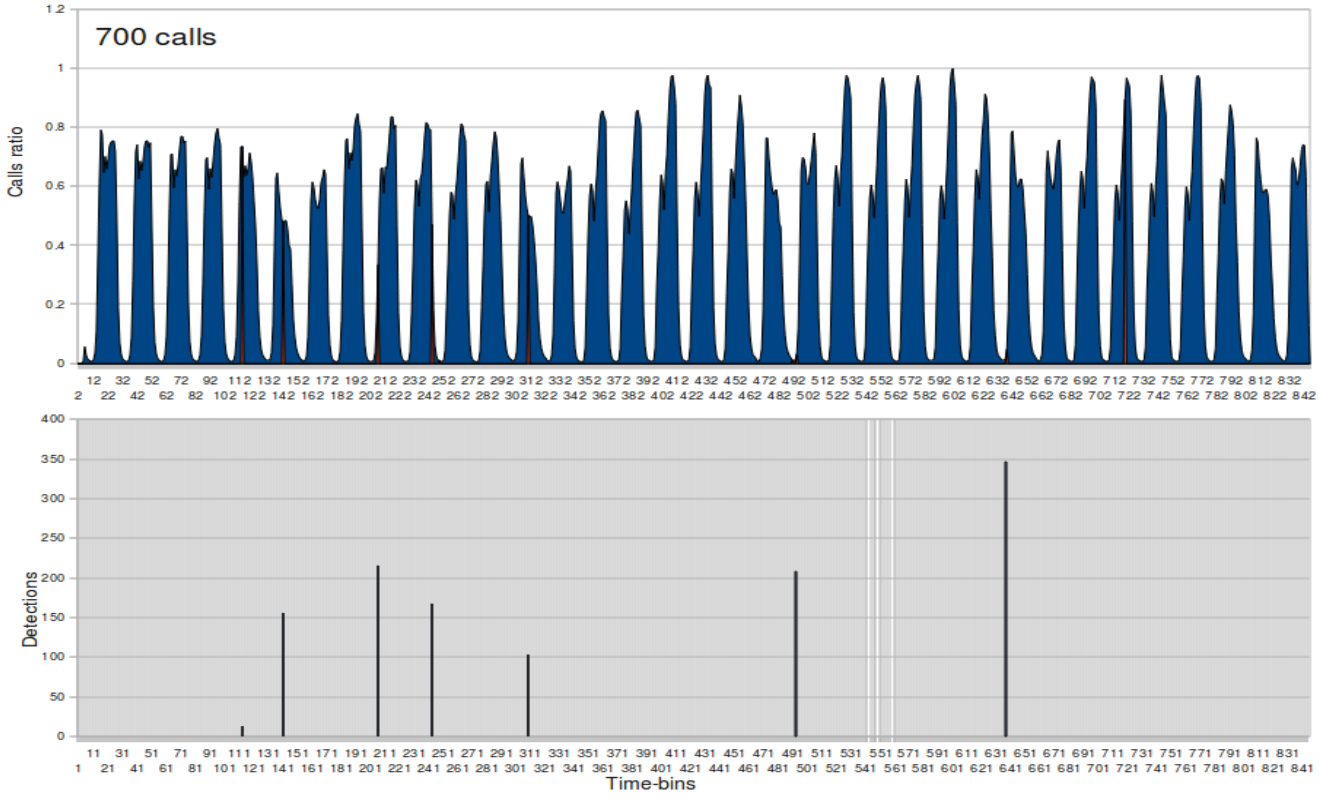


Figure B.8

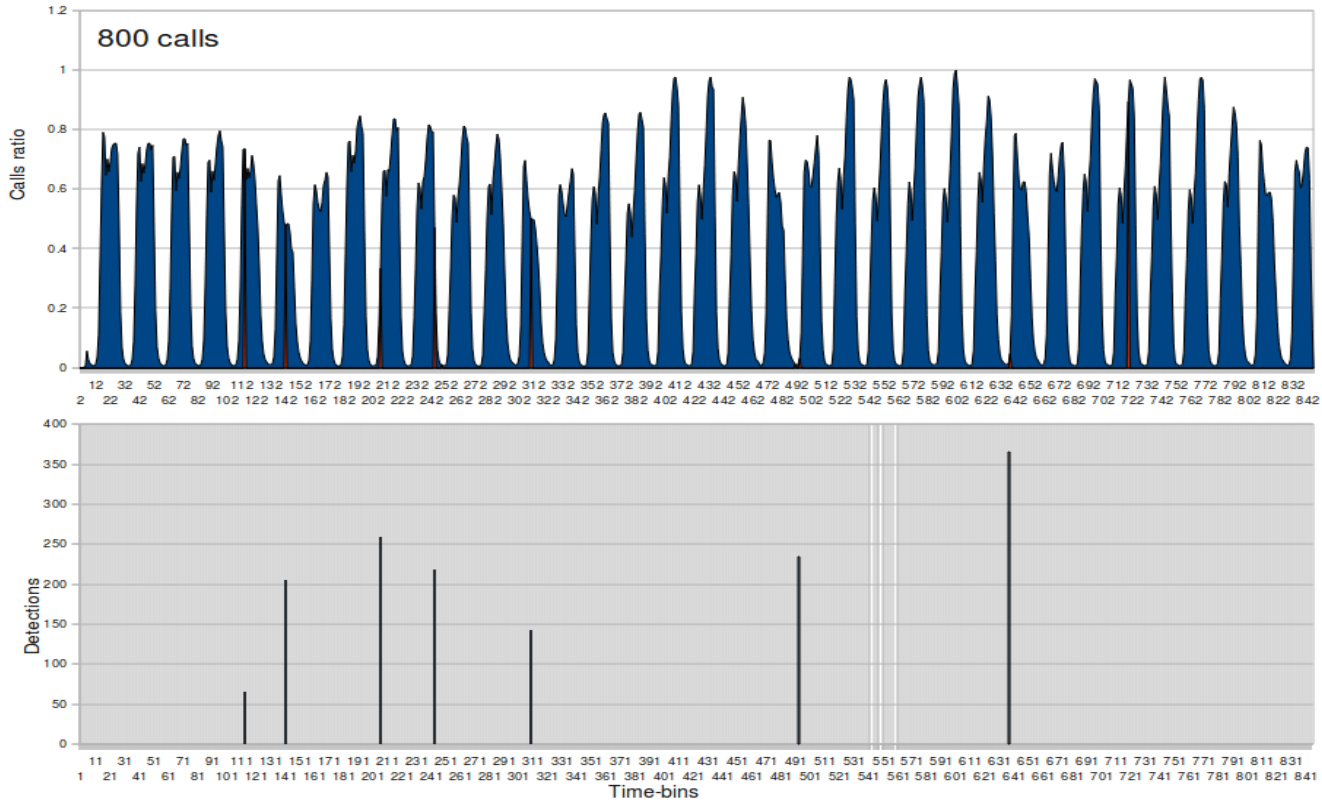


Figure B.9

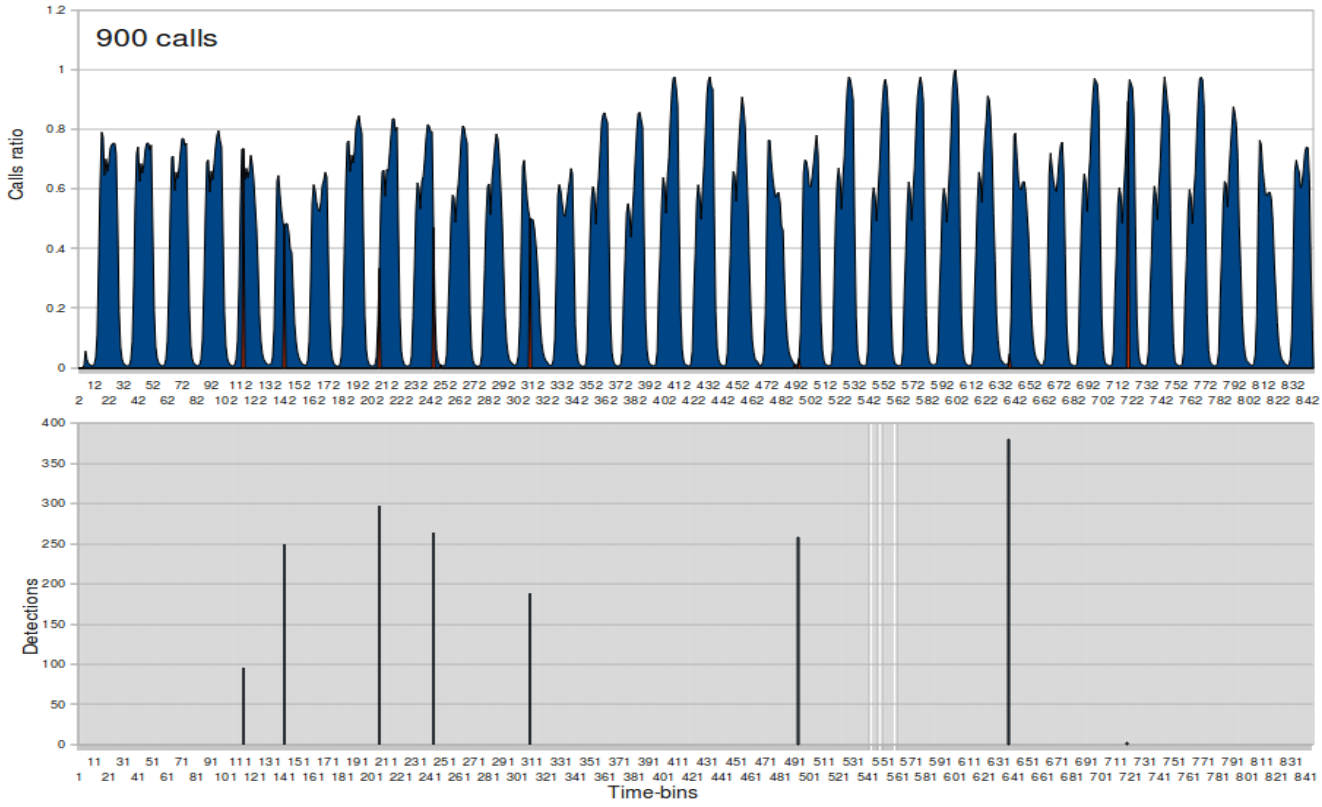
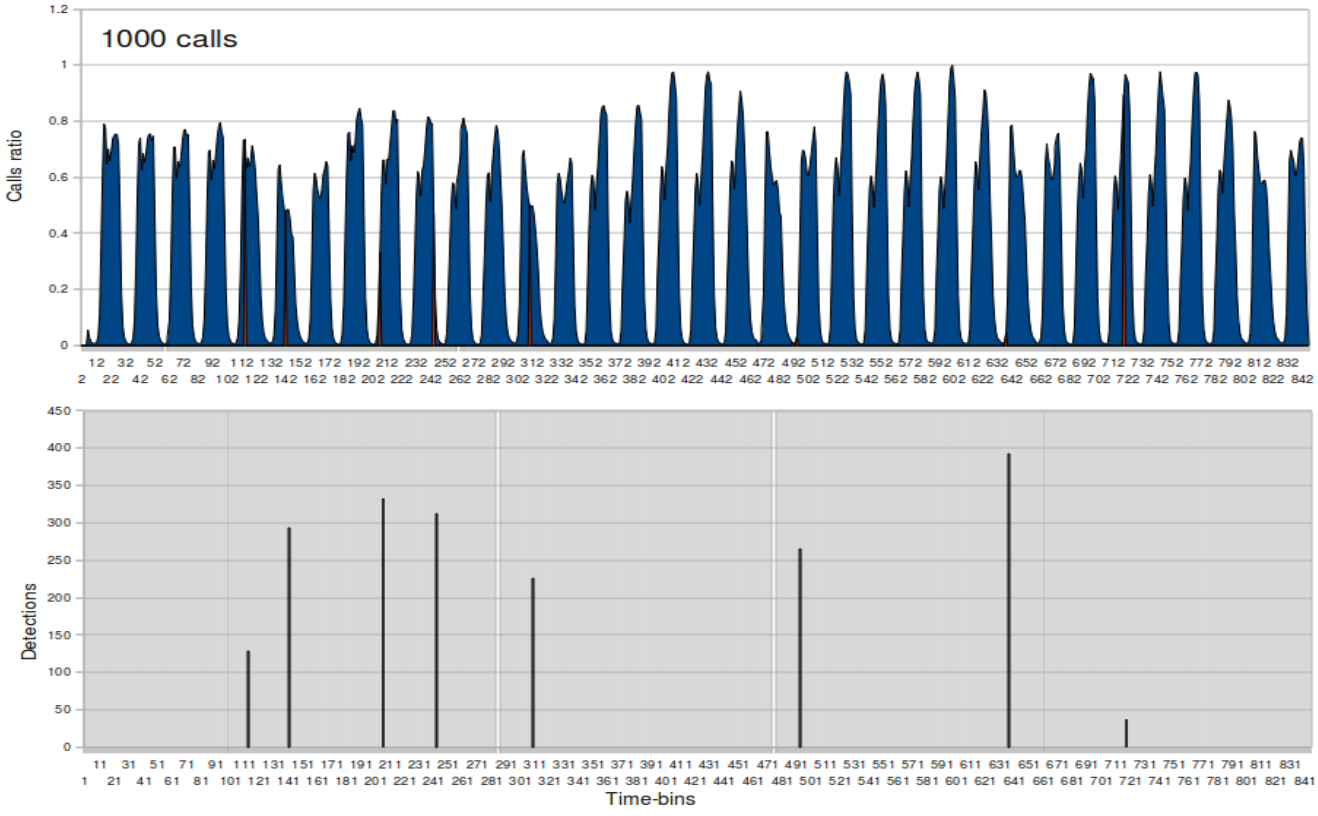


Figure B.10



Ratio of anomalous bins detected and responsible URIs recognized (upon the total of them) with “single threshold-single number of PCs” analysis (anomaly height: 1000 calls)

Figure B.11

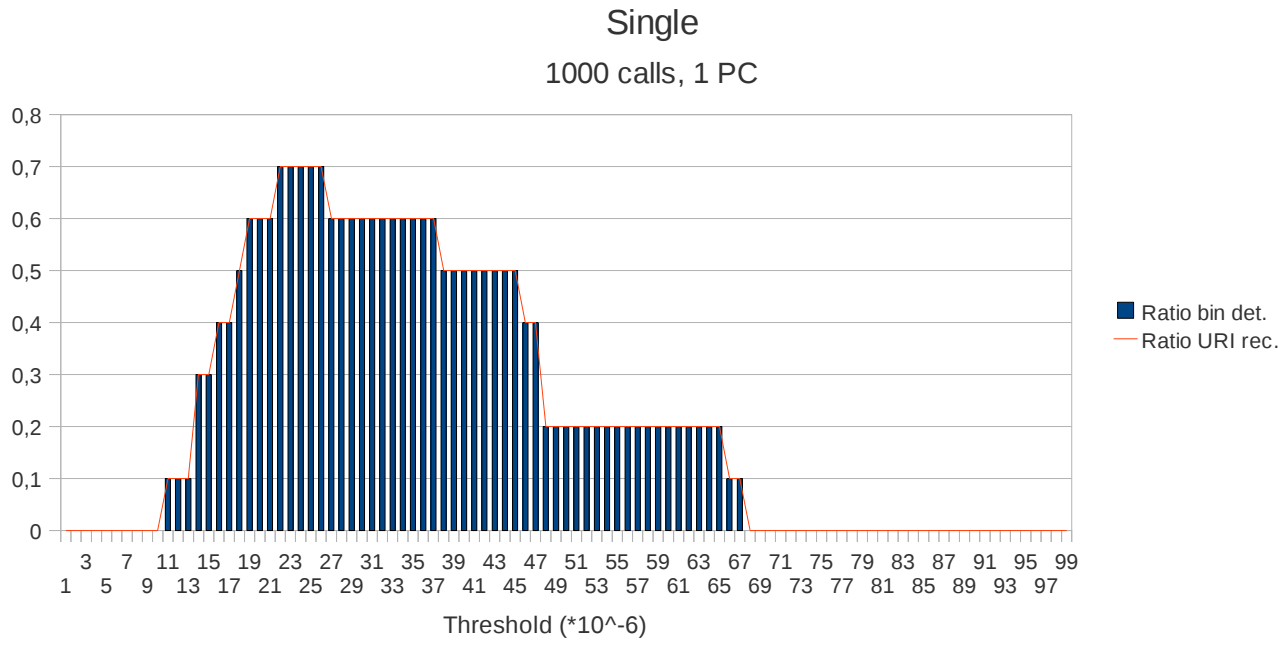


Figure B.12

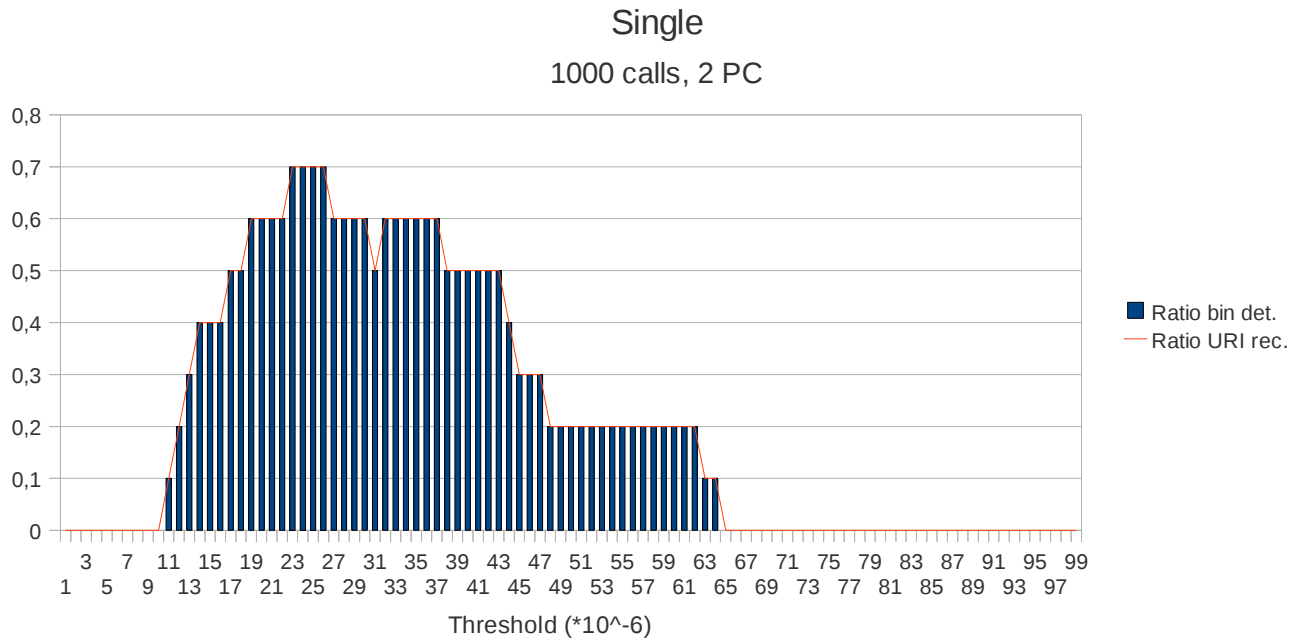


Figure B.13

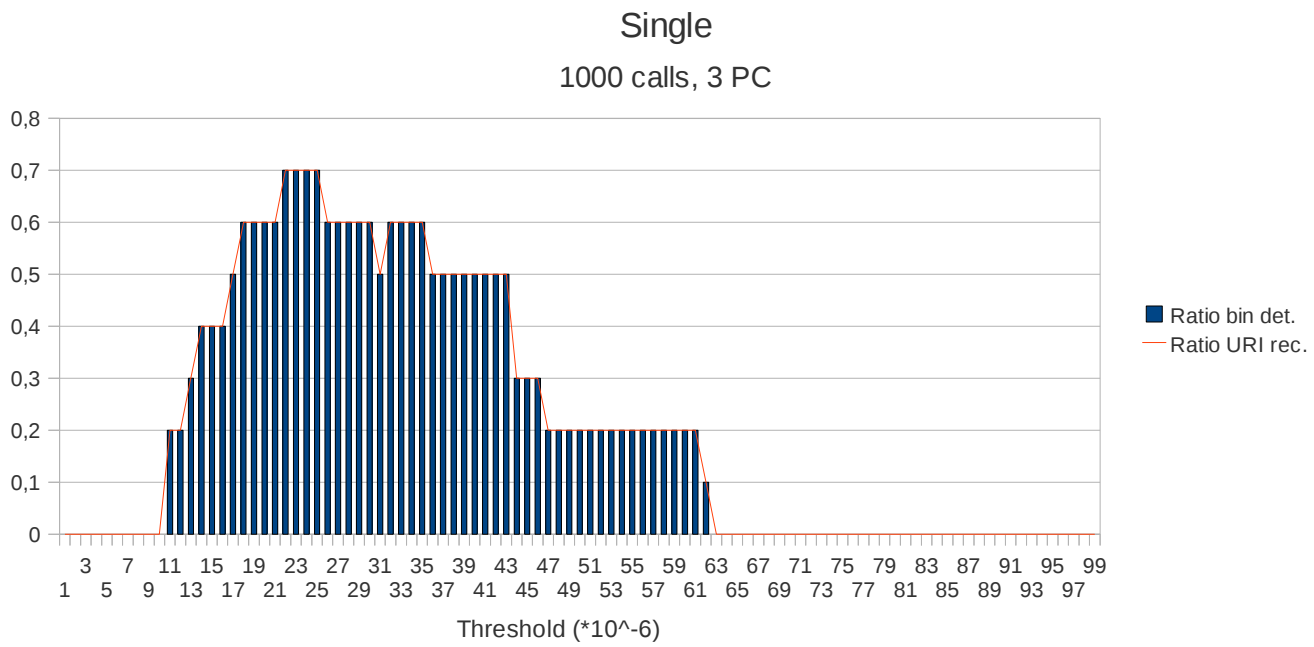


Figure B.14

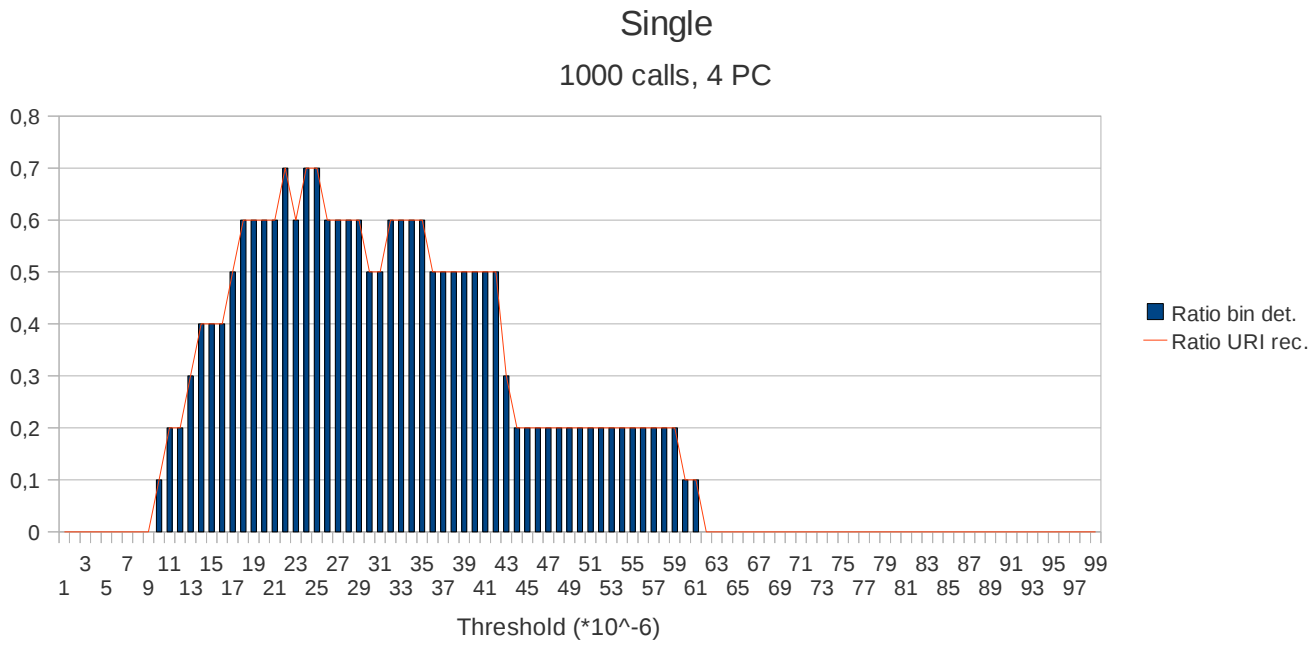


Figure B.15

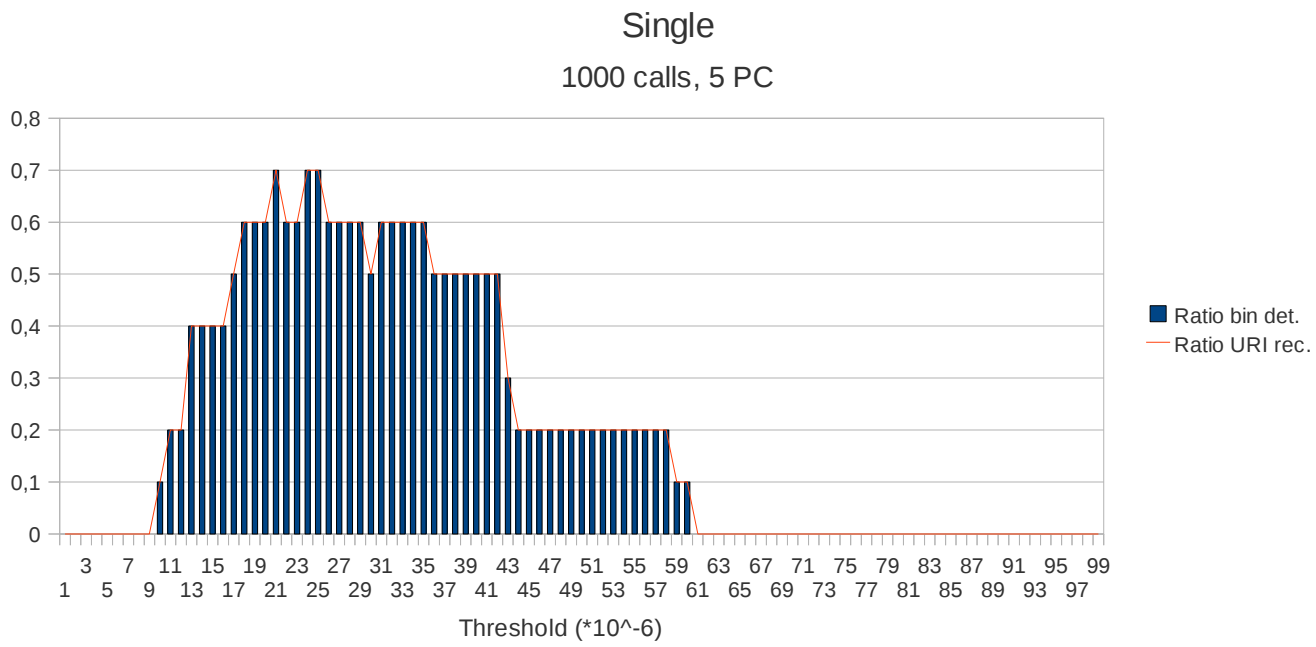


Figure B.16

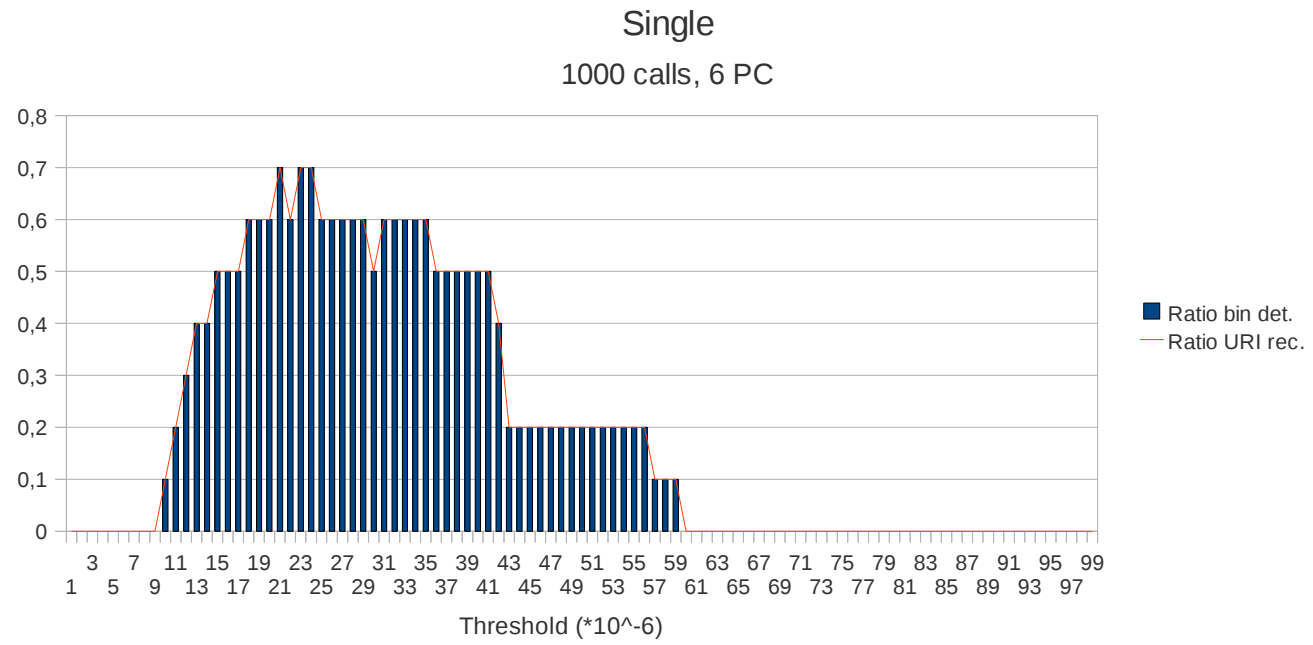


Figure B.17

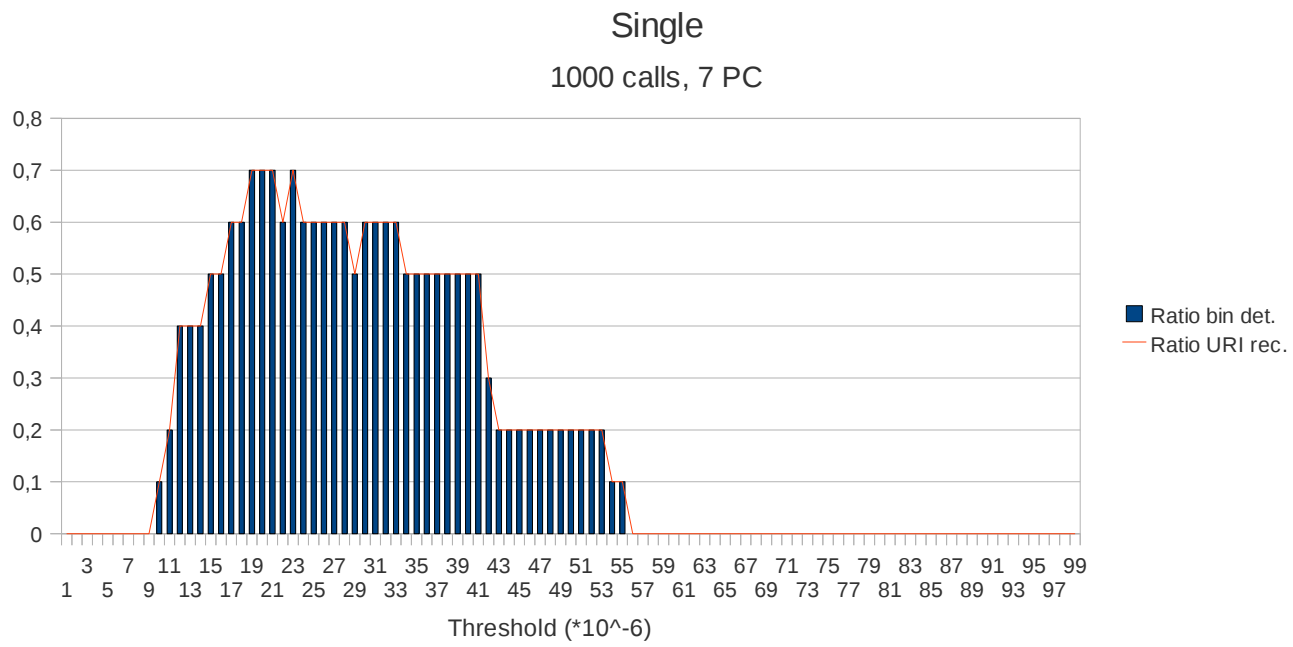


Figure B.18

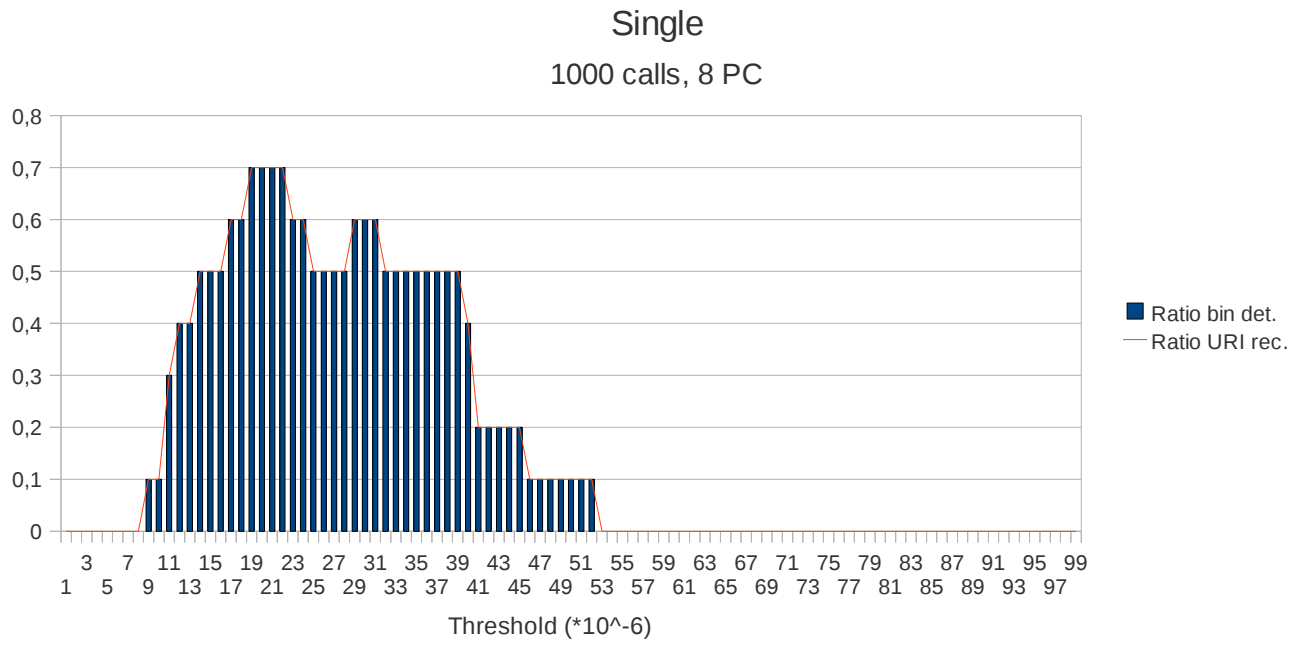


Figure B.19

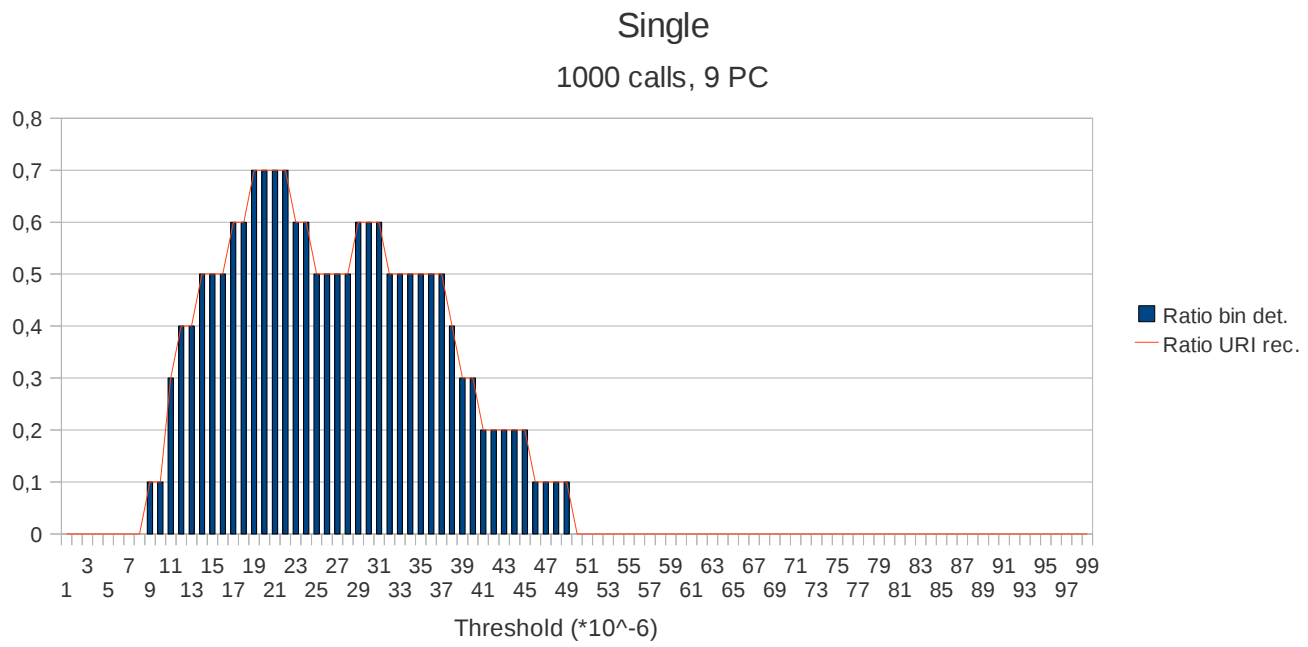
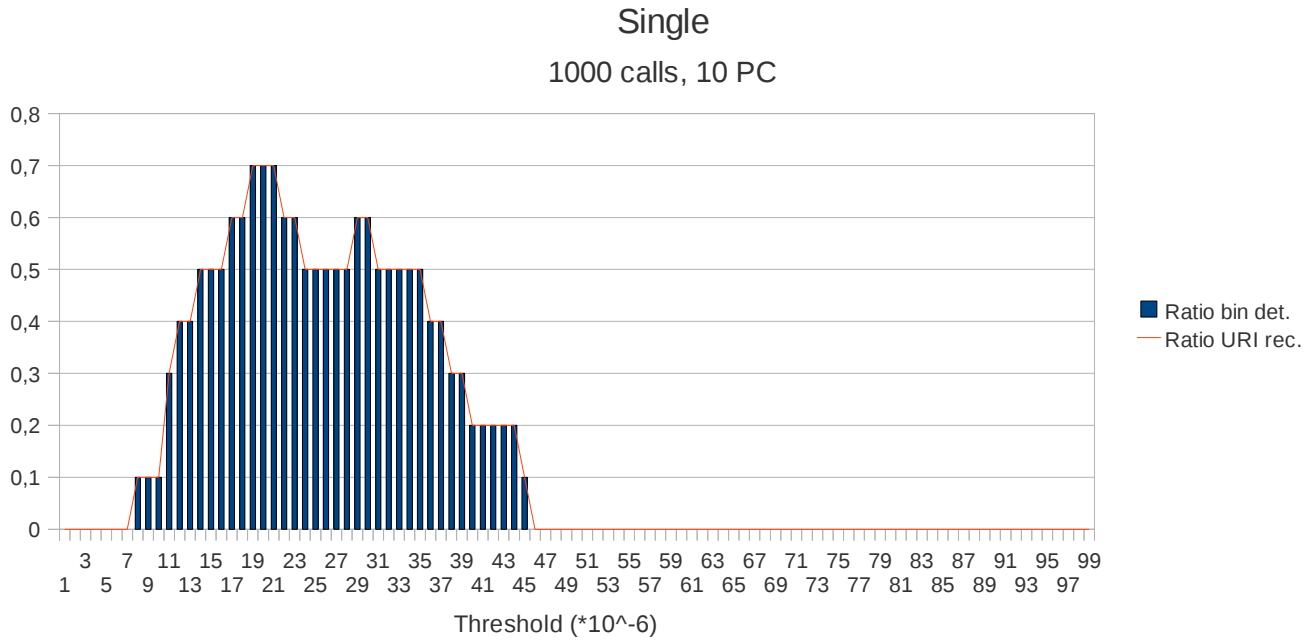


Figure B.20



Appendix C

(Rectangular artificial anomalies, width 5)

Number of detections for each time-bin in a "100 thresholds-10 numbers of PCs" analysis VS ratio of global calls per time-bin on their maximum value

Figure C.1

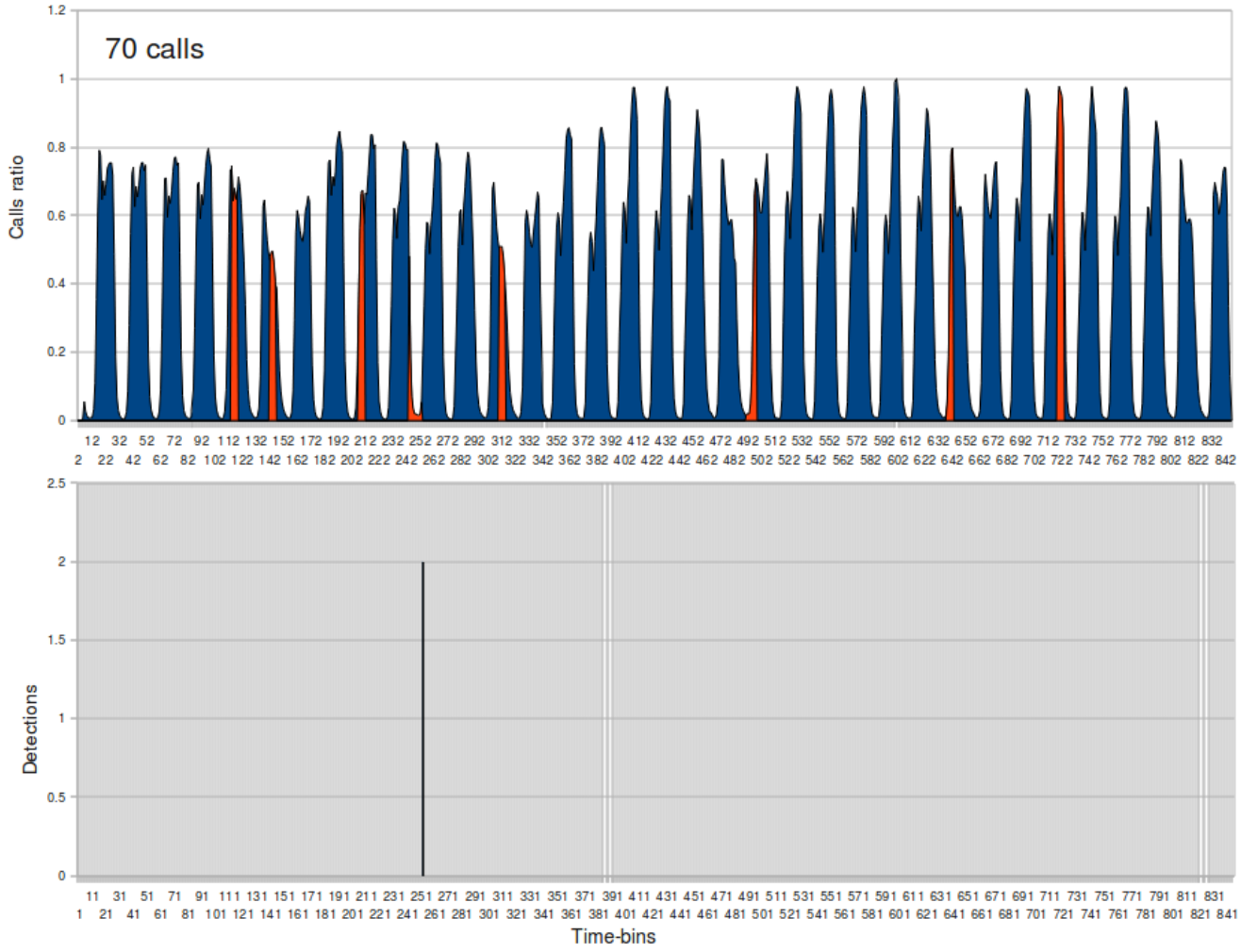


Figure C.2

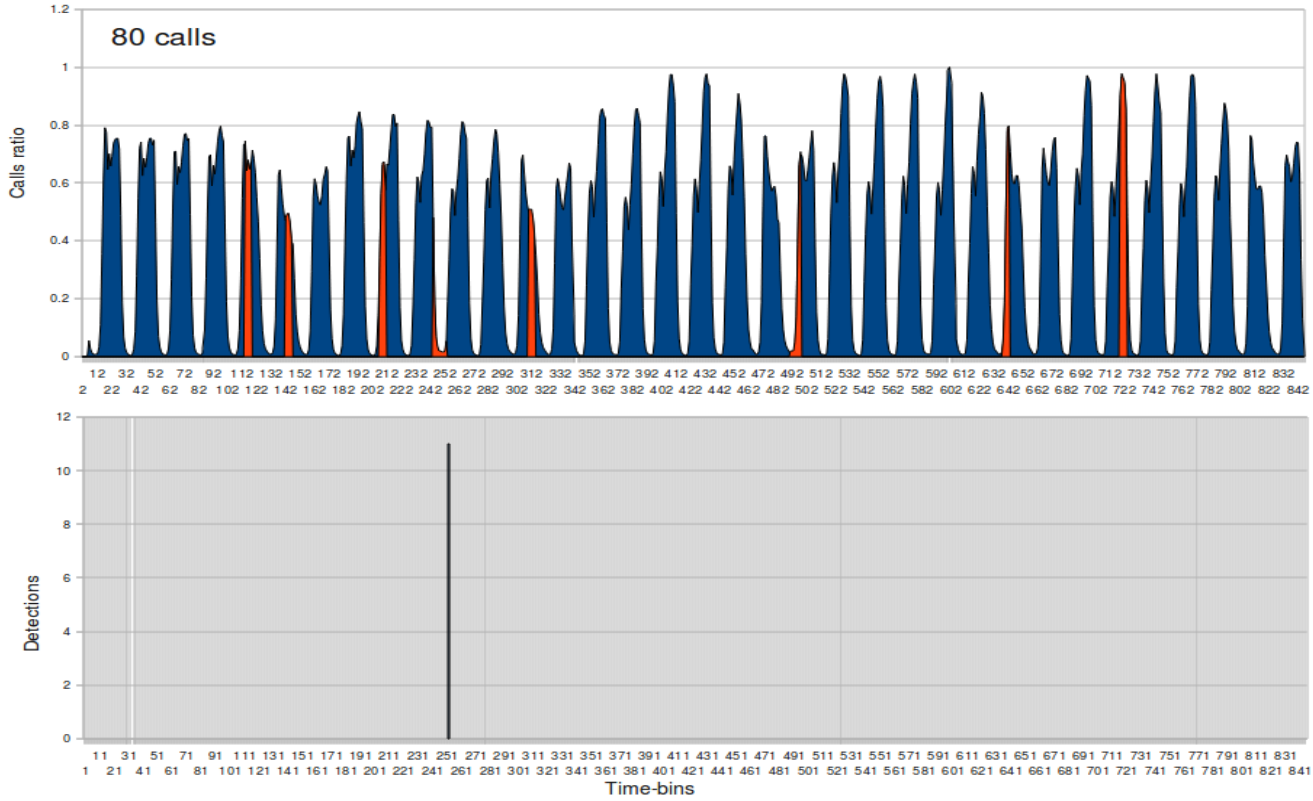


Figure C.3

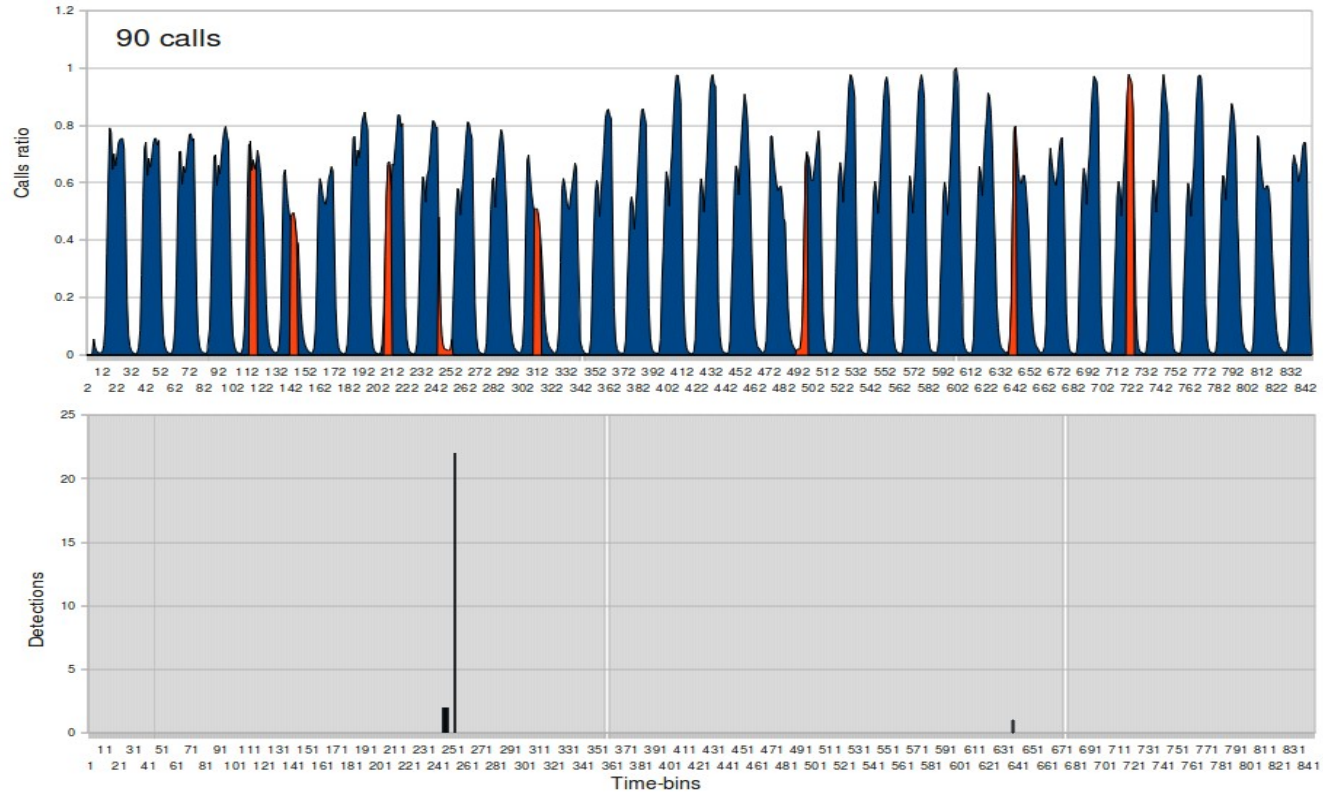


Figure C.4

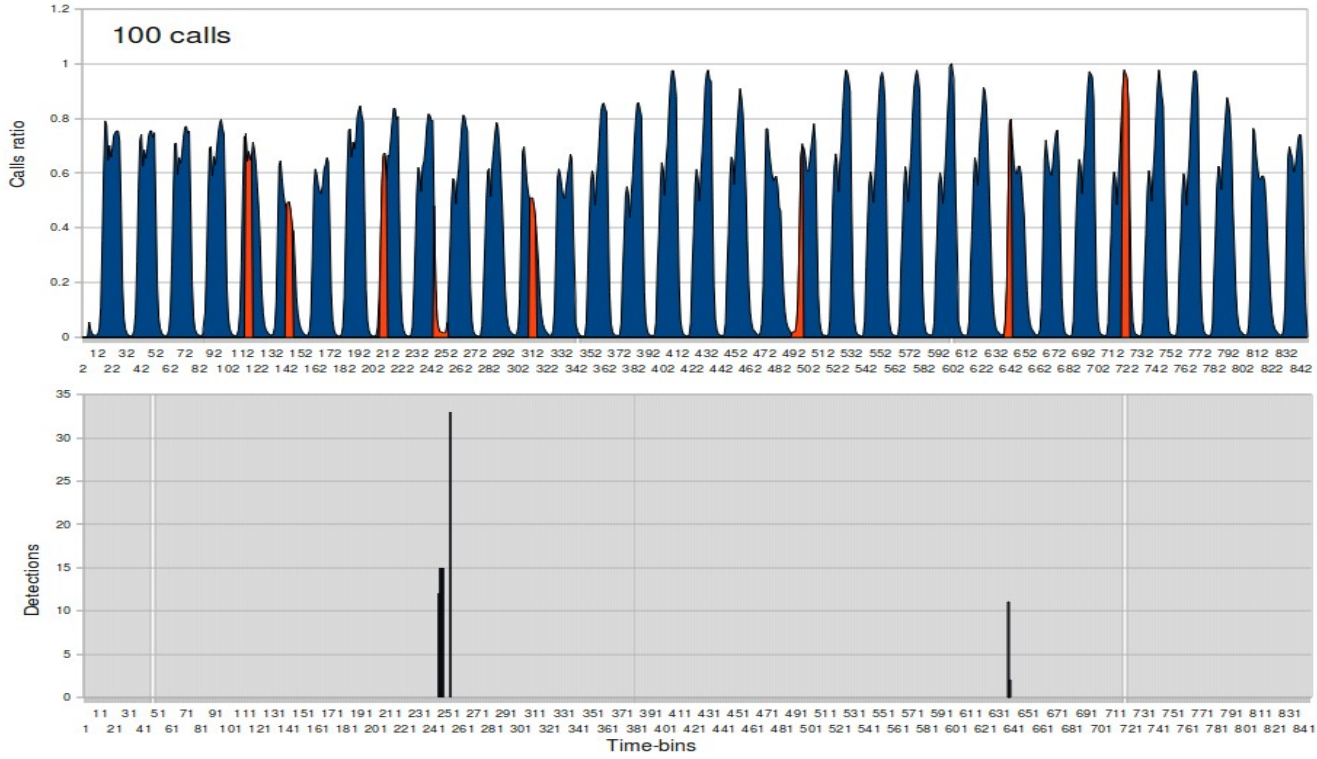


Figure C.5

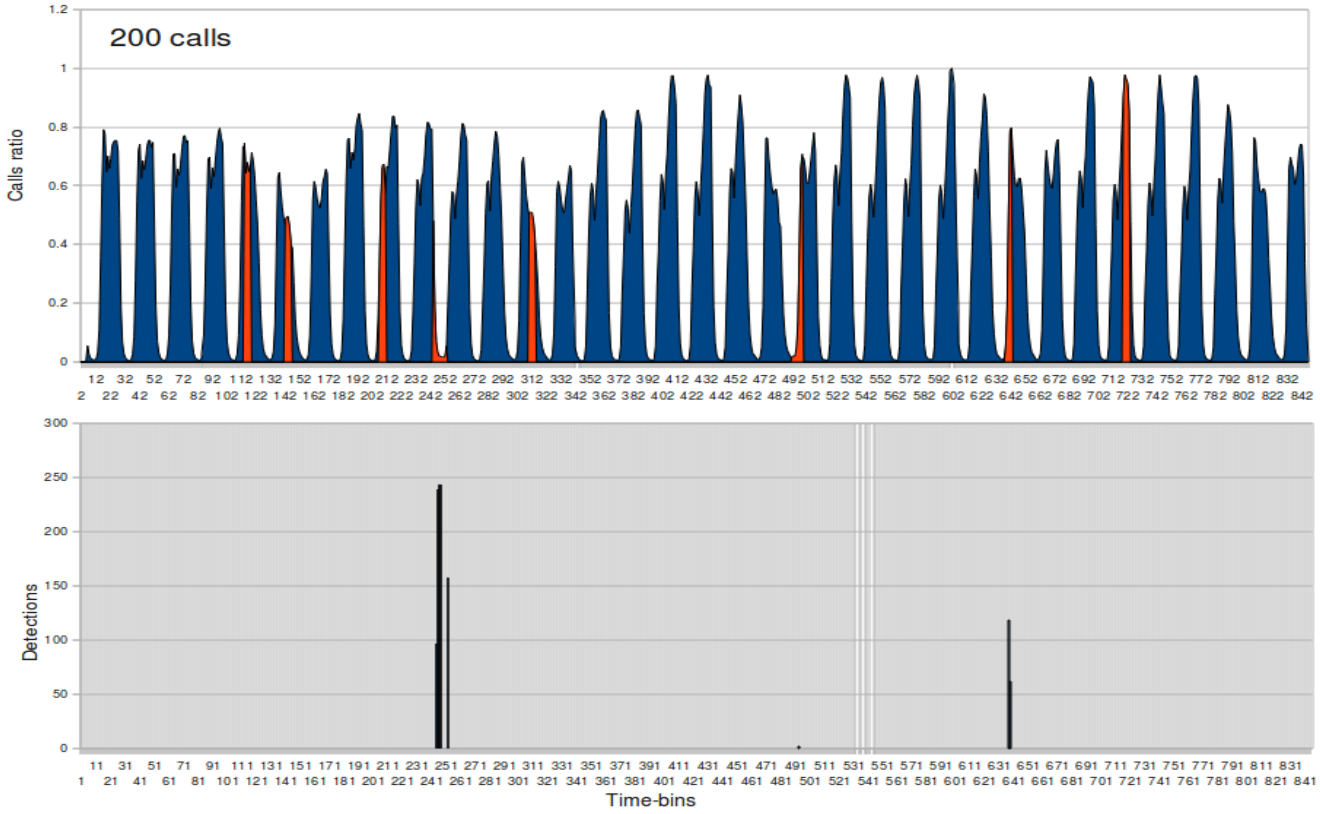


Figure C.6

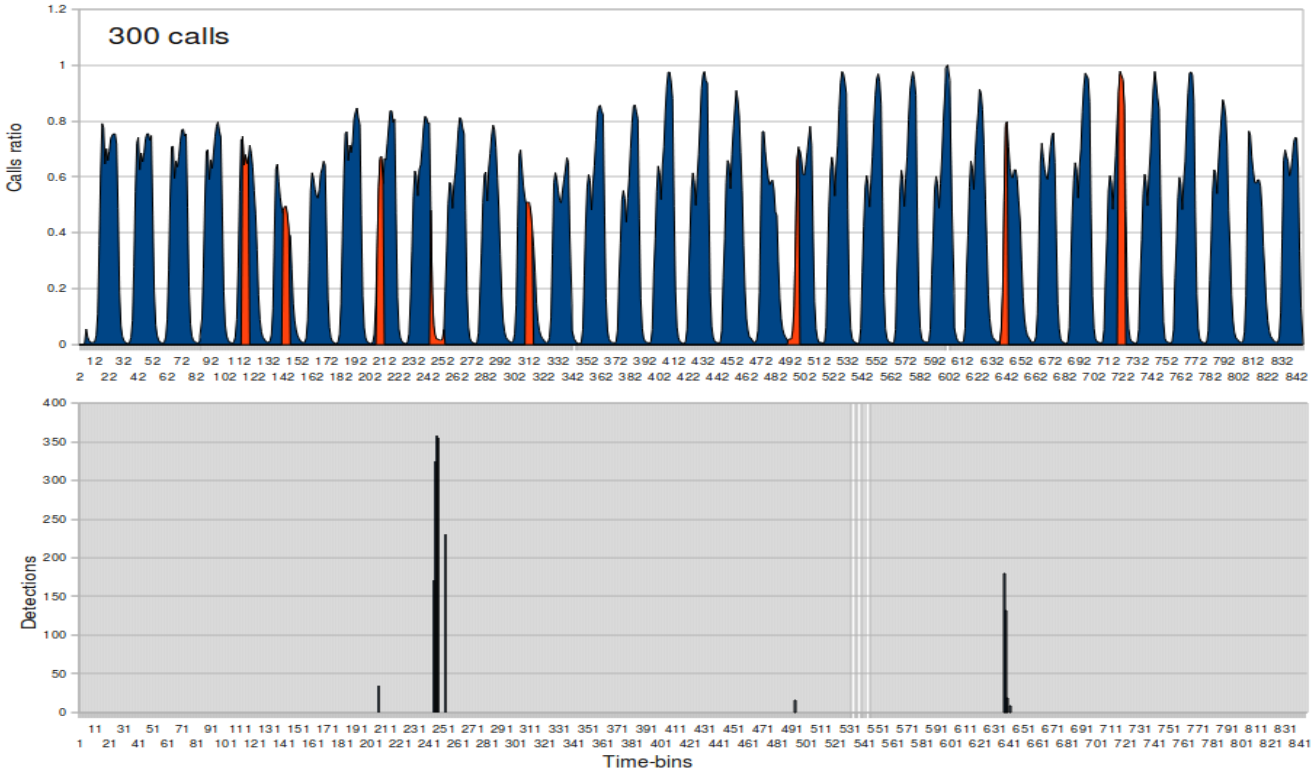


Figure C.7

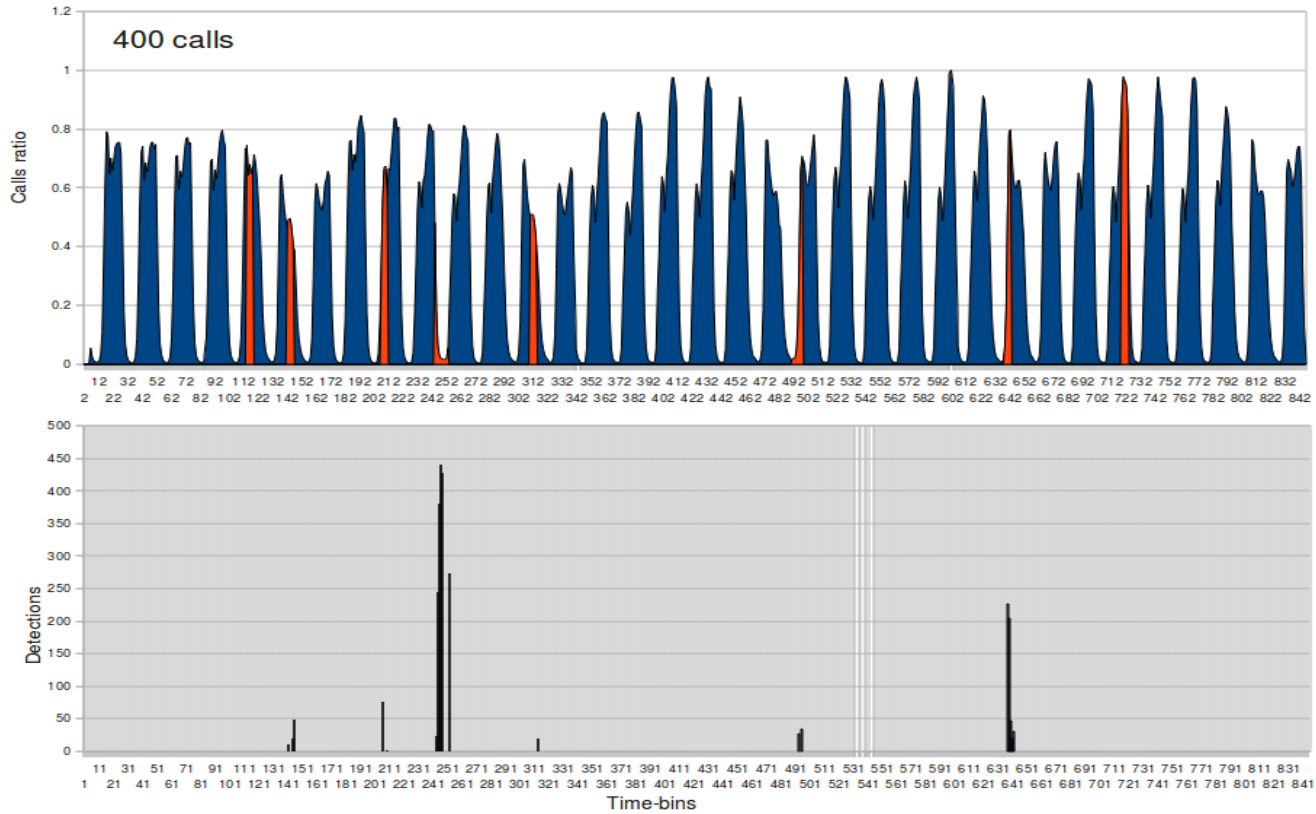


Figure C.8

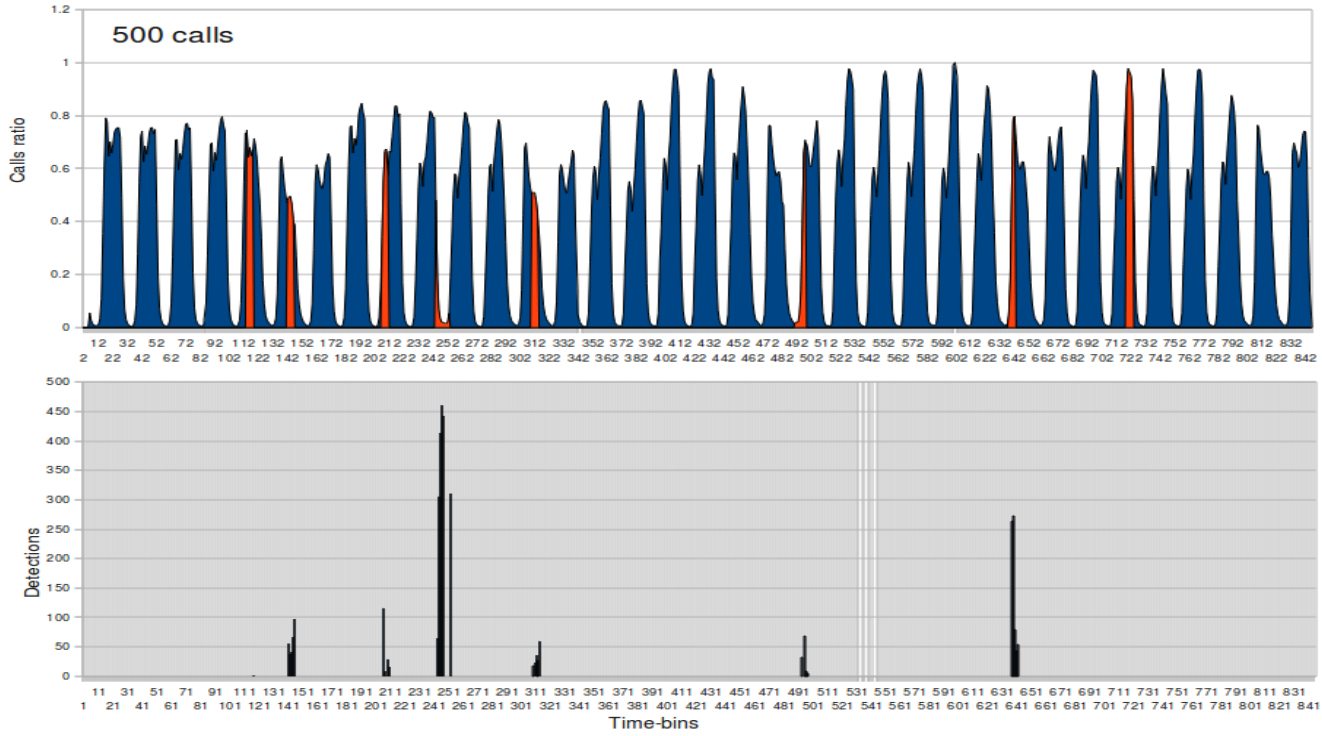


Figure C.9

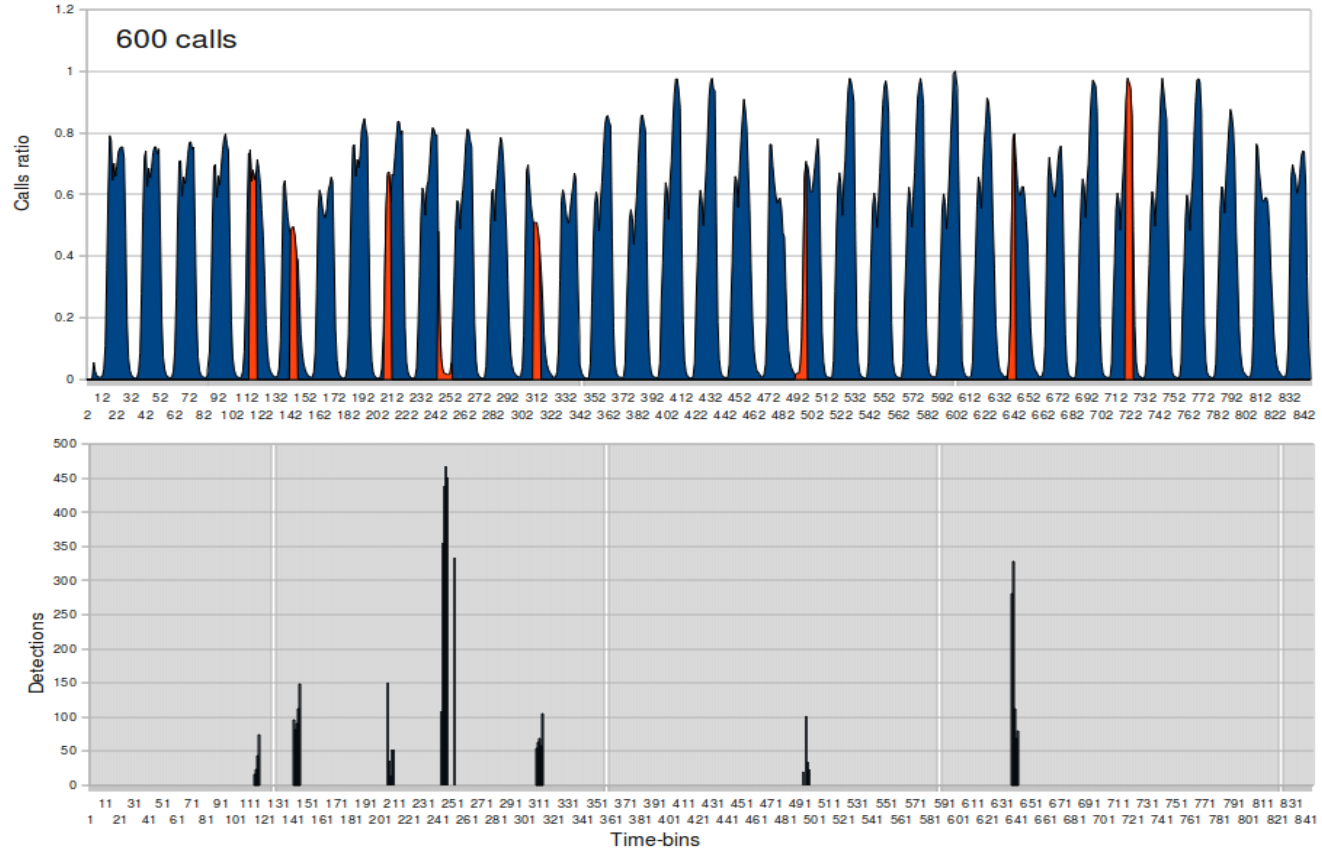


Figure C.10

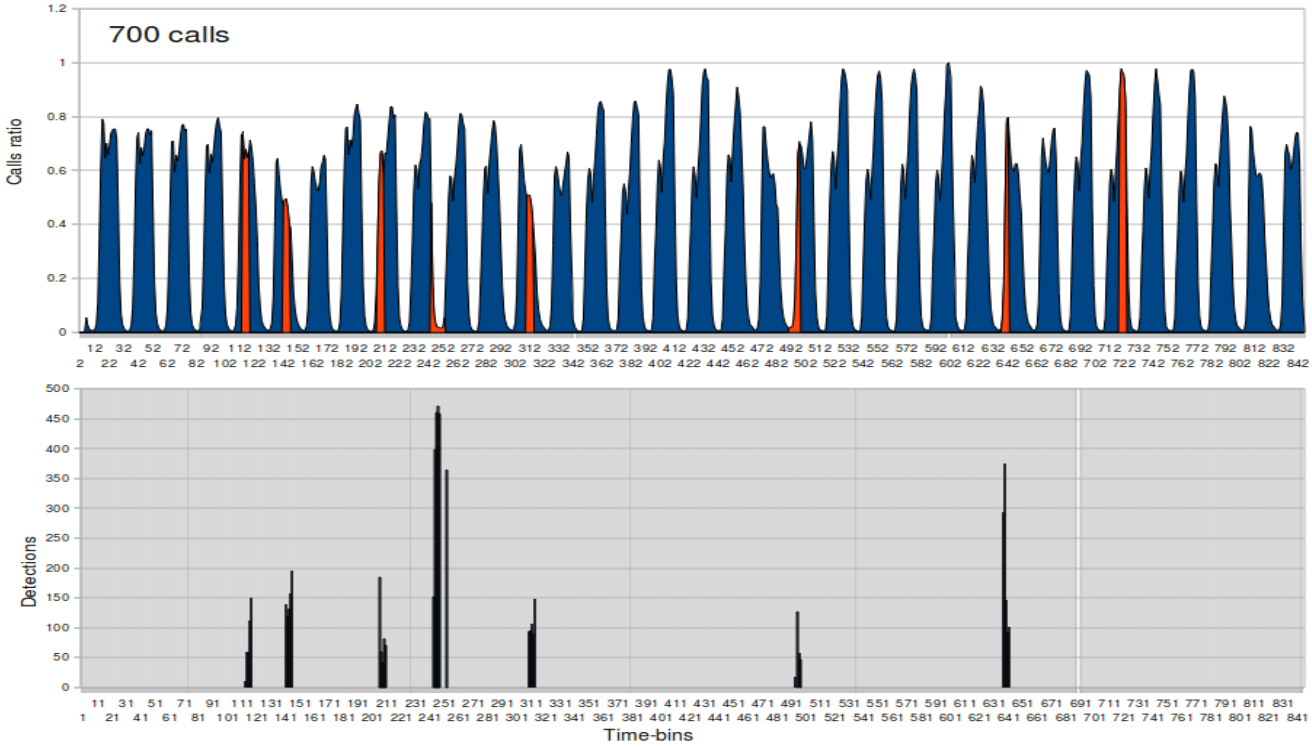


Figure C.11

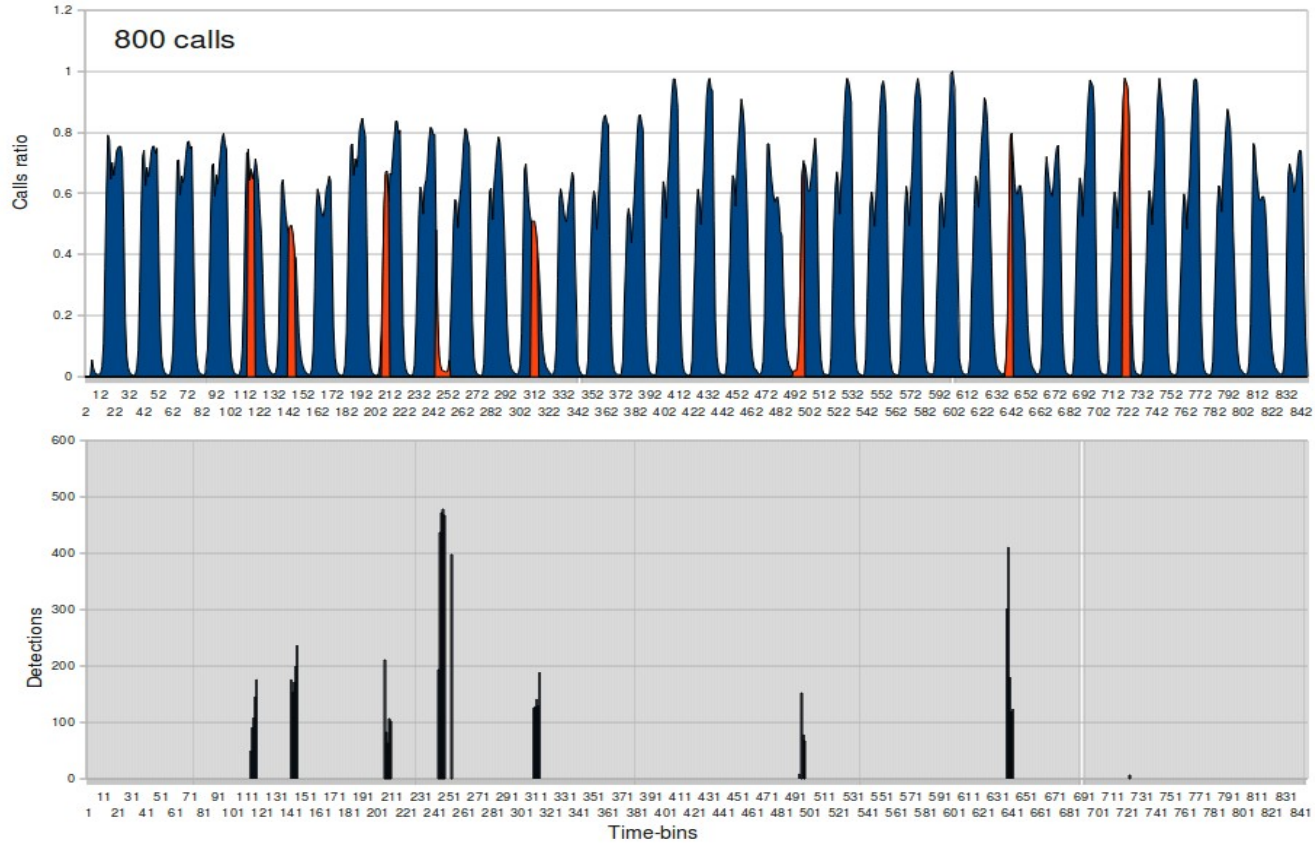


Figure C.12

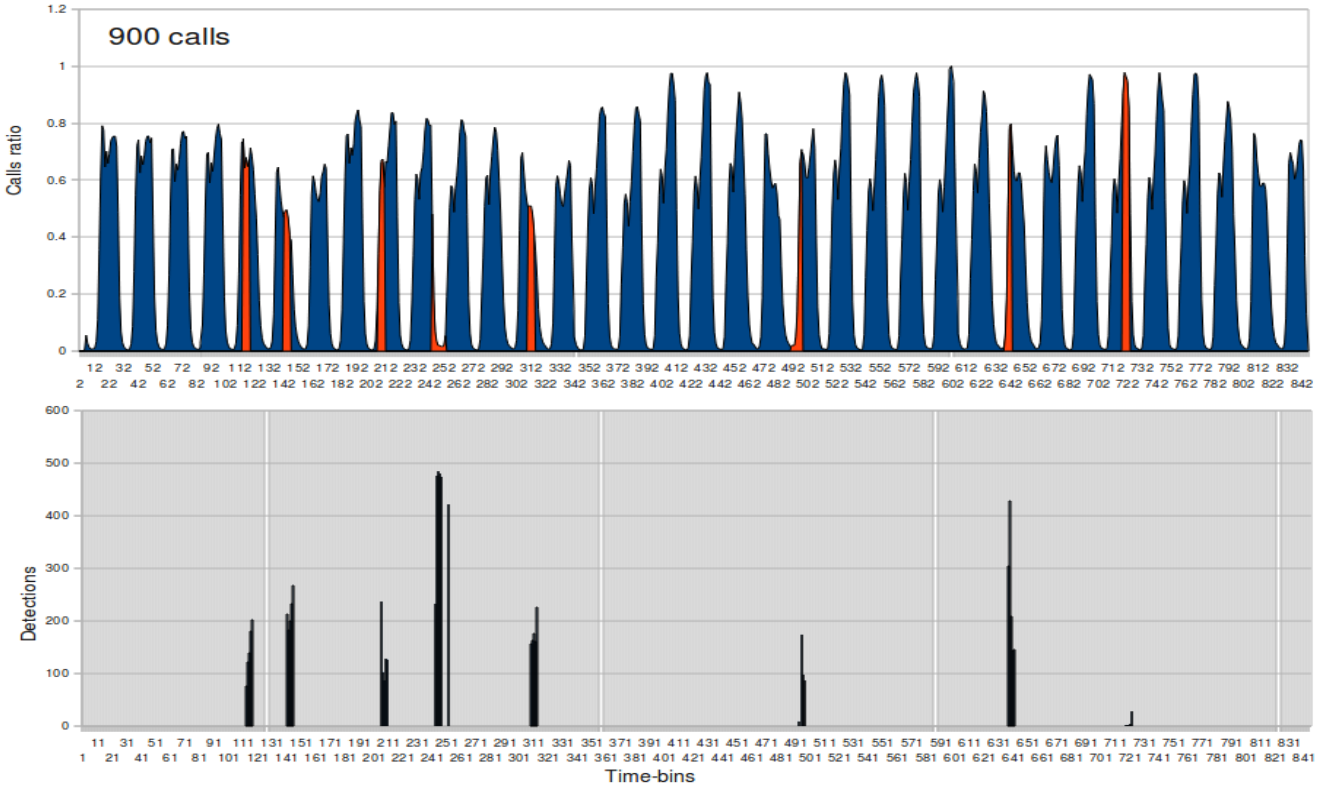


Figure C.13

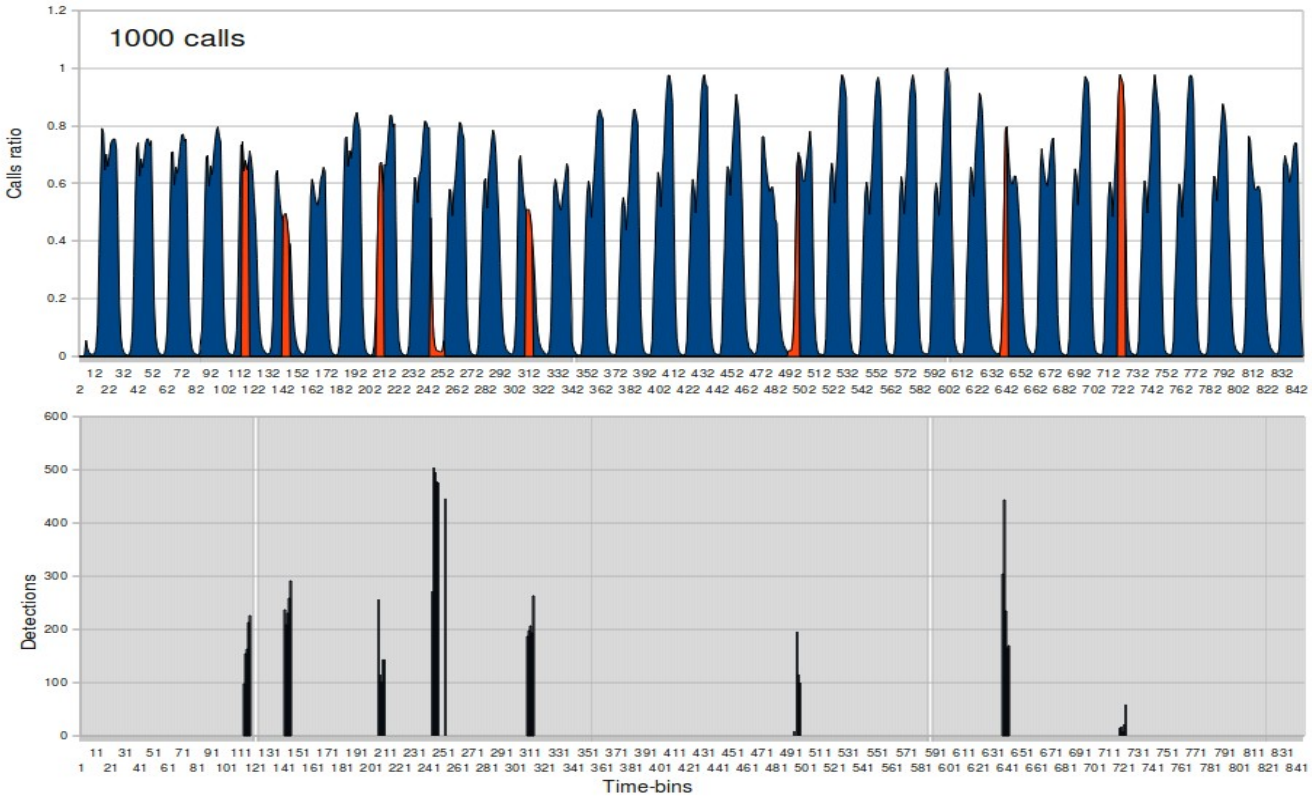
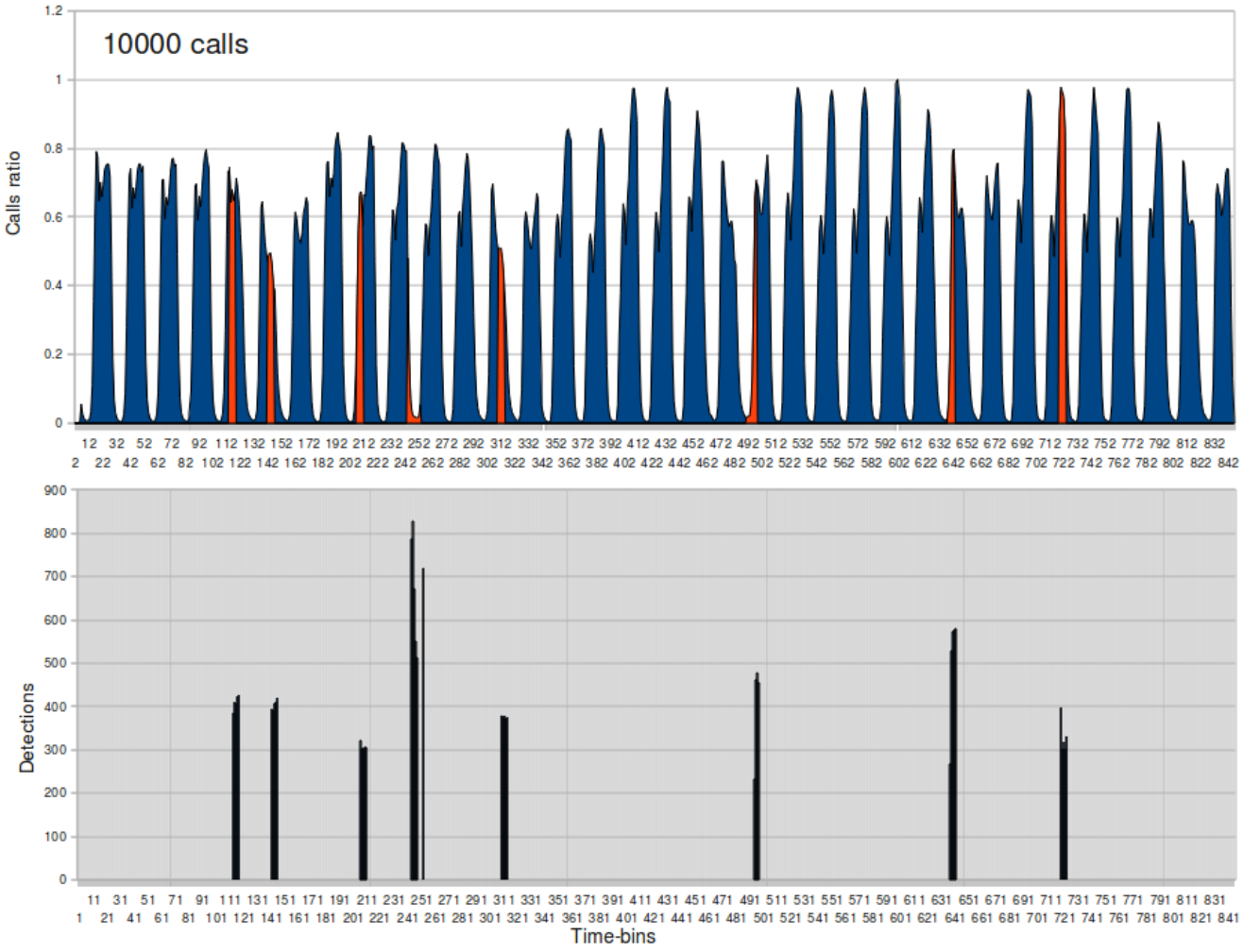


Figure C.14



Ratio of anomalous bins detected and responsible URIs recognized (upon the total of them) with “single threshold-single number of PCs” analysis (anomaly height: 1000 calls)

Figure C.15

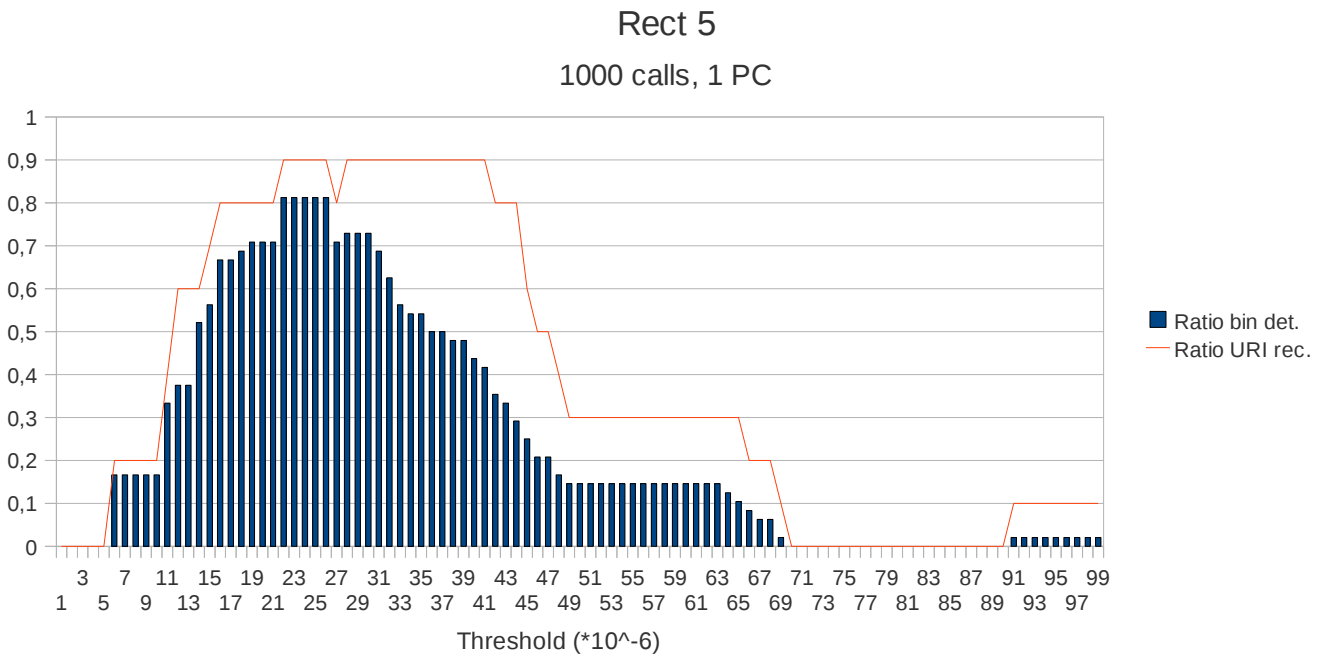


Figure C.16

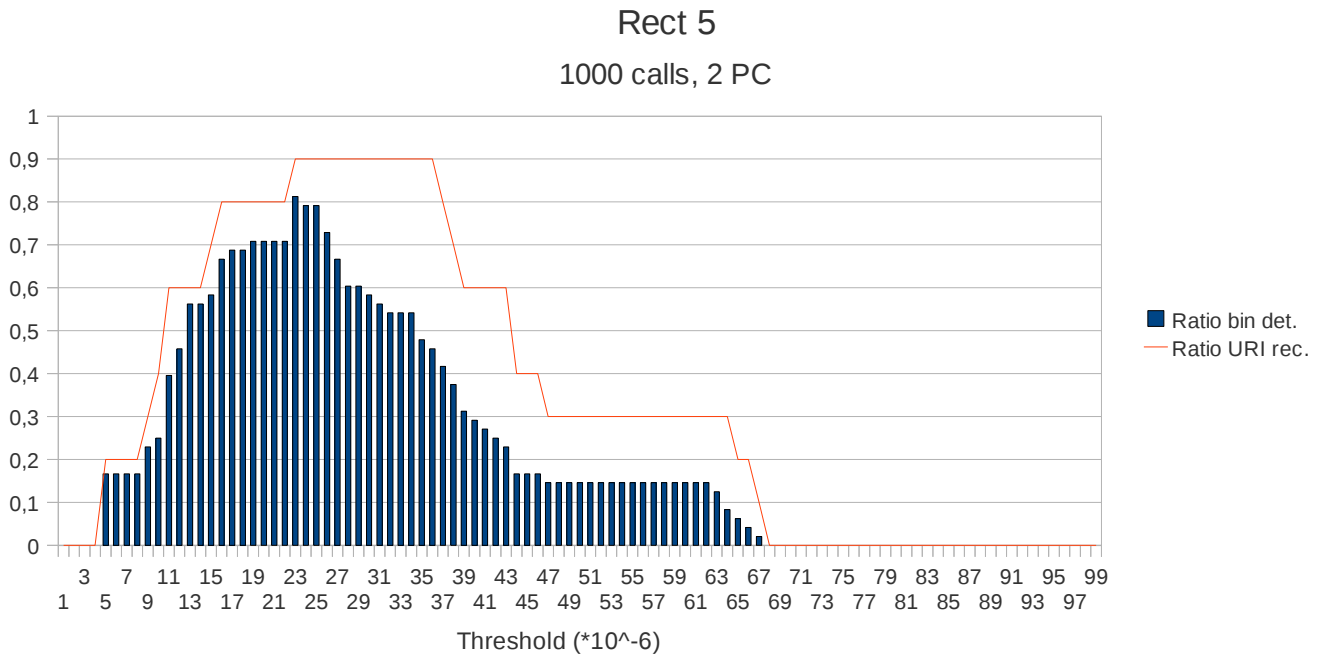


Figure C.17

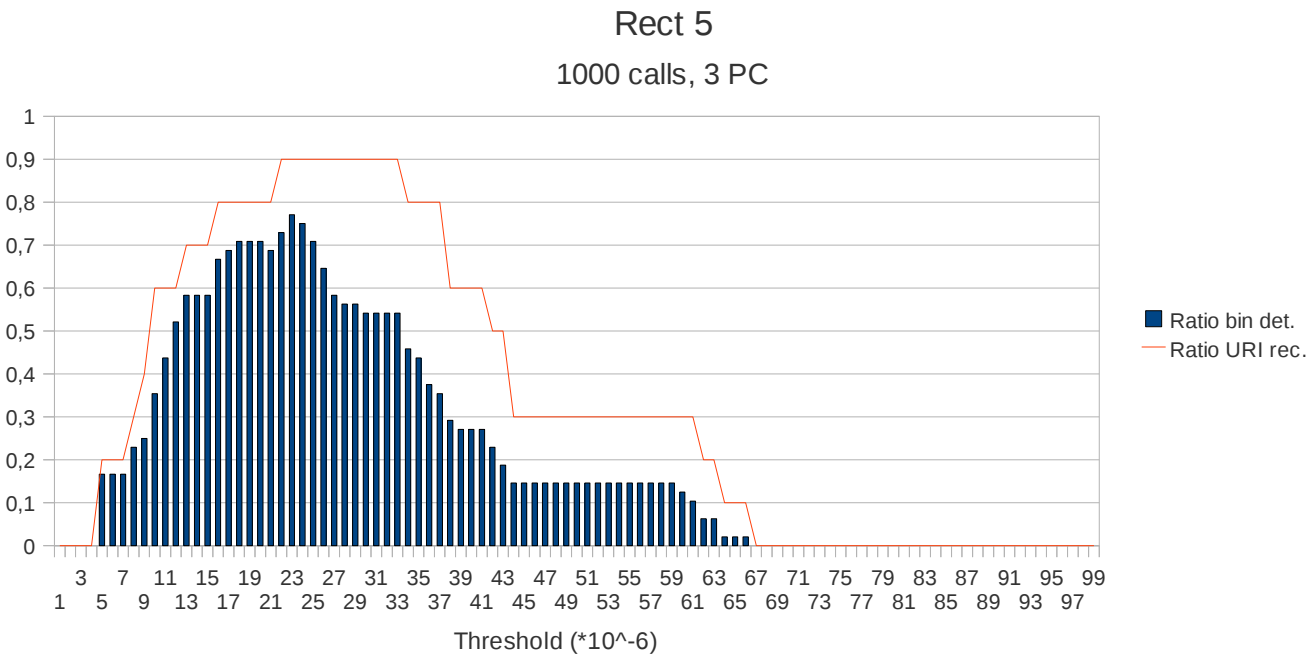


Figure C.18

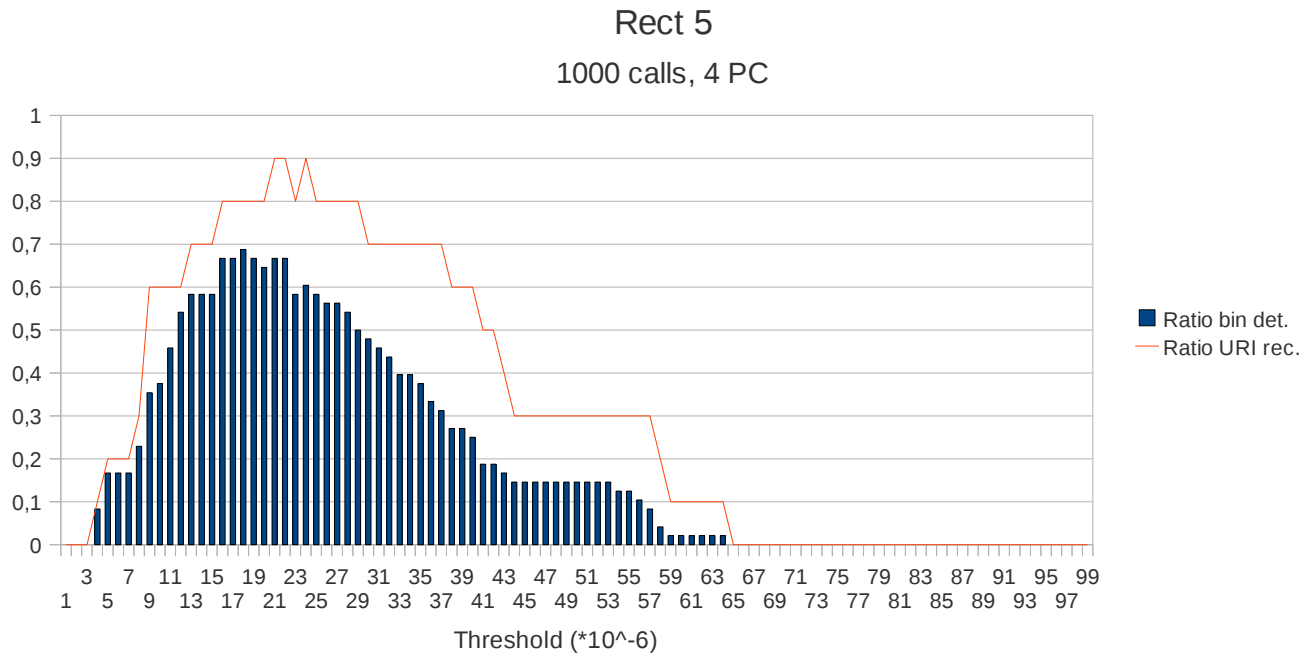


Figure C.19

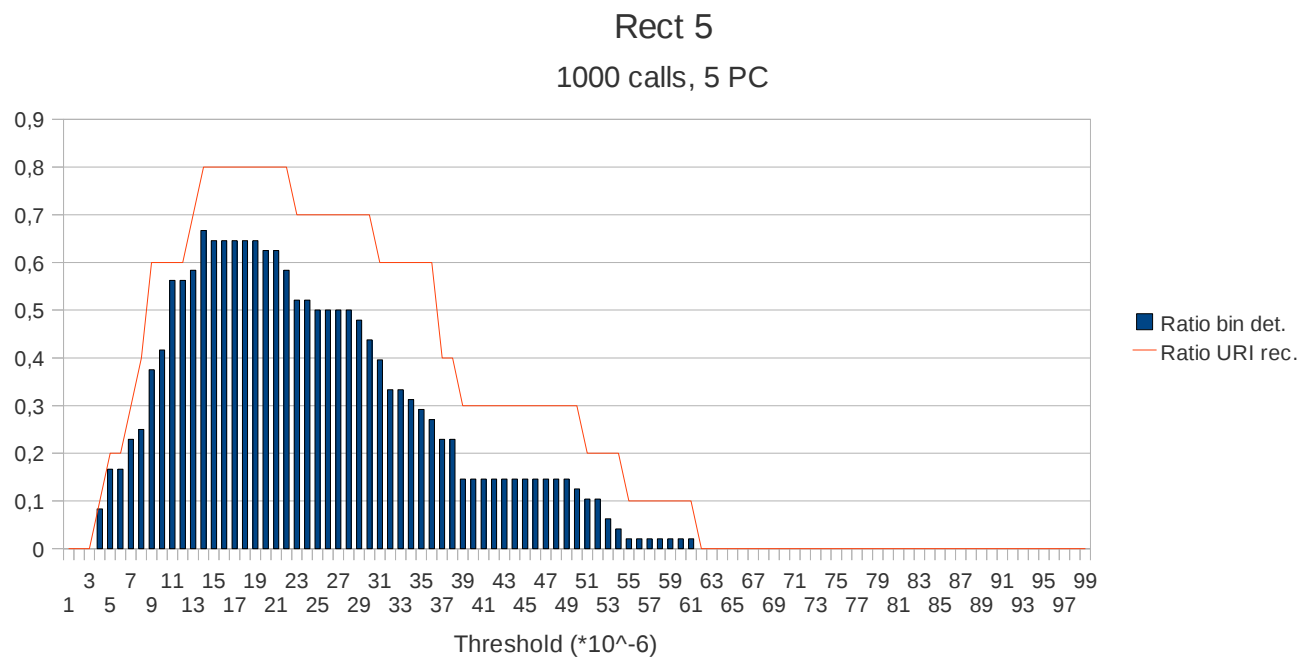


Figure C.20

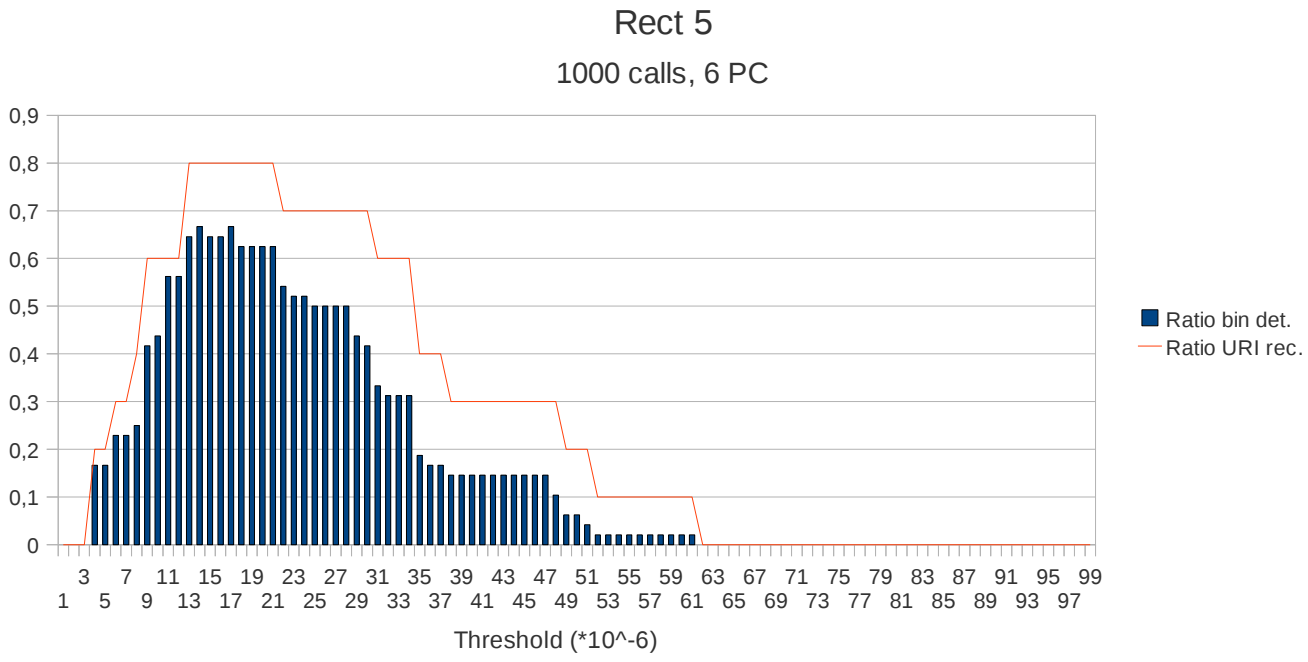


Figure C.21

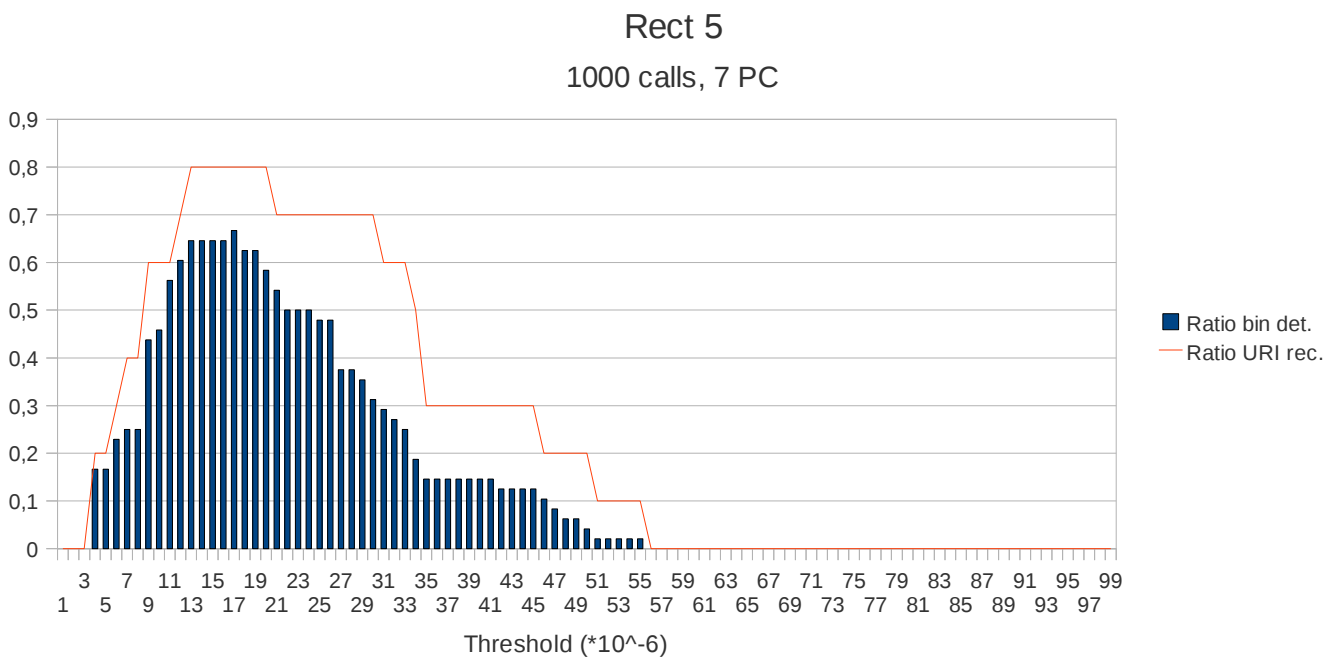


Figure C.22

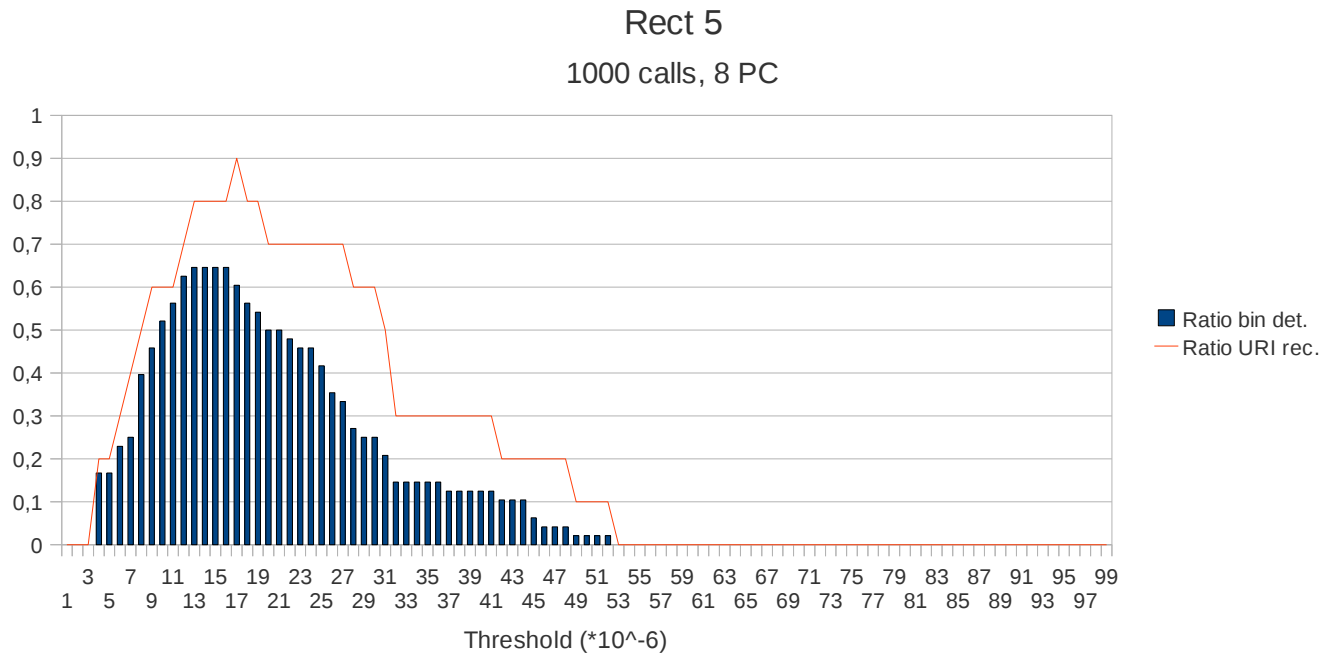


Figure C.23

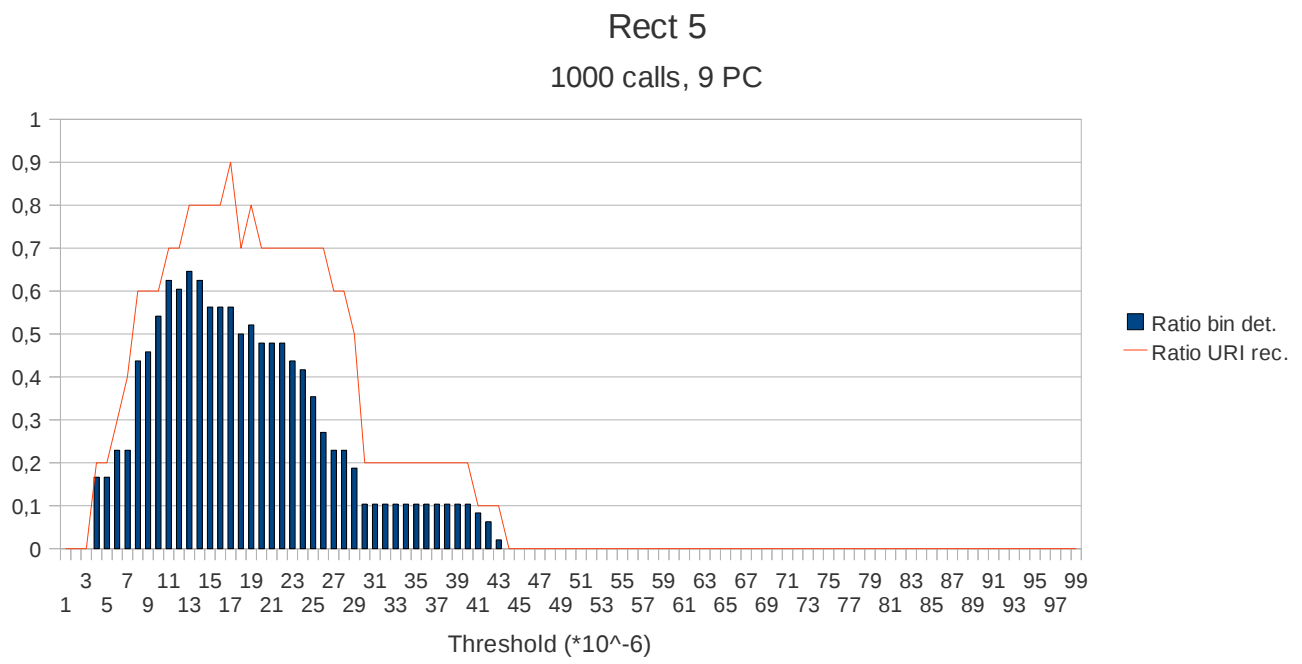
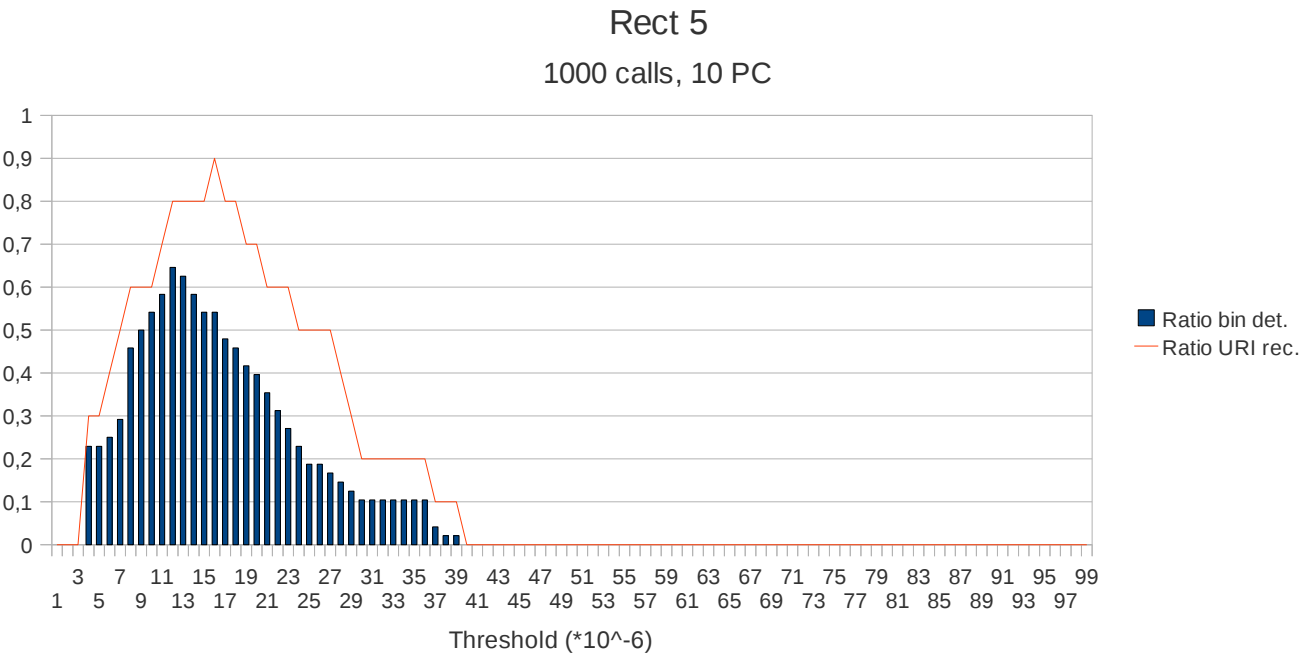


Figure C.24



Appendix D

(Rectangular artificial anomalies, width 10)

Number of detections for each time-bin in a "100 thresholds-10 numbers of PCs" analysis VS ratio of global calls per time-bin on their maximum value

Figure D.1

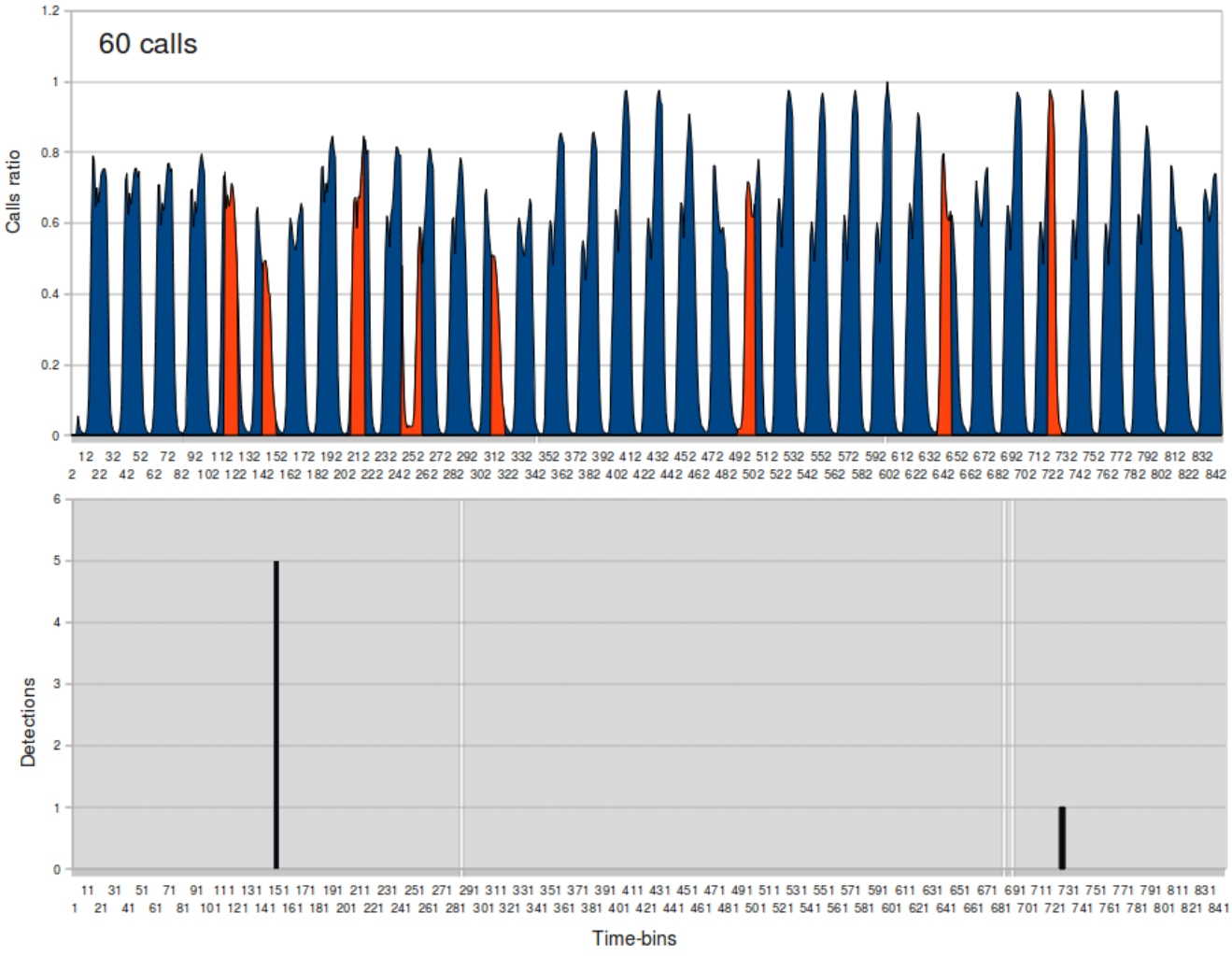


Figure D.2

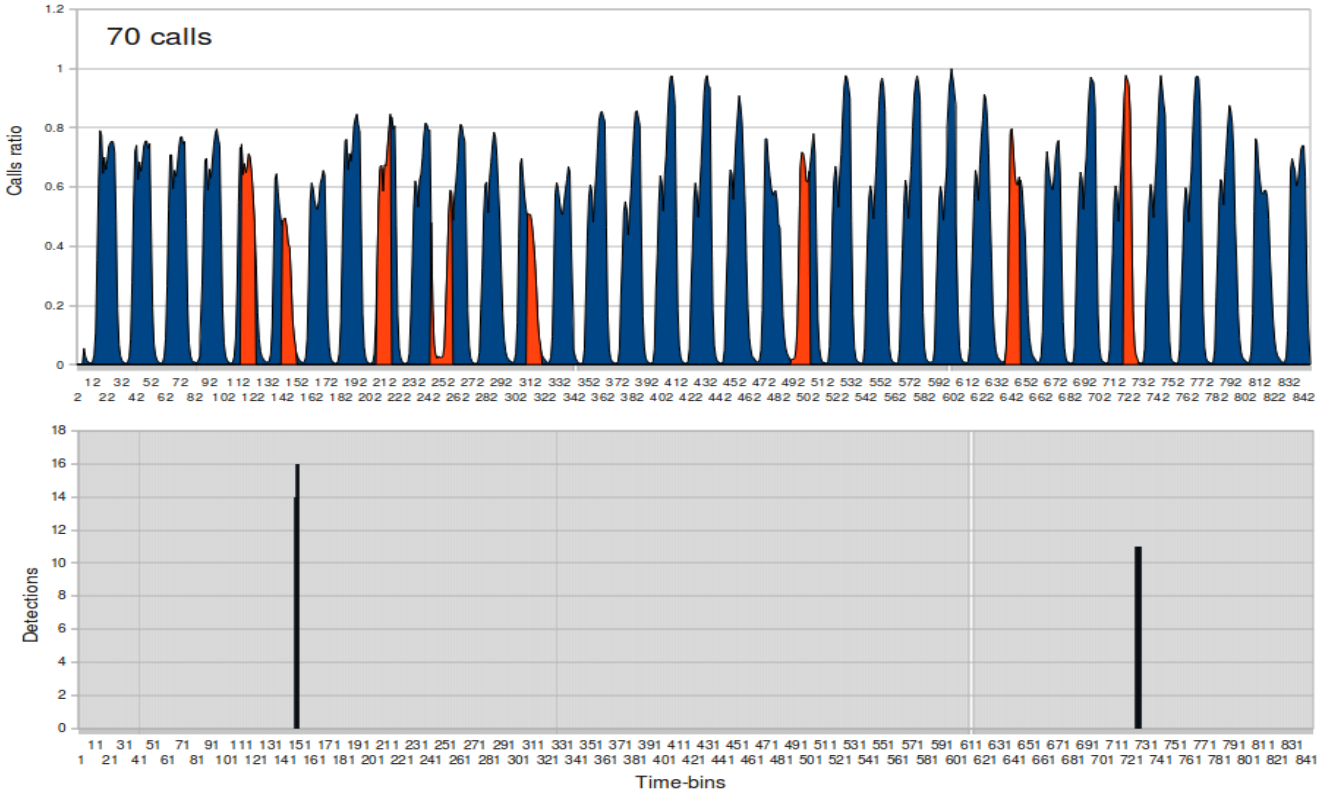


Figure D.3

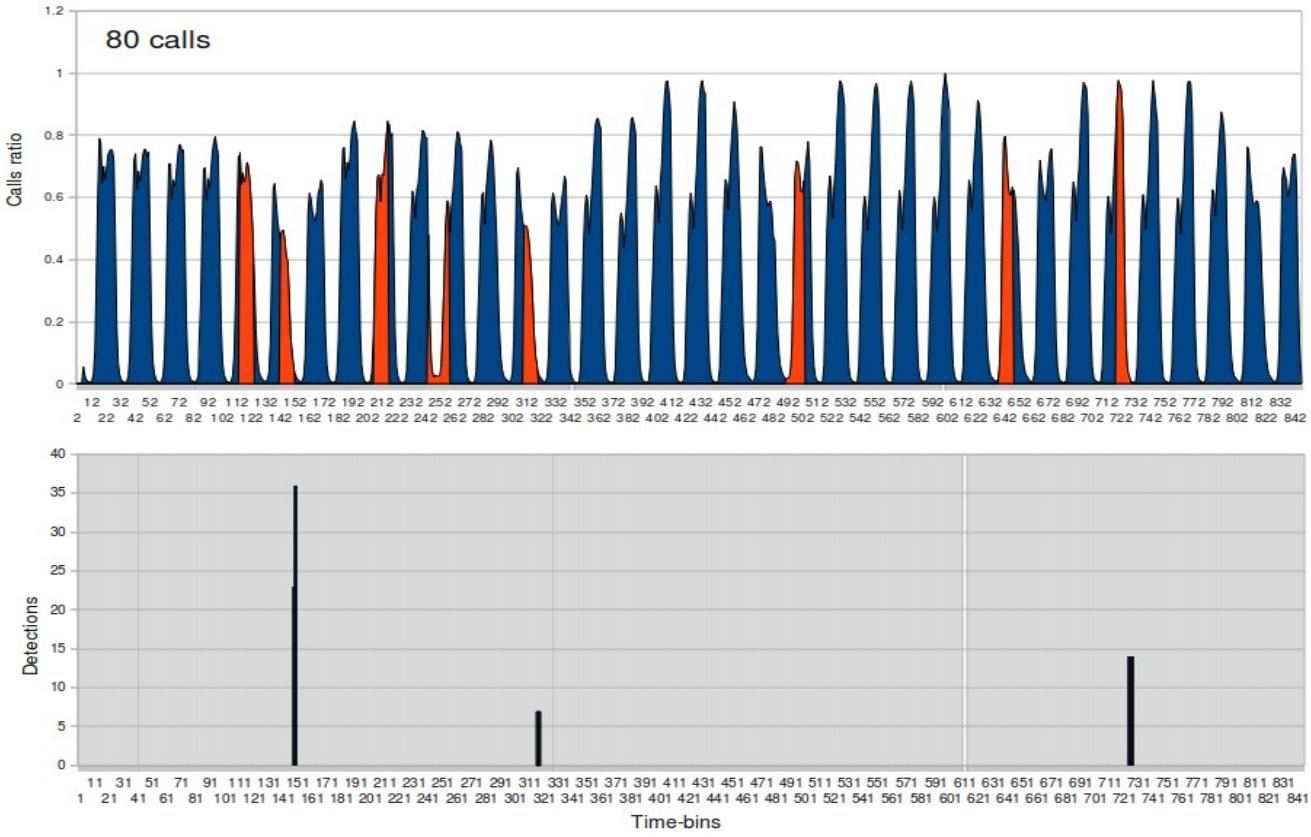


Figure D.4

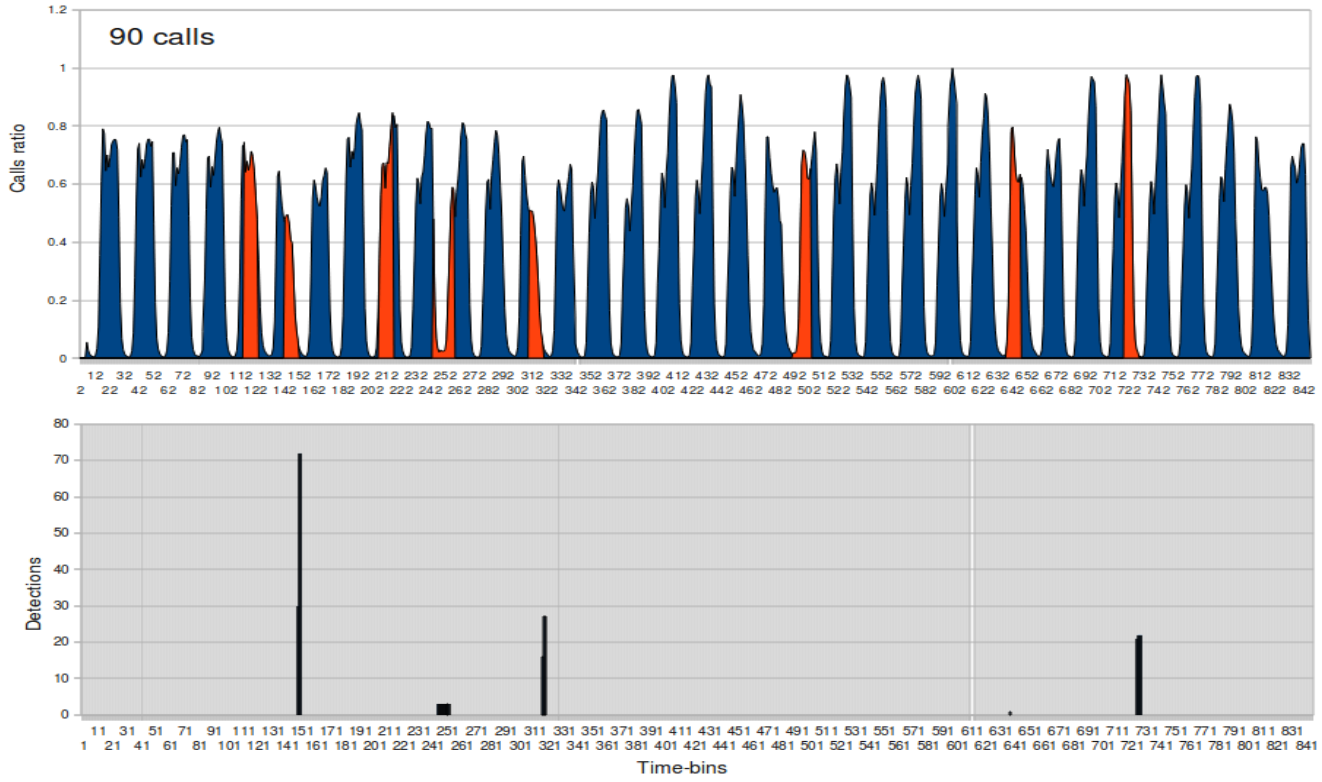


Figure D.5

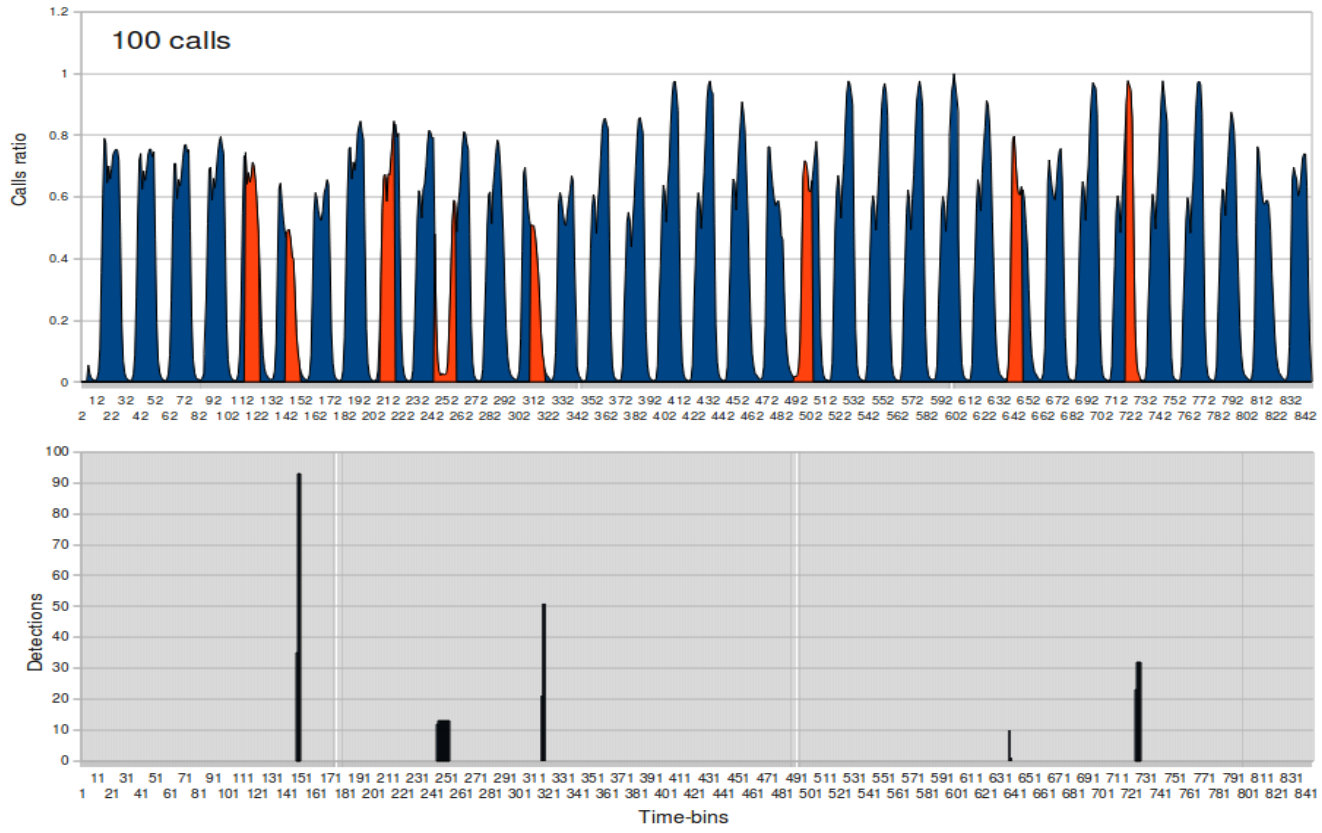


Figure D.6

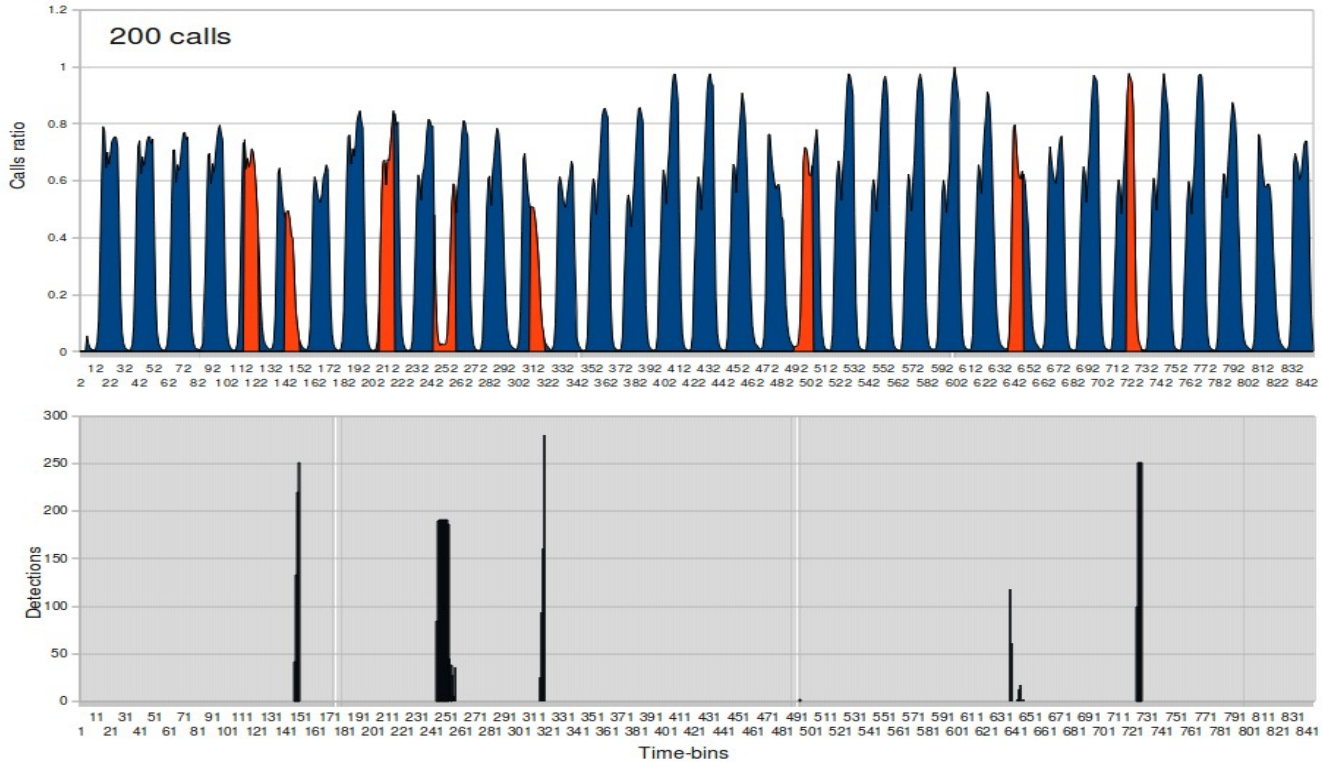


Figure D.7

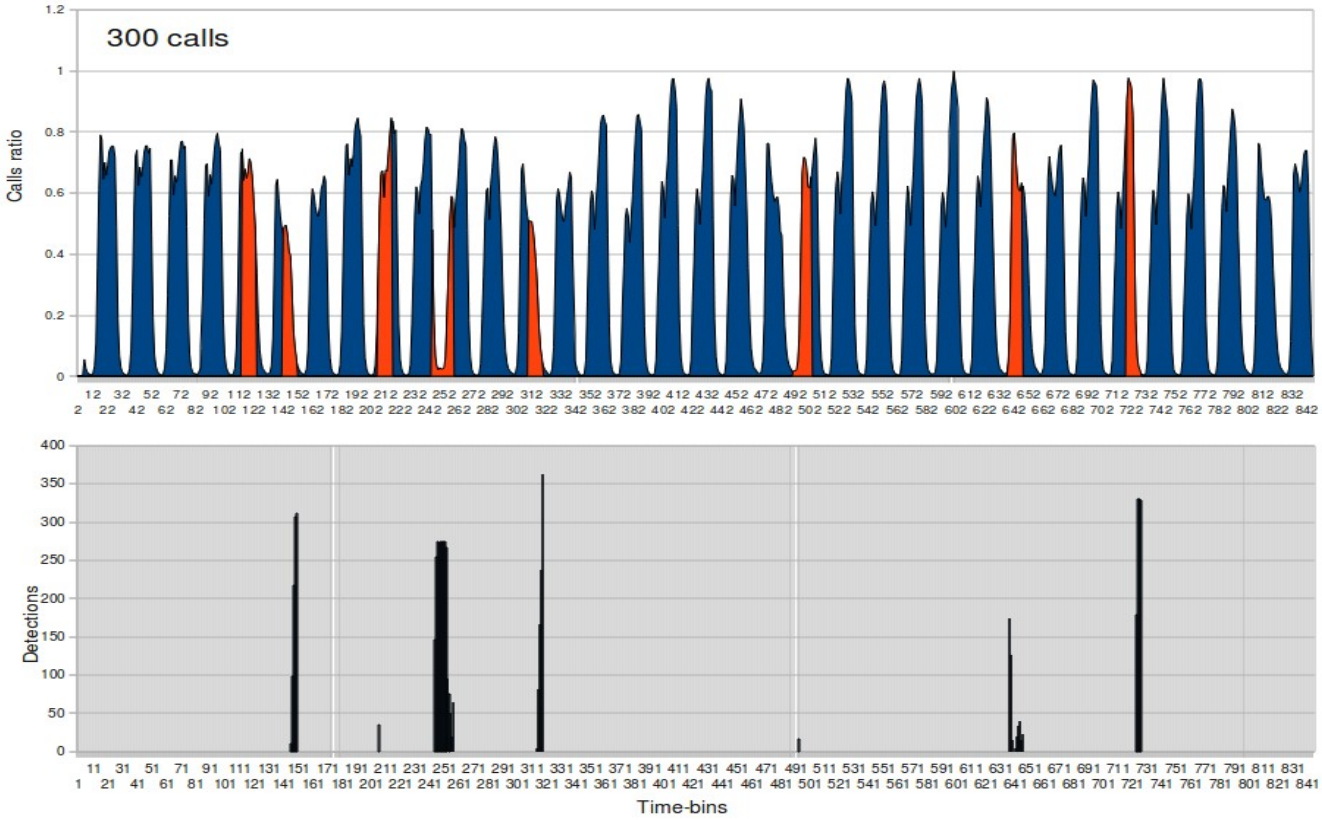


Figure D.8

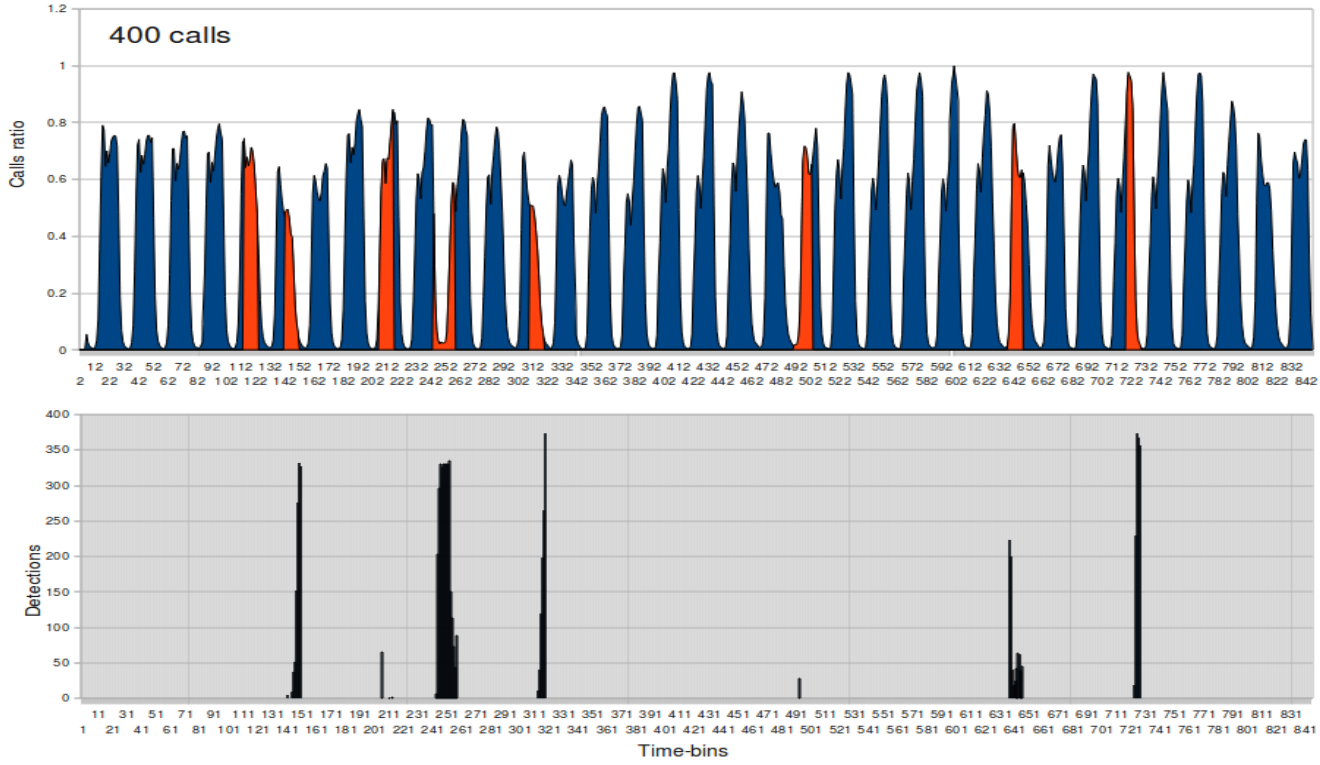


Figure D.9

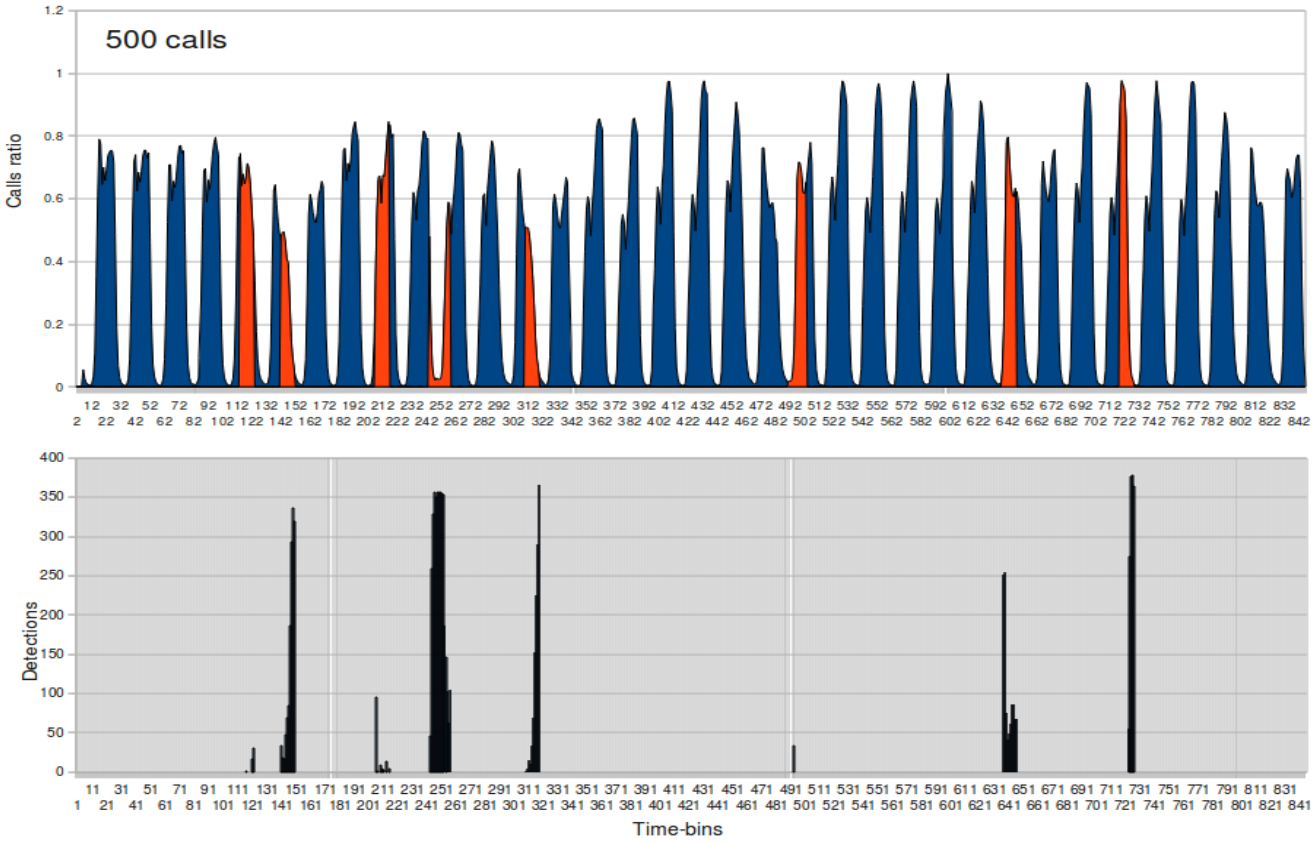


Figure D.10

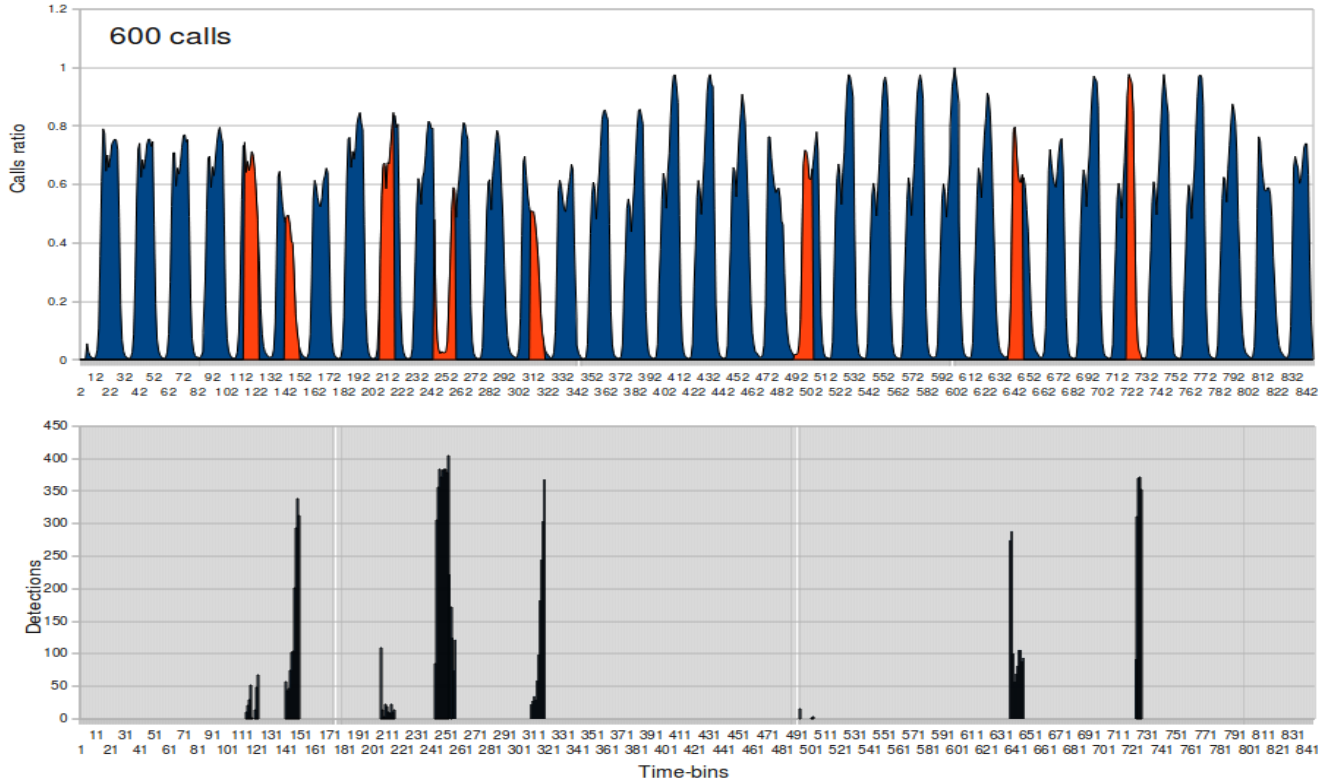


Figure D.11

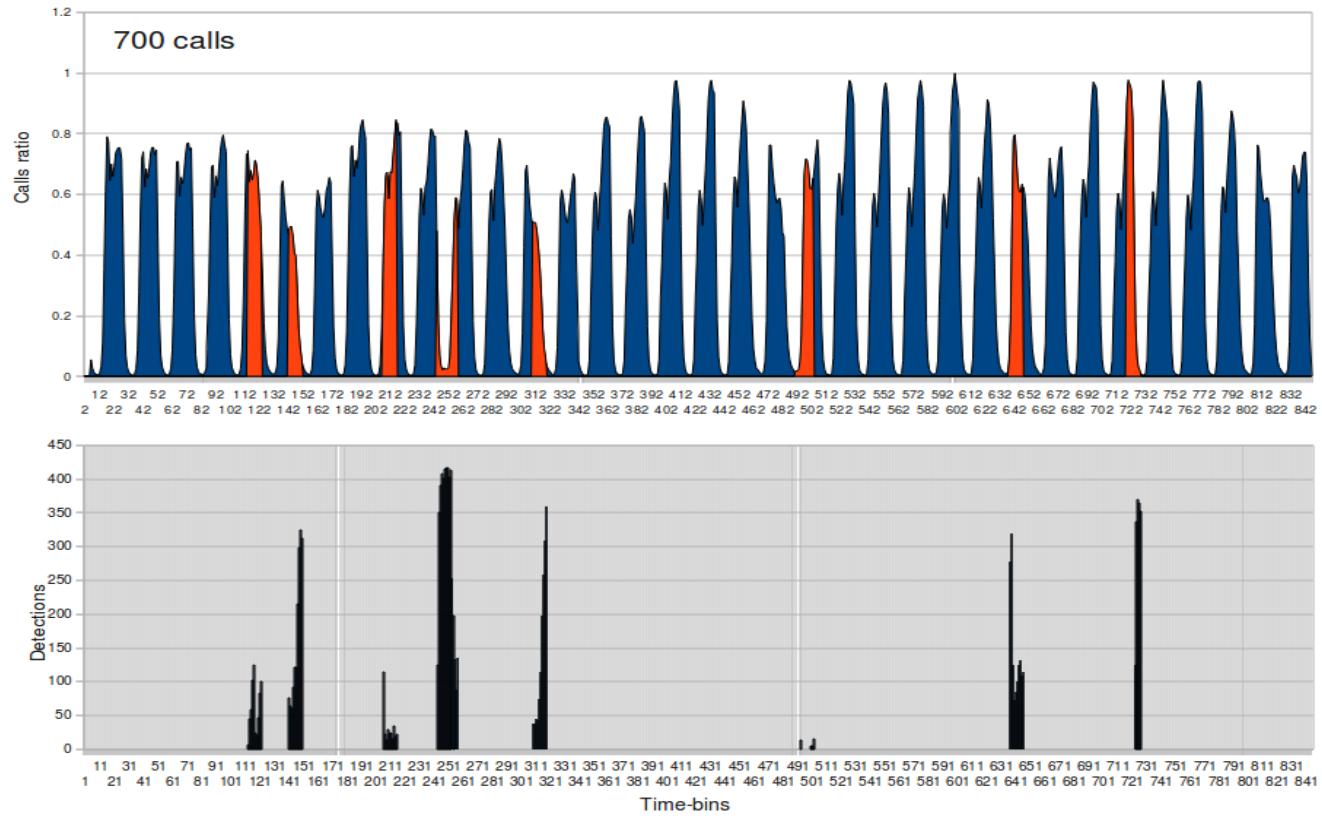


Figure D.12

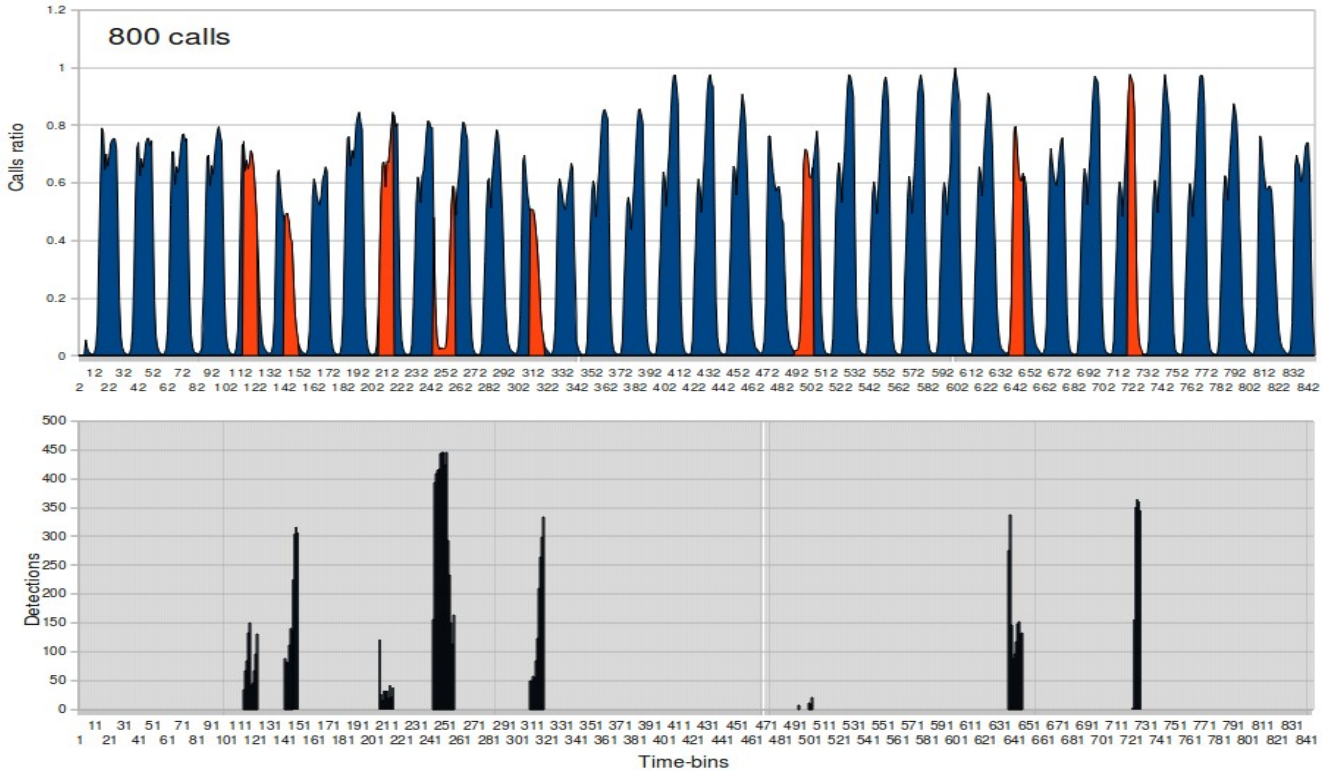


Figure D.13

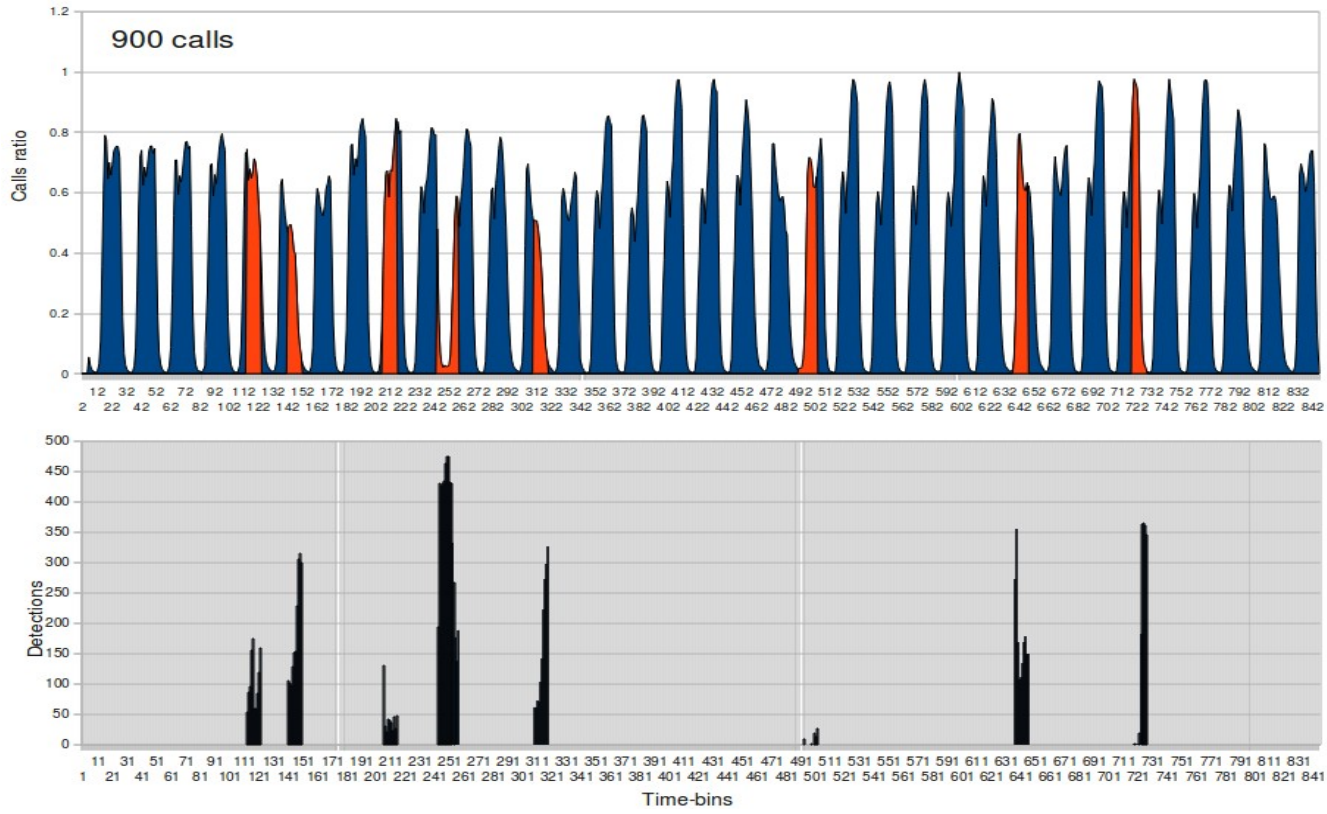
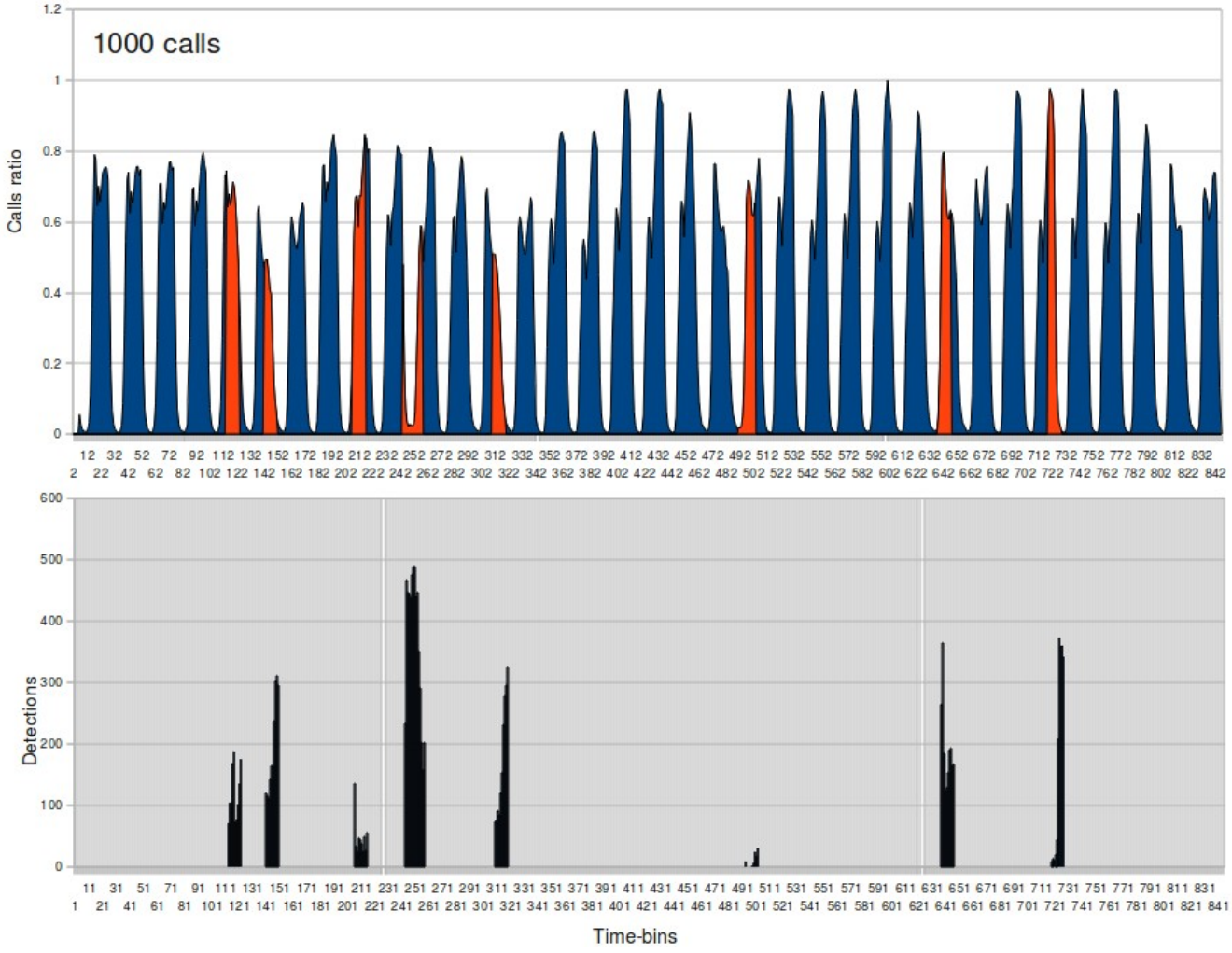


Figure D.14



Ratio of anomalous bins detected and responsible URIs recognized (upon the total of them) with “single threshold-single number of PCs” analysis (anomaly height: 1000 calls)

Figure D.15

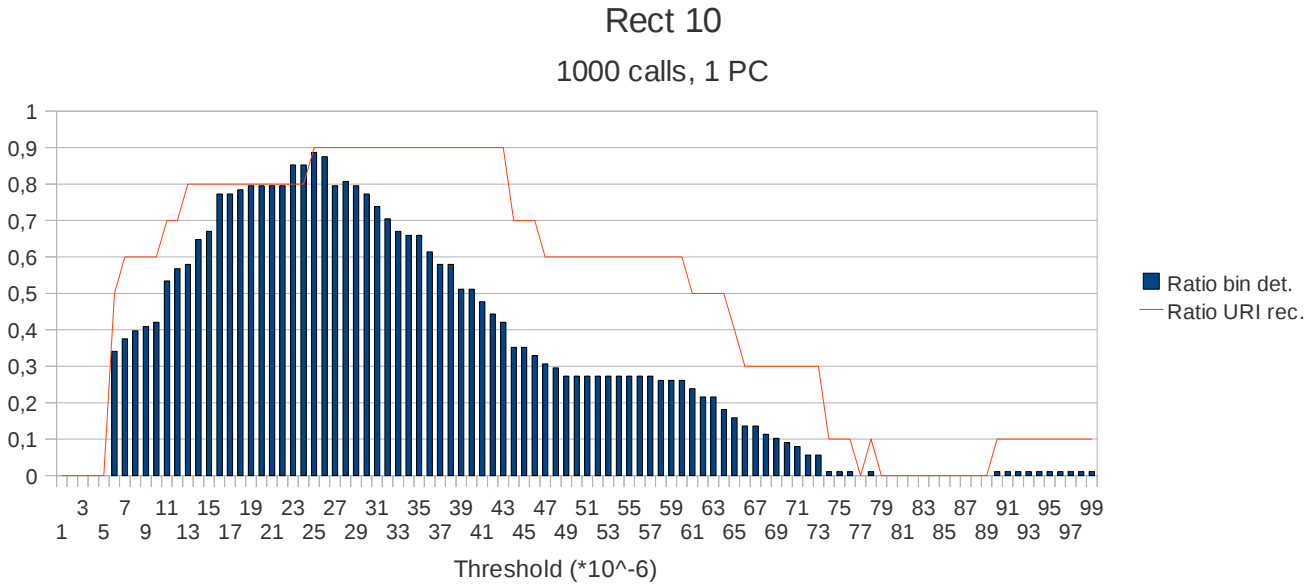


Figure D.16

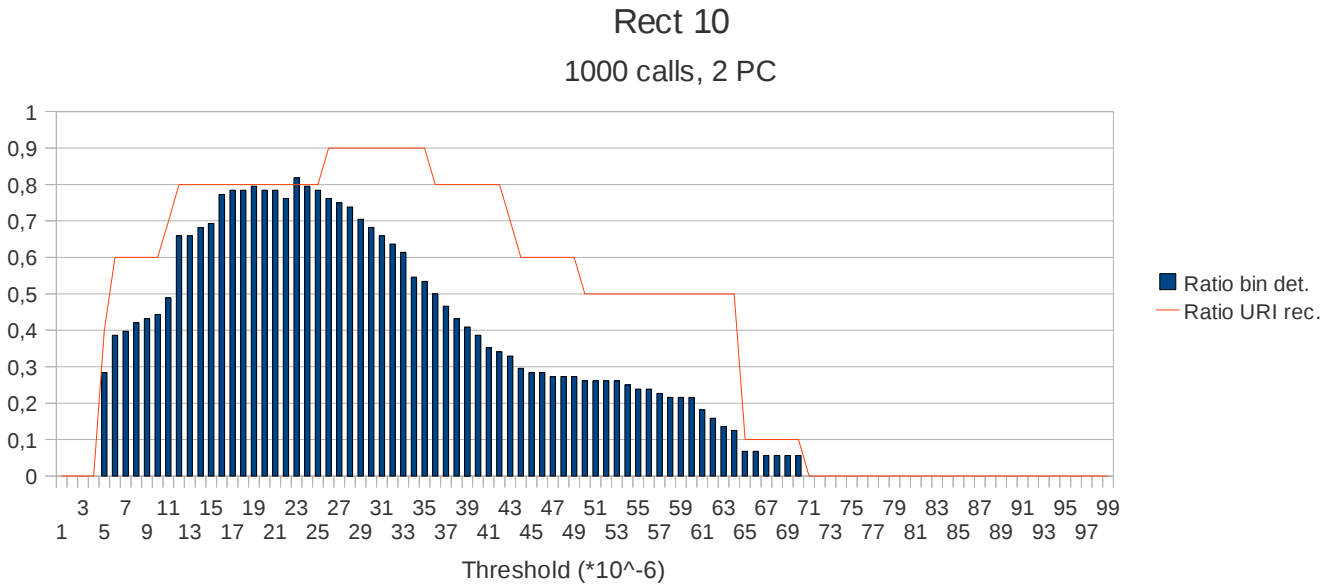


Figure D.17

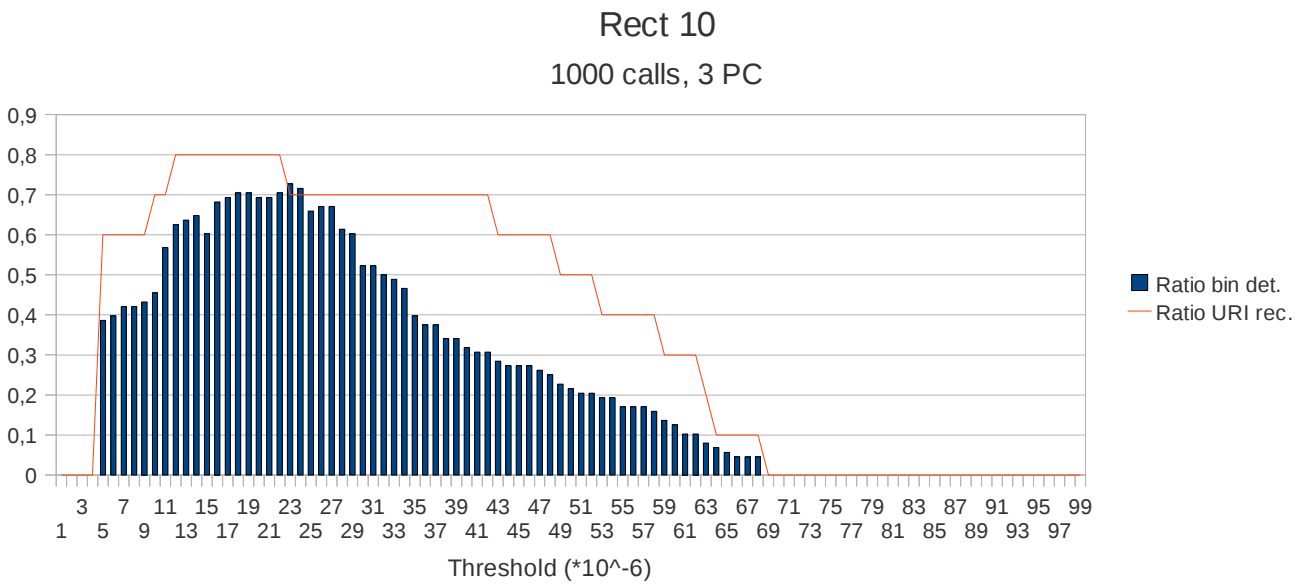


Figure D.18

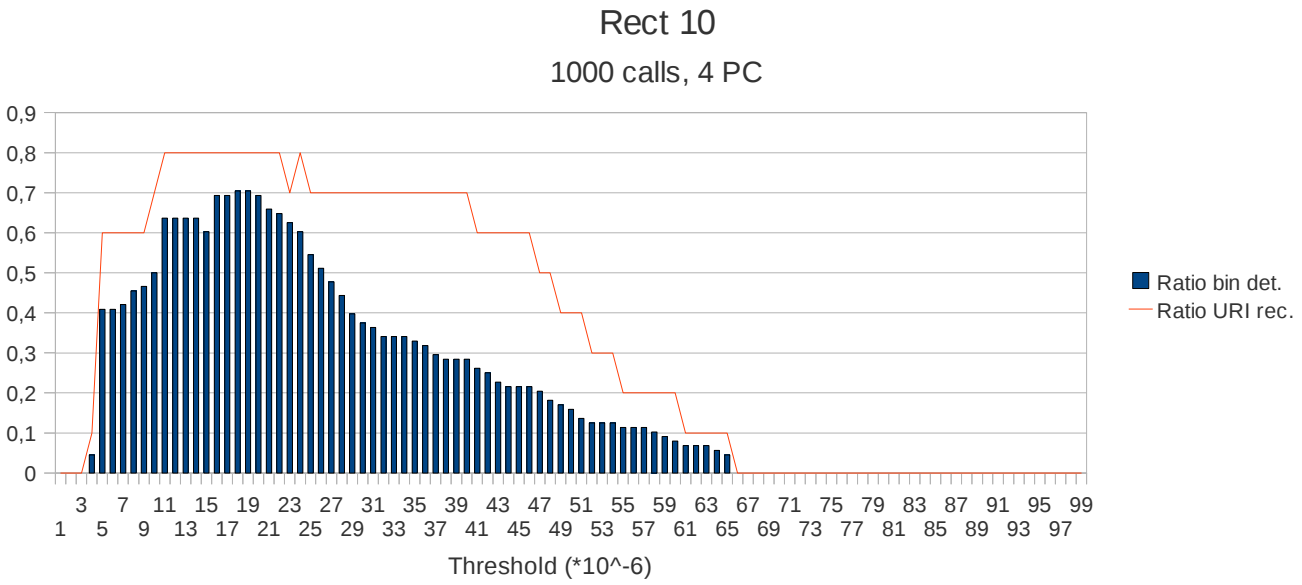


Figure D.19

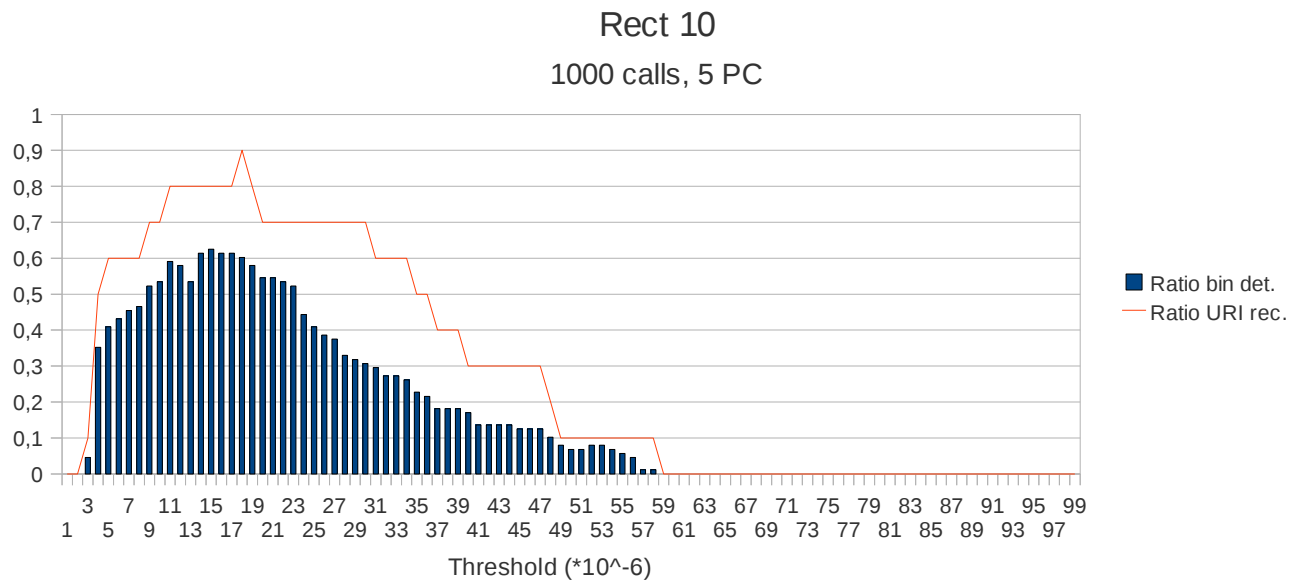


Figure D.20

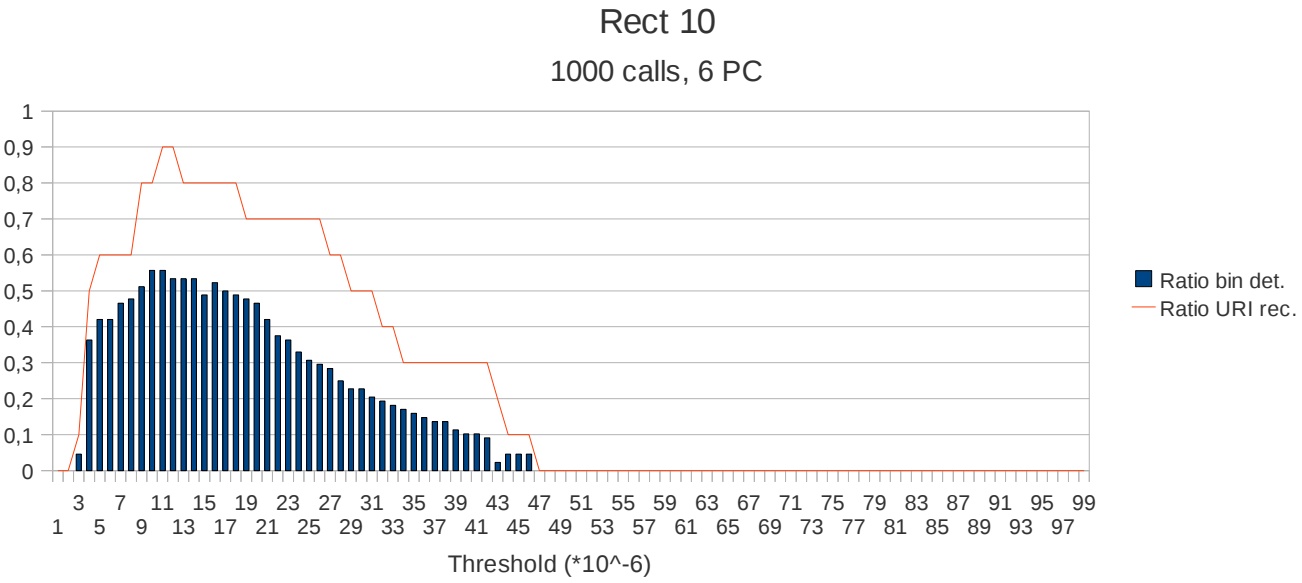


Figure D.21

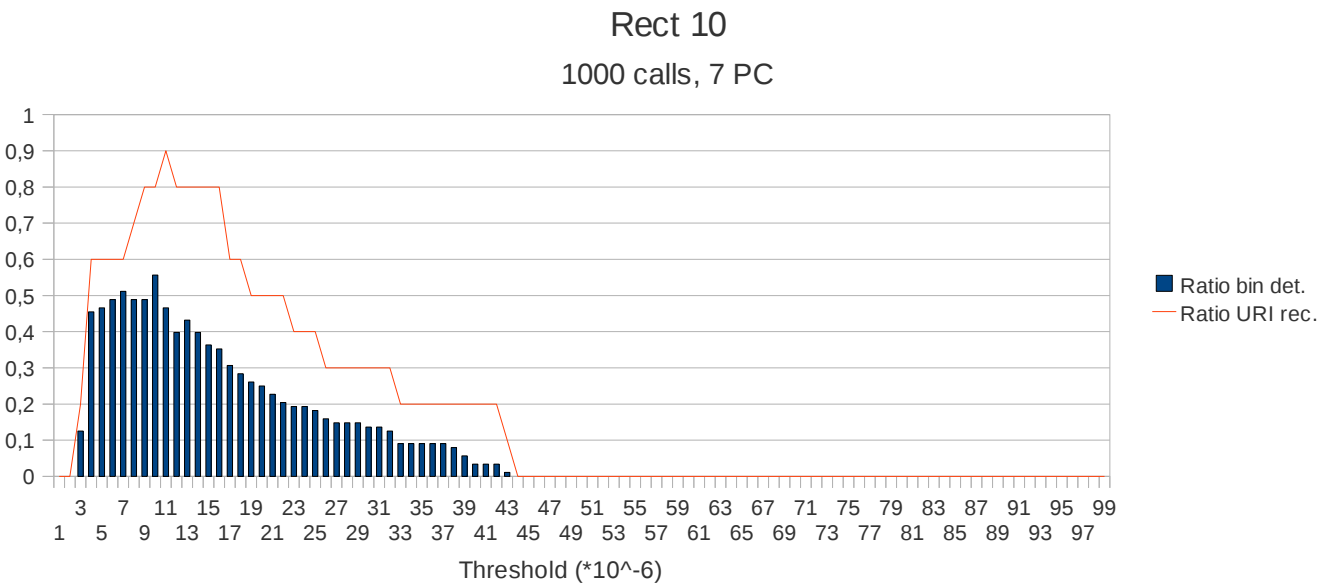


Figure D.22

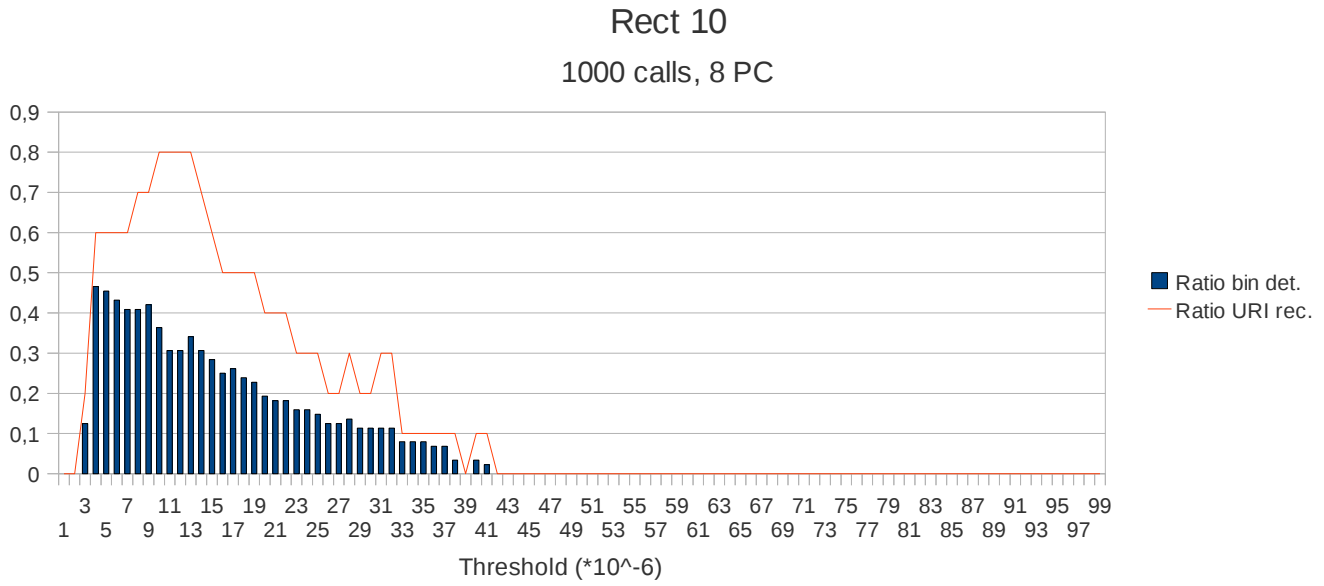


Figure D.23

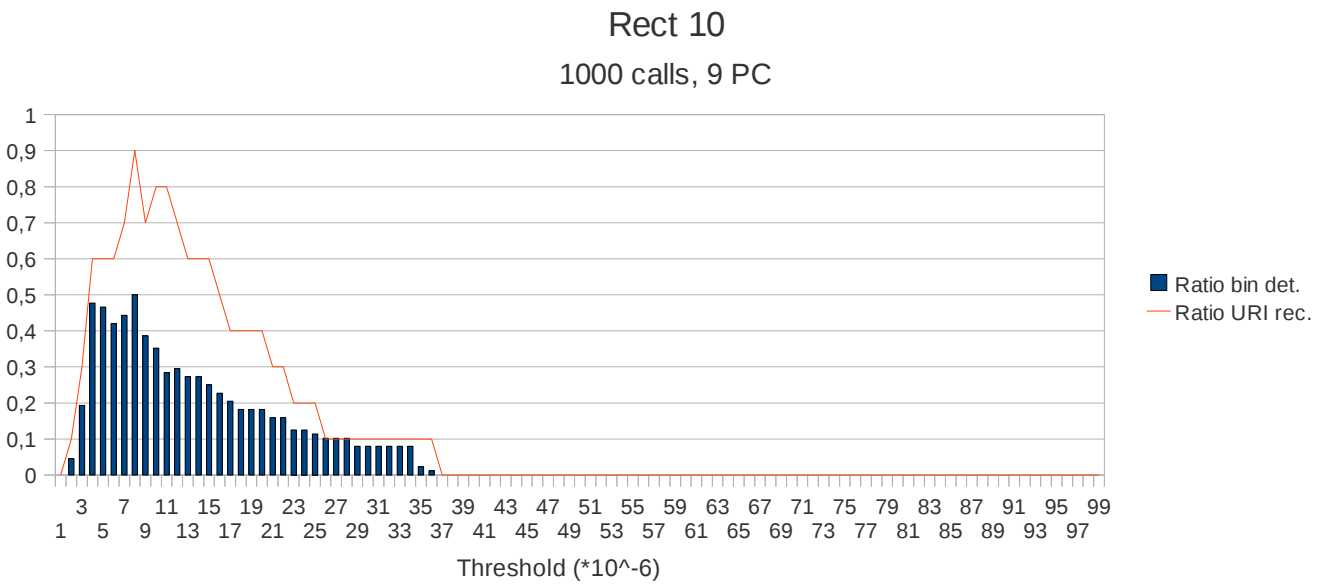
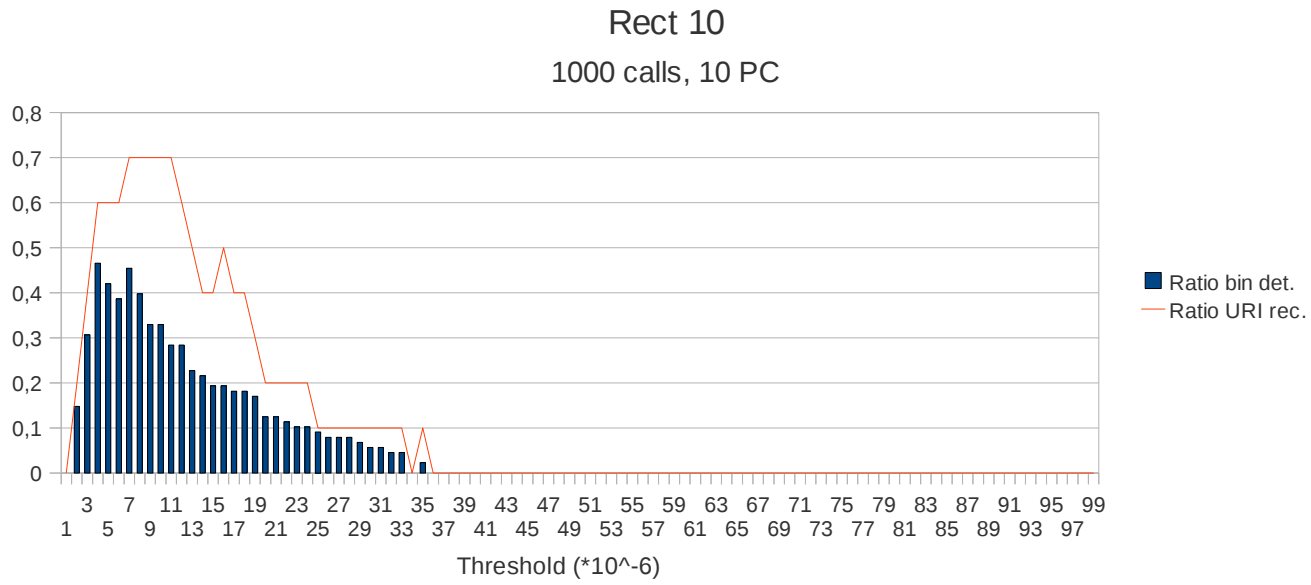


Figure D.24



Appendix E

(Triangular artificial anomalies, width 5)

Number of detections for each time-bin in a "100 thresholds-10 numbers of PCs" analysis VS ratio of global calls per time-bin on their maximum value

Figure E.1

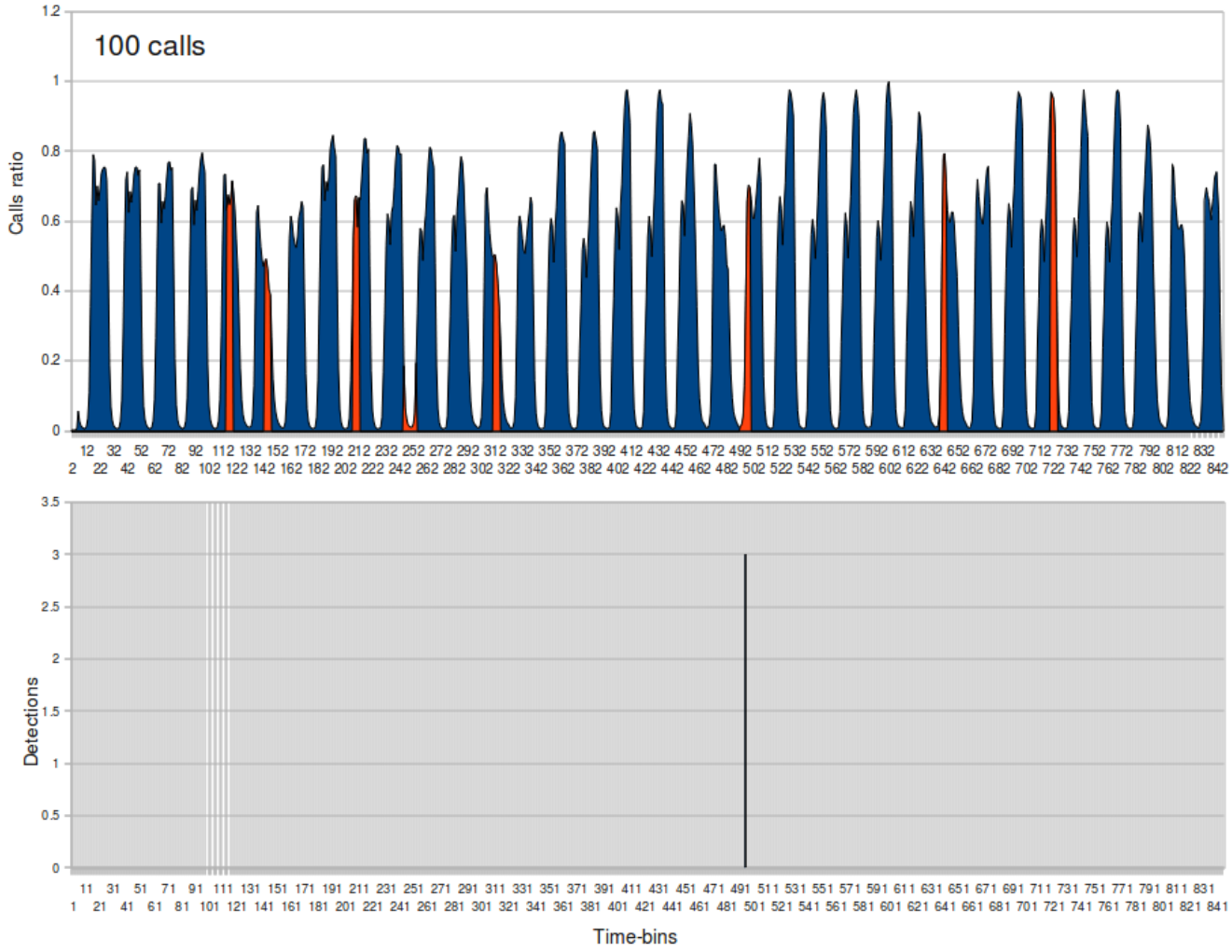


Figure E.2

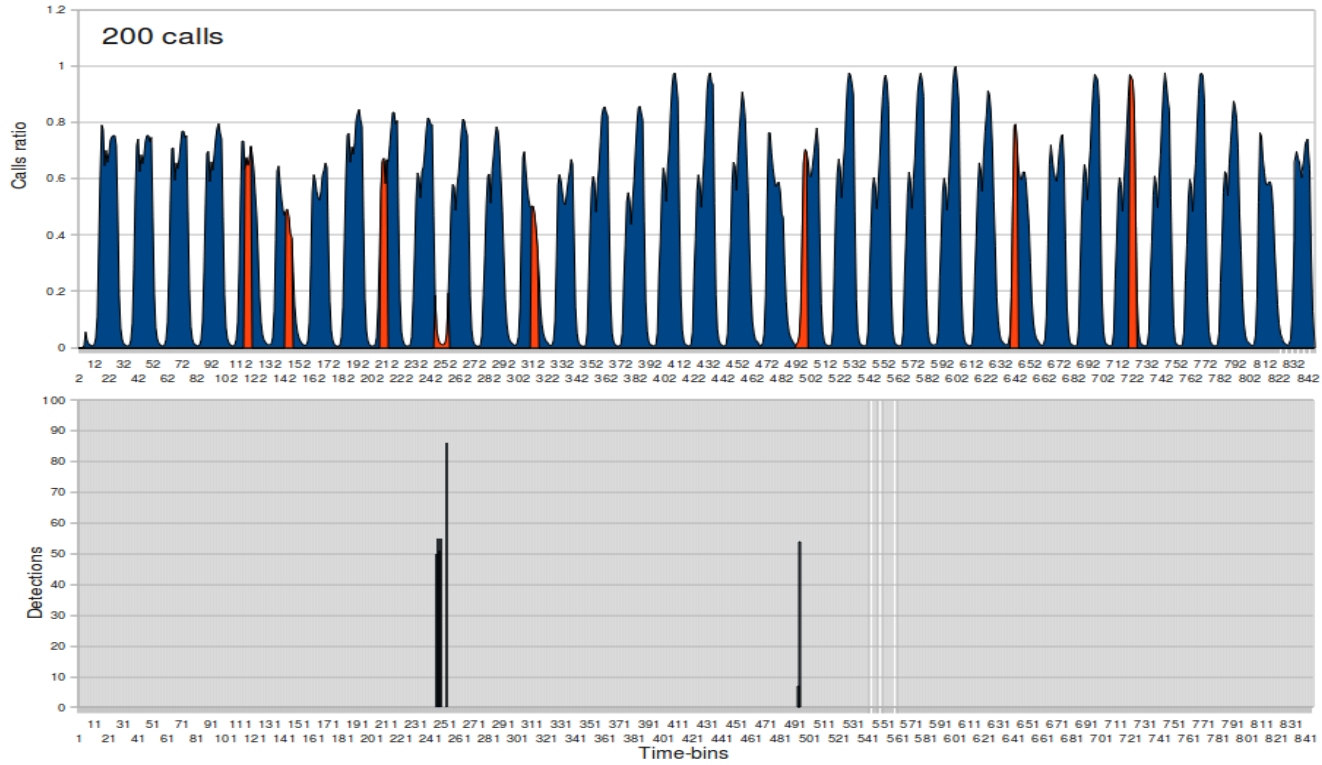


Figure E.3

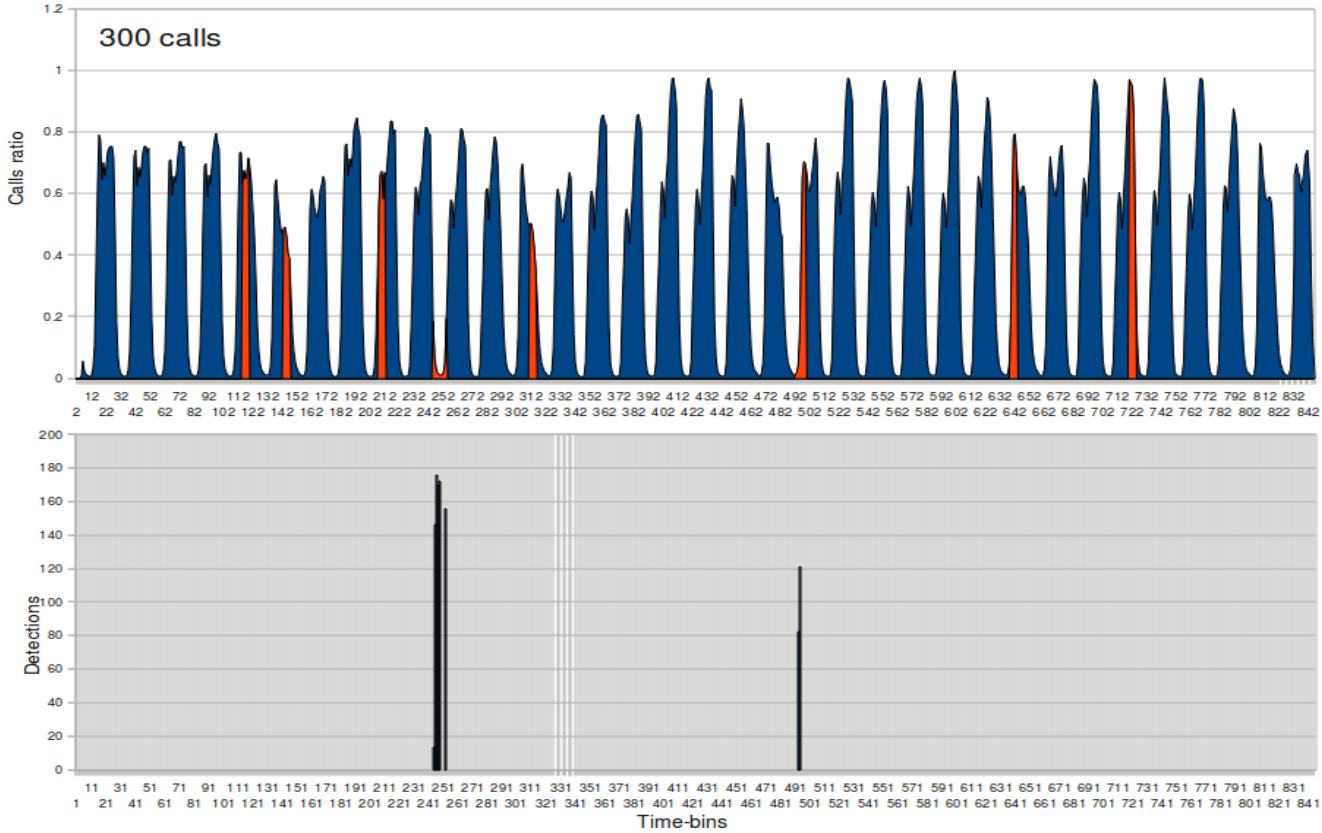


Figure E.4

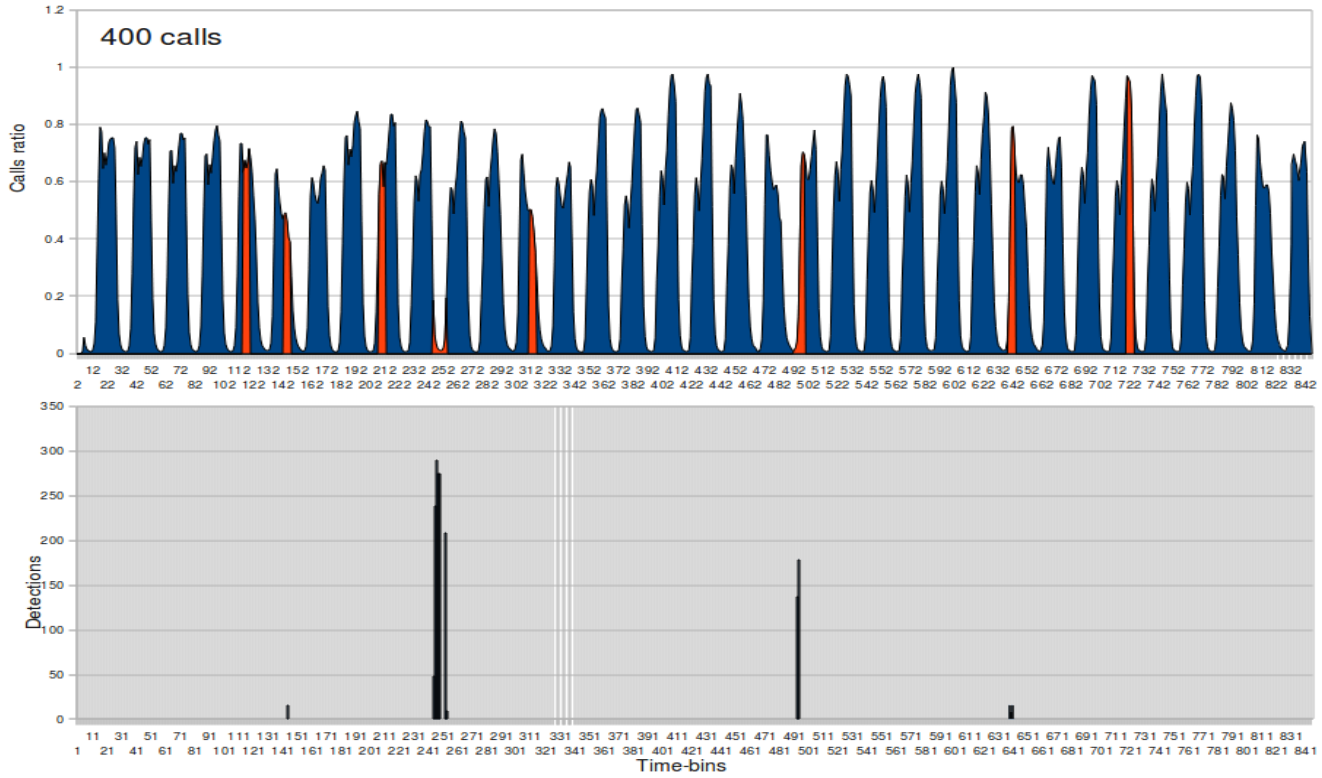


Figure E.5

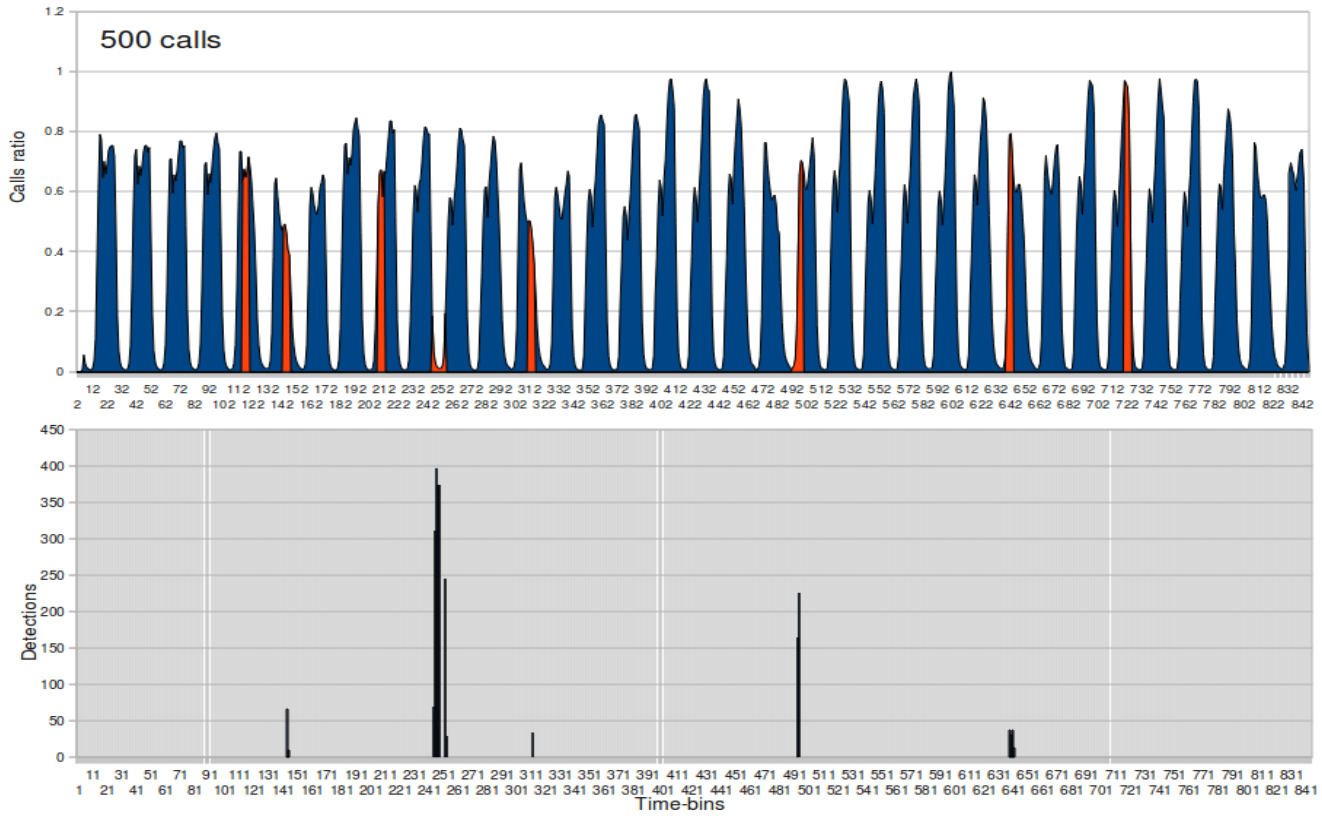


Figure E.6

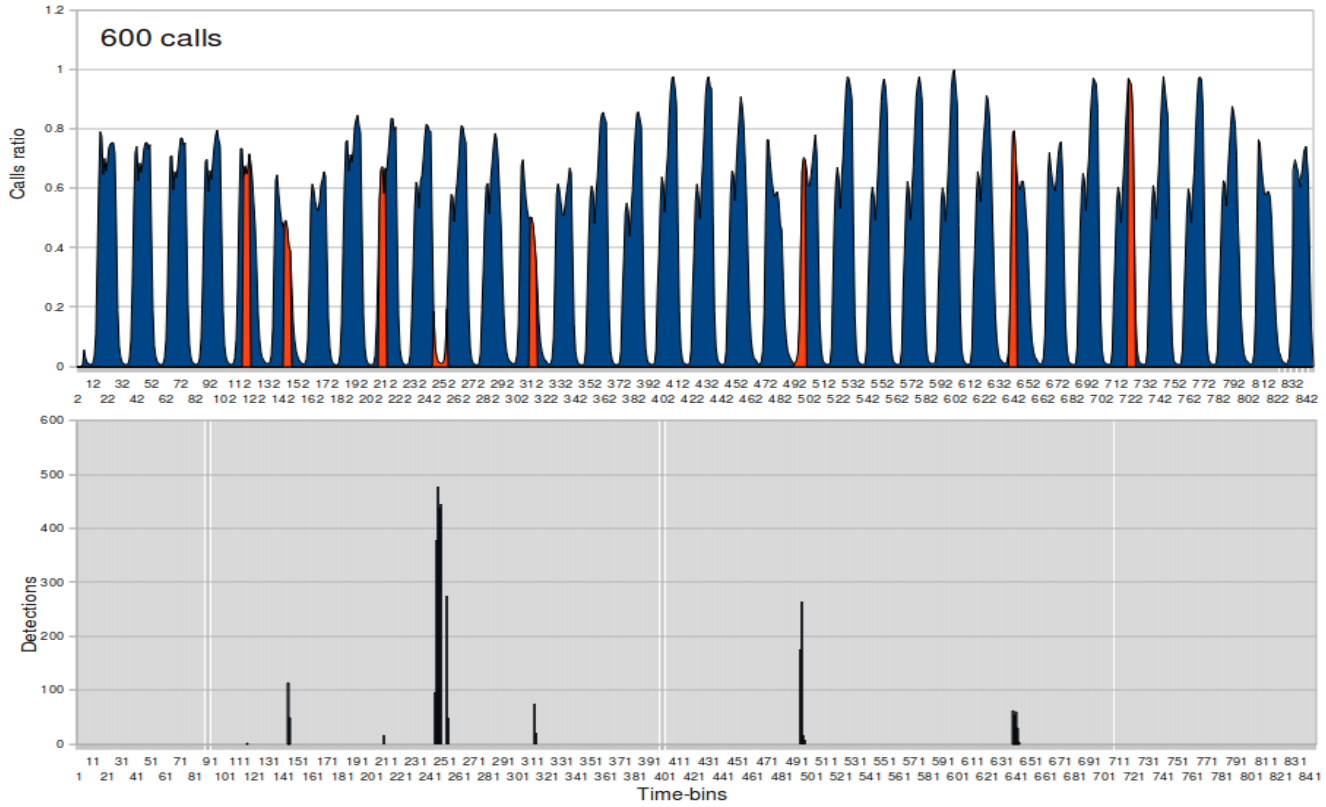


Figure E.7

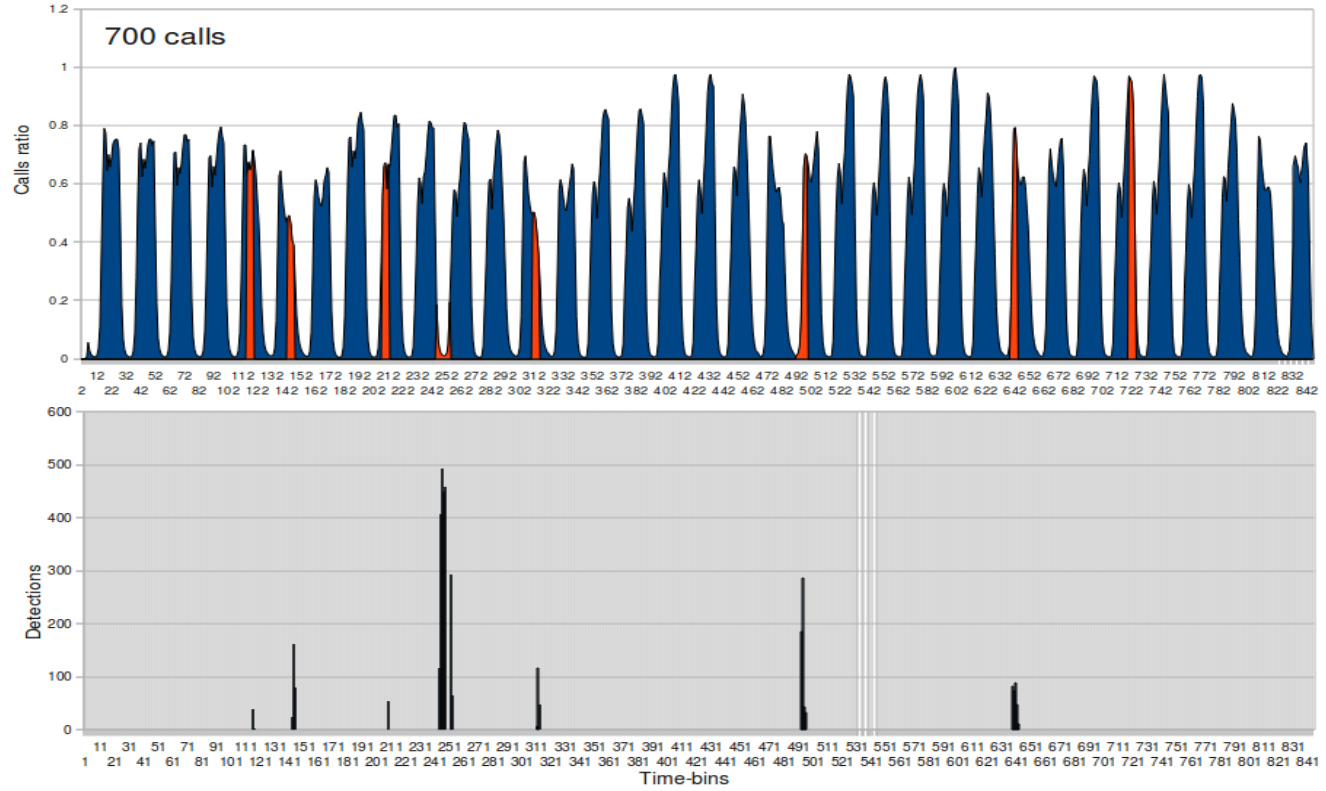


Figure E.8

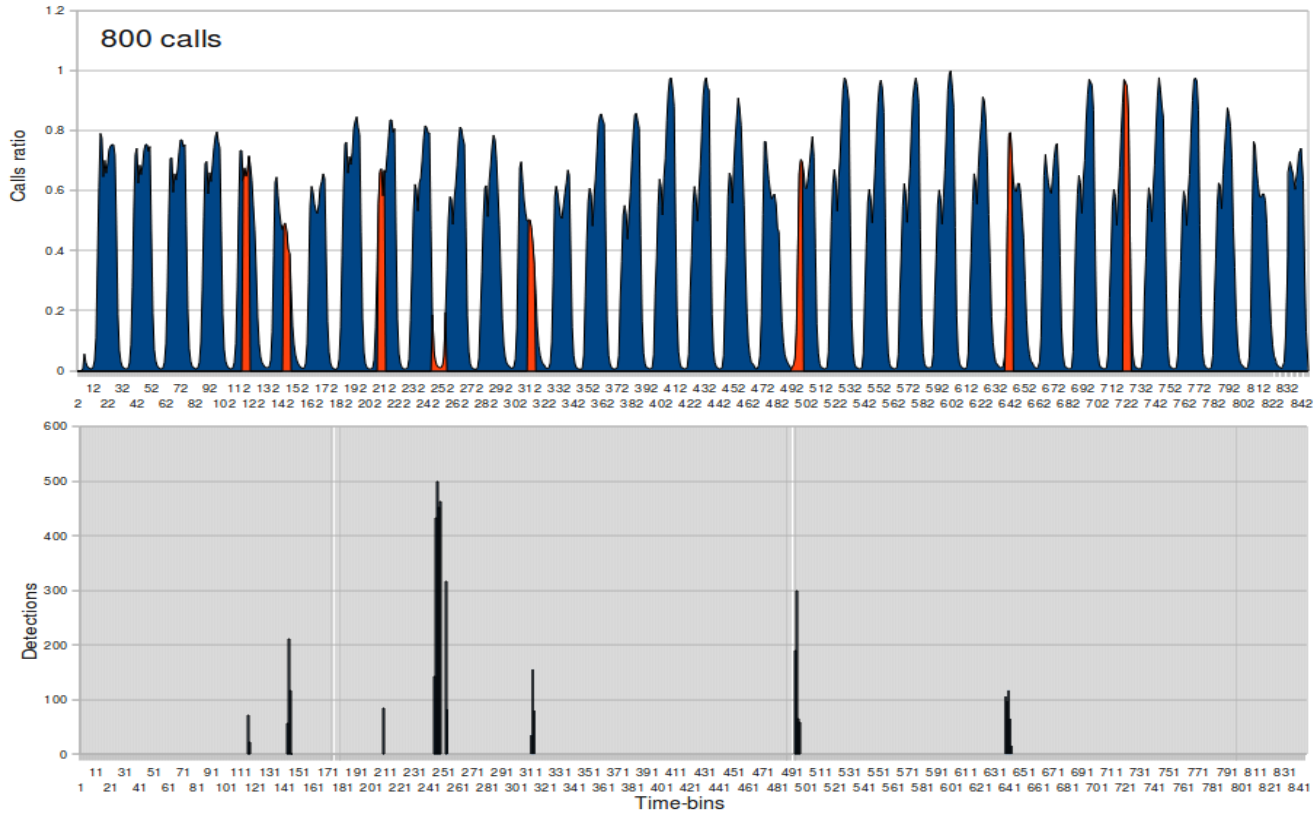


Figure E.9

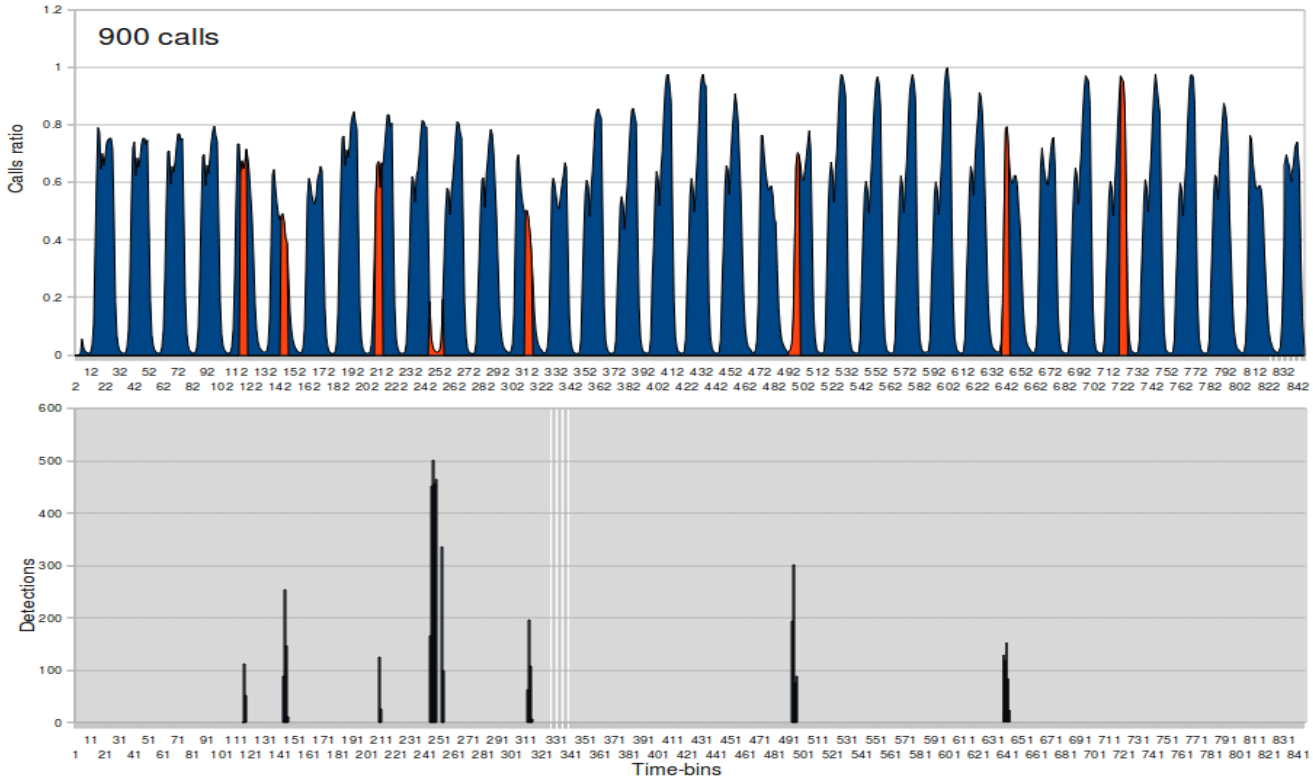
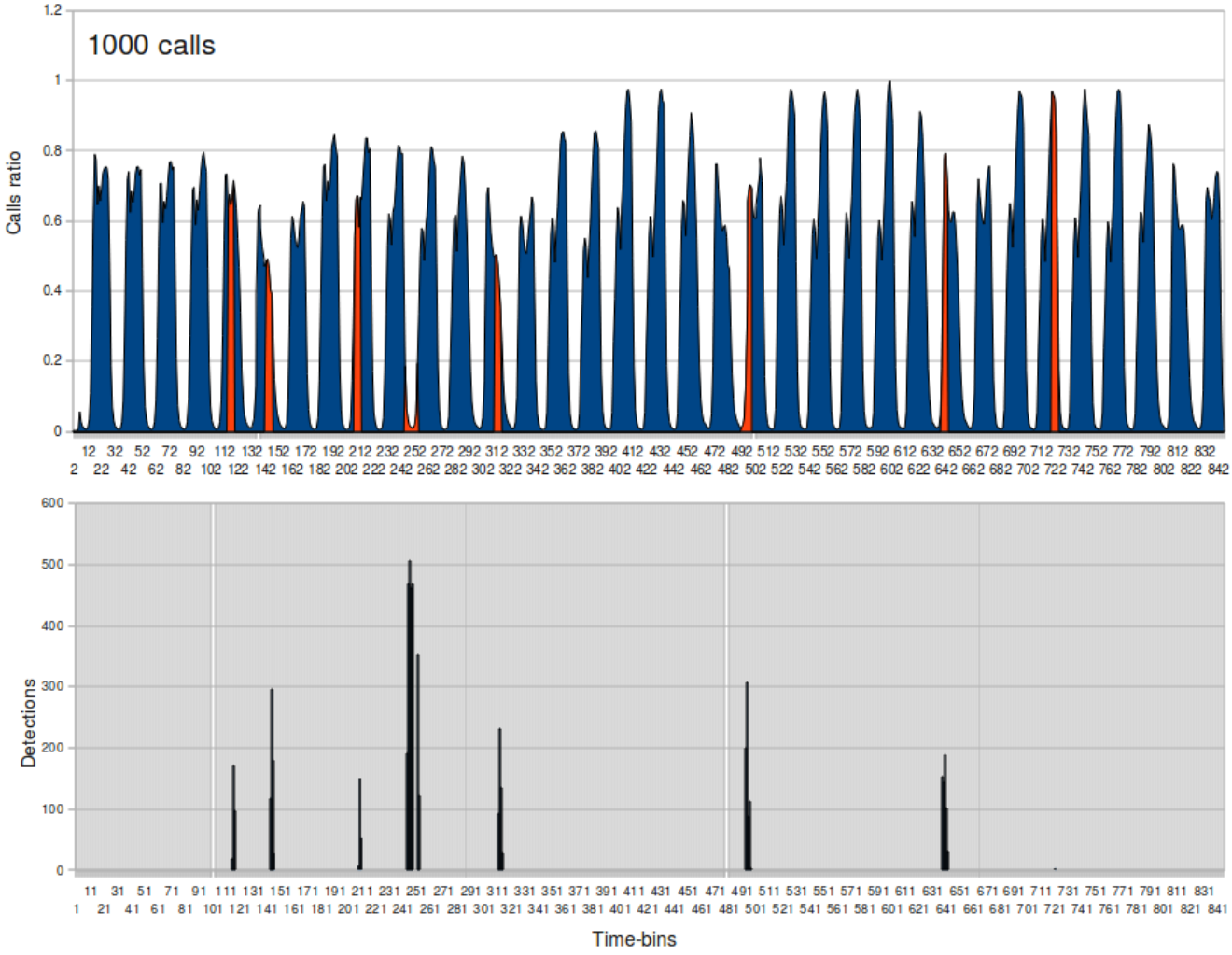


Figure E.10


1000 calls
Calls ratio
1.2
1
0.8
0.6
0.4
0.2
0
12 32 52 72 92 112 132 152 172 192 212 232 252 272 292 312 332 352 372 392 412 432 452 472 492 512 532 552 572 592 612 632 652 672 692 712 732 752 772 792 812 832
2 22 42 62 82 102 122 142 162 182 202 222 242 262 282 302 322 342 362 382 402 422 442 462 482 502 522 542 562 582 602 622 642 662 682 702 722 742 762 782 802 822 842
Detections
600
500
400
300
200
100
0
11 31 51 71 91 111 131 151 171 191 211 231 251 271 291 311 331 351 371 391 411 431 451 471 491 511 531 551 571 591 611 631 651 671 691 711 731 751 771 791 811 831
1 21 41 61 81 101 121 141 161 181 201 221 241 261 281 301 321 341 361 381 401 421 441 461 481 501 521 541 561 581 601 621 641 661 681 701 721 741 761 781 801 821 841
Time-bins

Ratio of anomalous bins detected and responsible URIs recognized (upon the total of them) with “single threshold-single number of PCs” analysis (anomaly height: 1000 calls)

Figure E.11

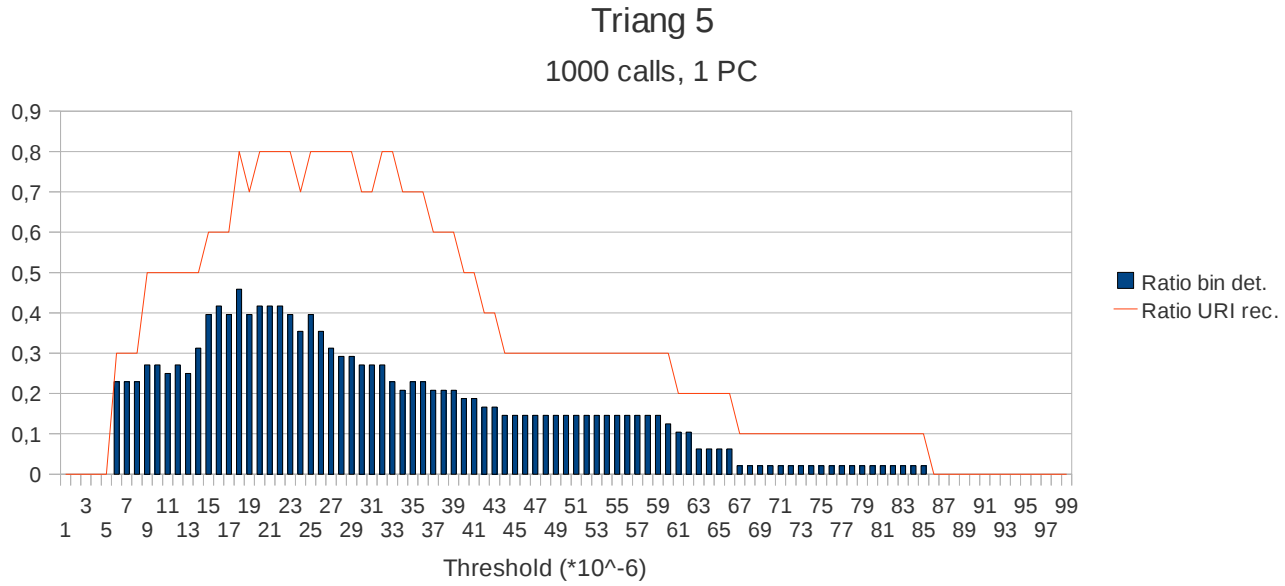


Figure E.12

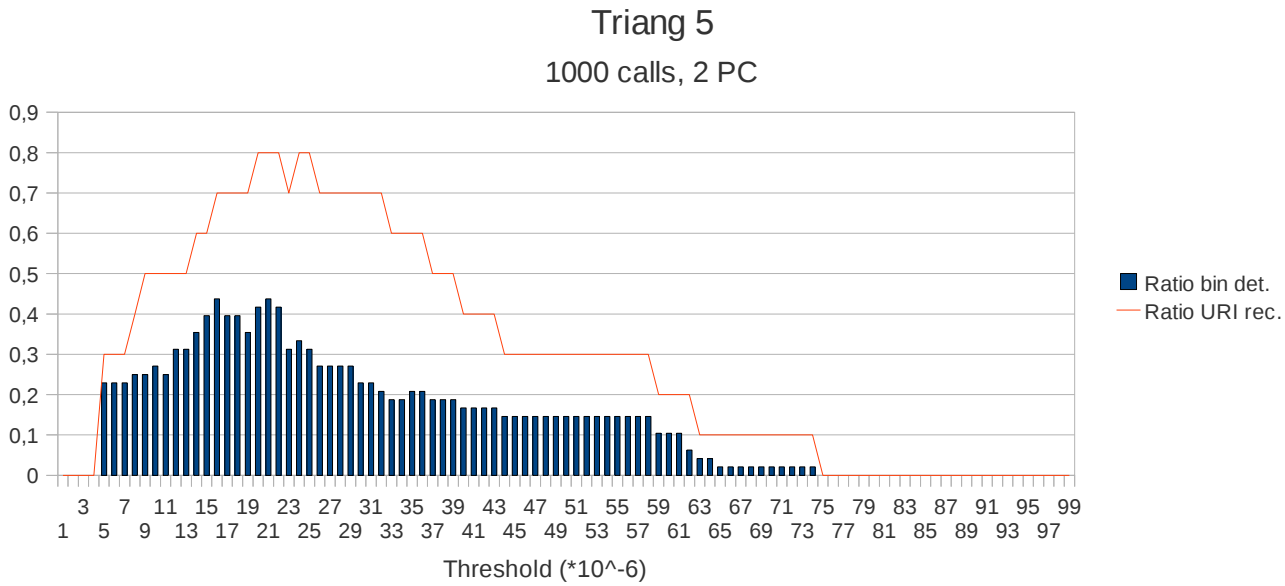


Figure E.13

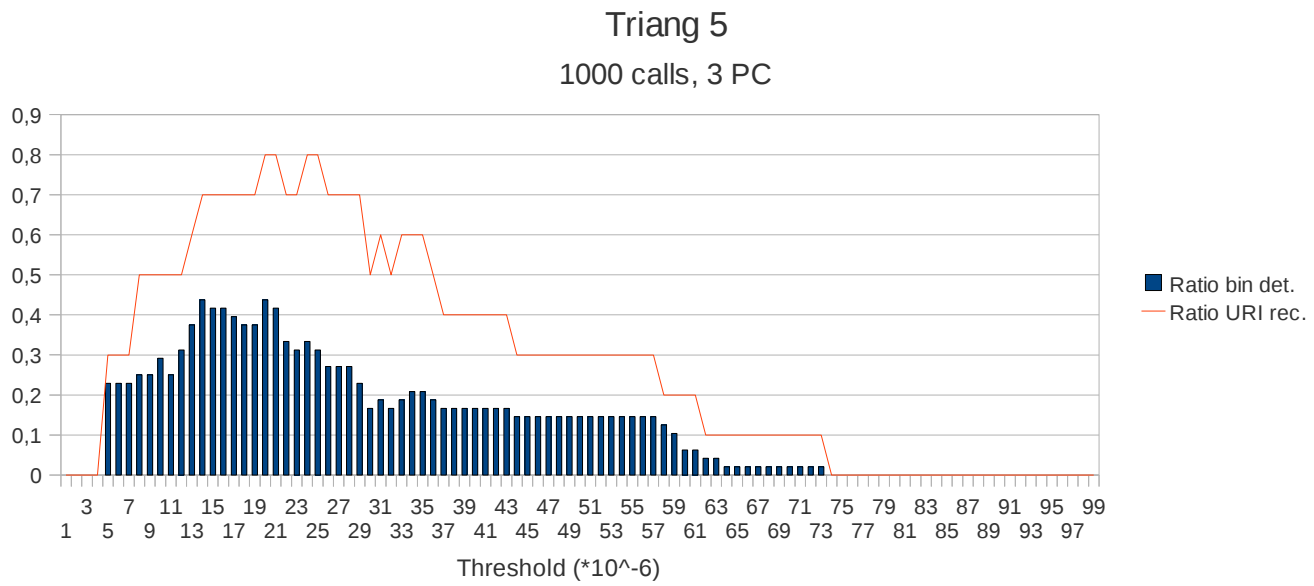


Figure E.14

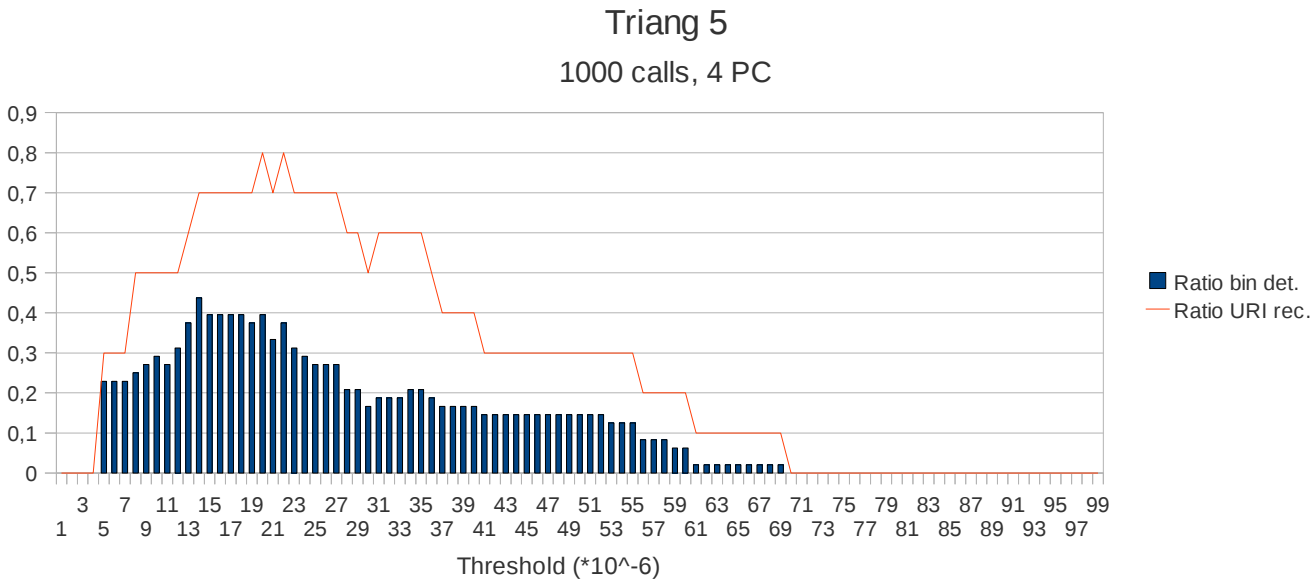


Figure E.15

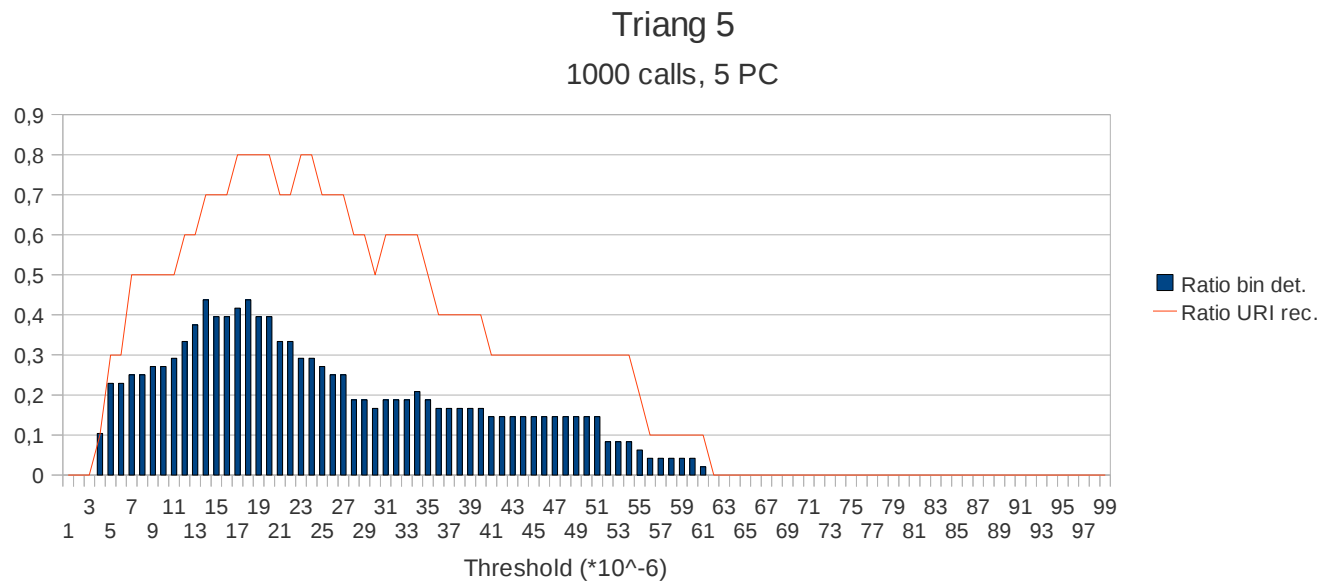


Figure E.16

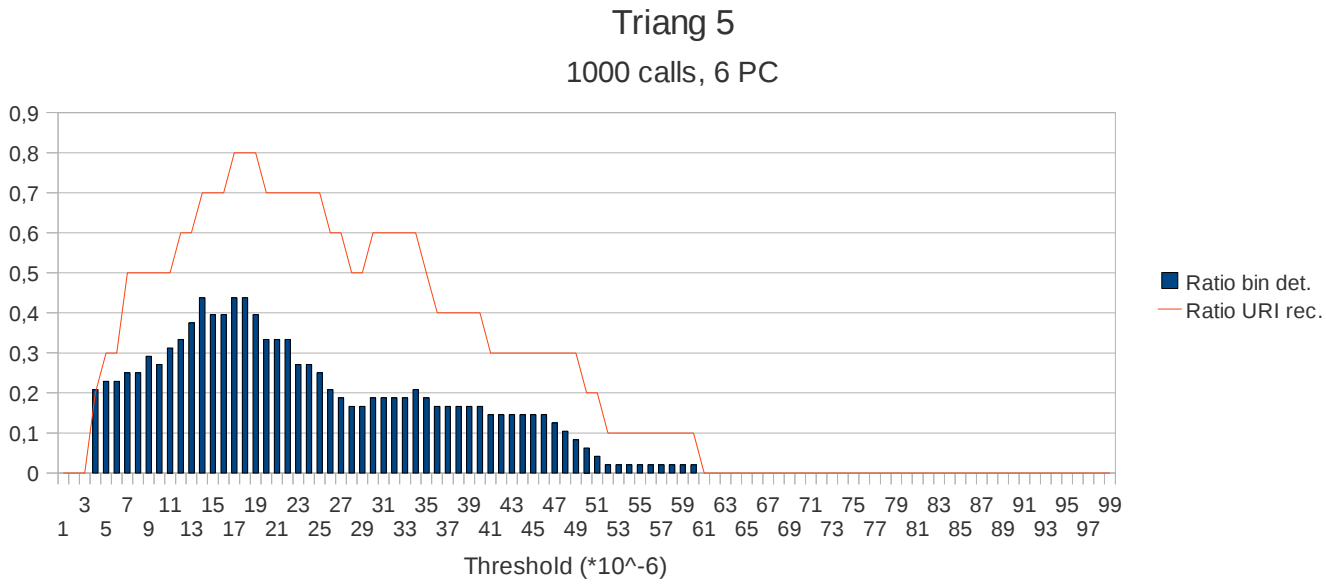


Figure E.17

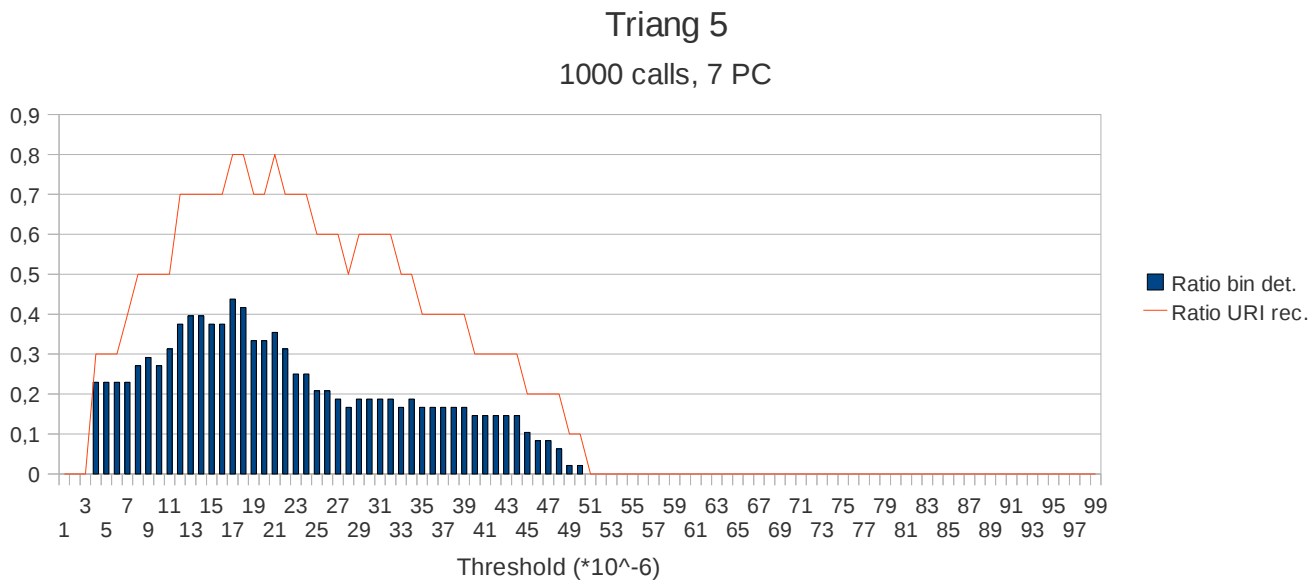


Figure E.18

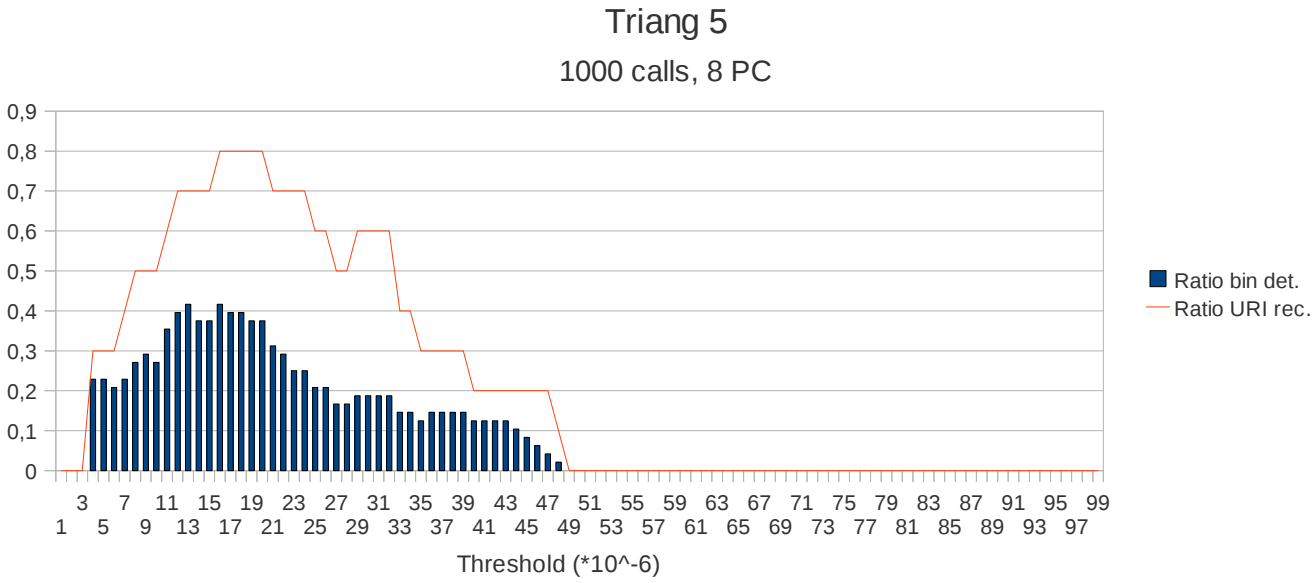


Figure E.19

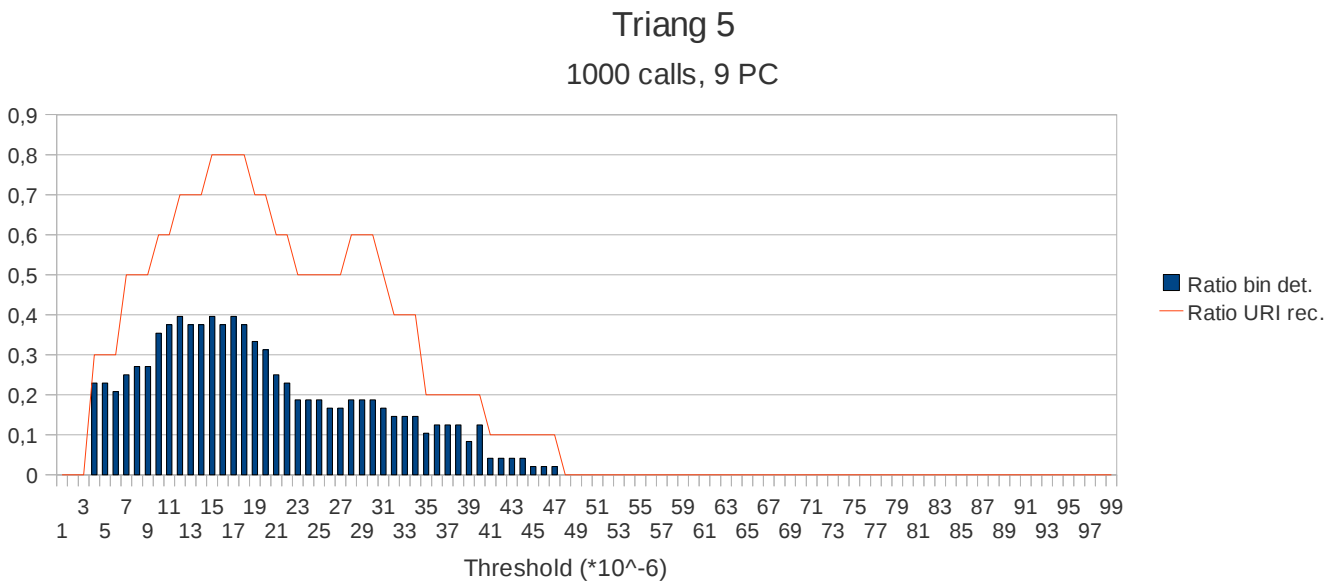
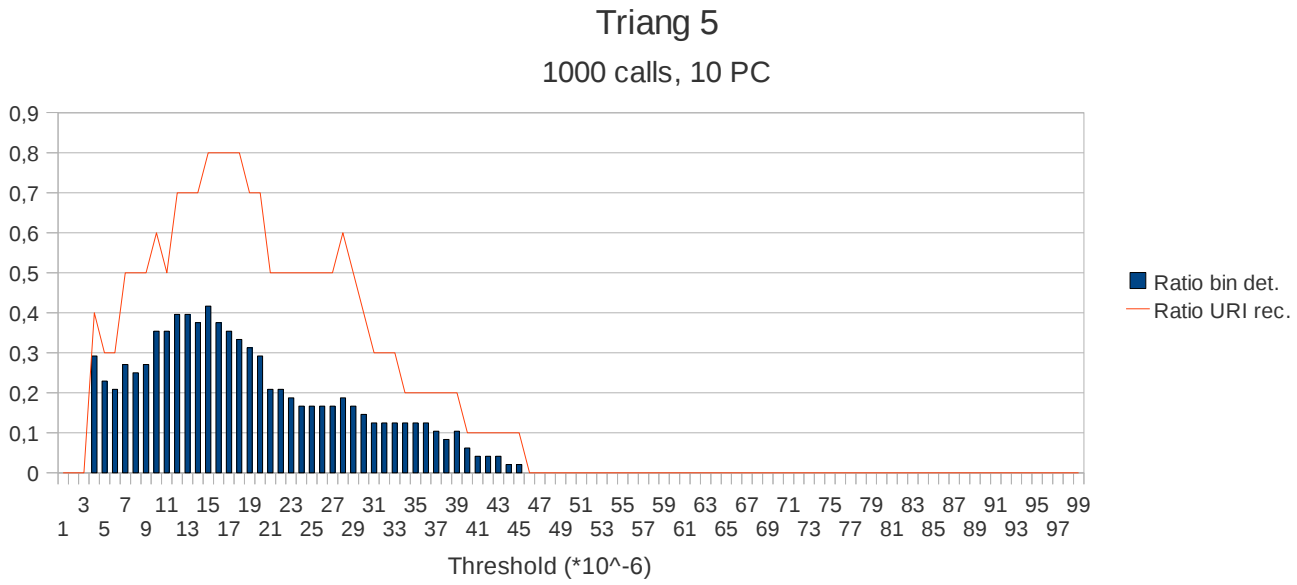


Figure E.20



Appendix F

(Triangular artificial anomalies, width 10)

Number of detections for each time-bin in a “100 thresholds-10 numbers of PCs” analysis VS ratio of global calls per time-bin on their maximum value

Figure F.1

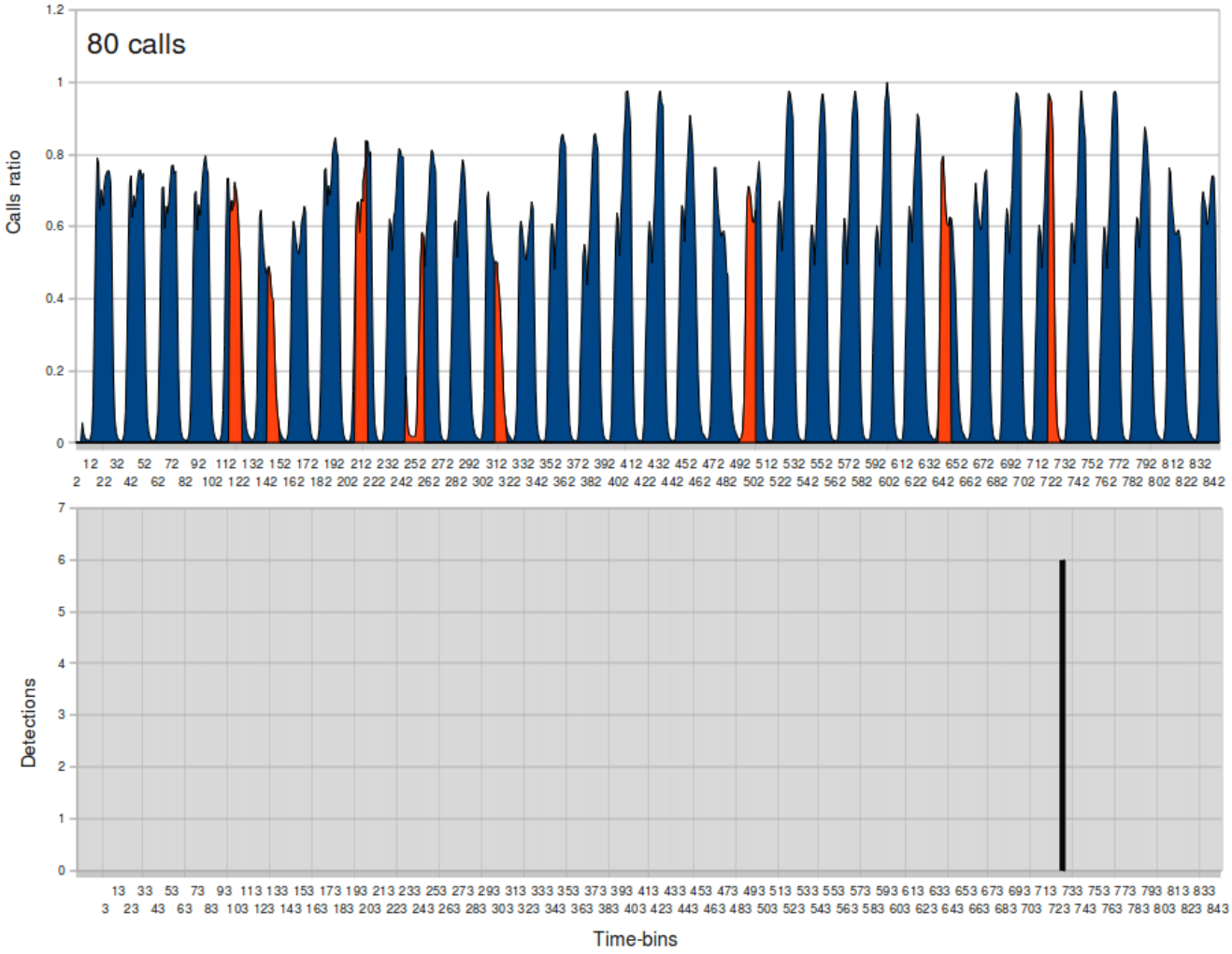


Figure F.2

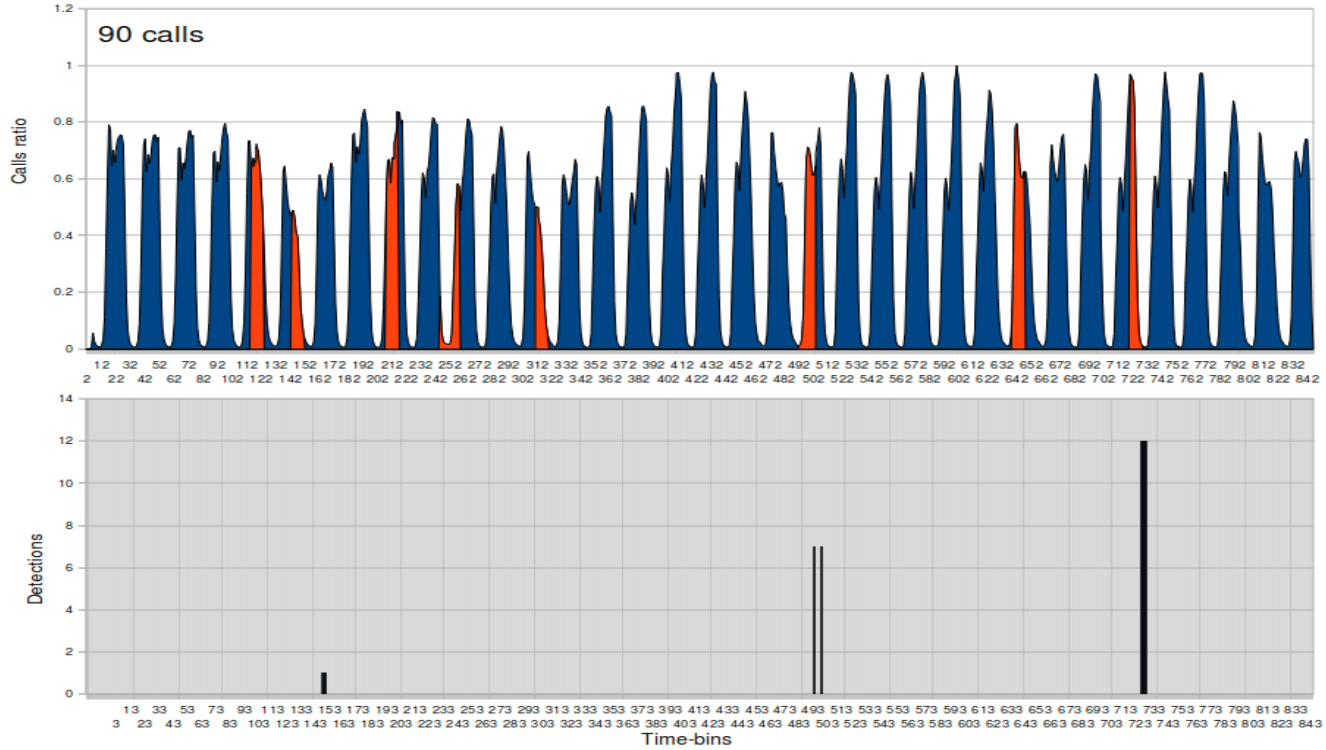


Figure F.3

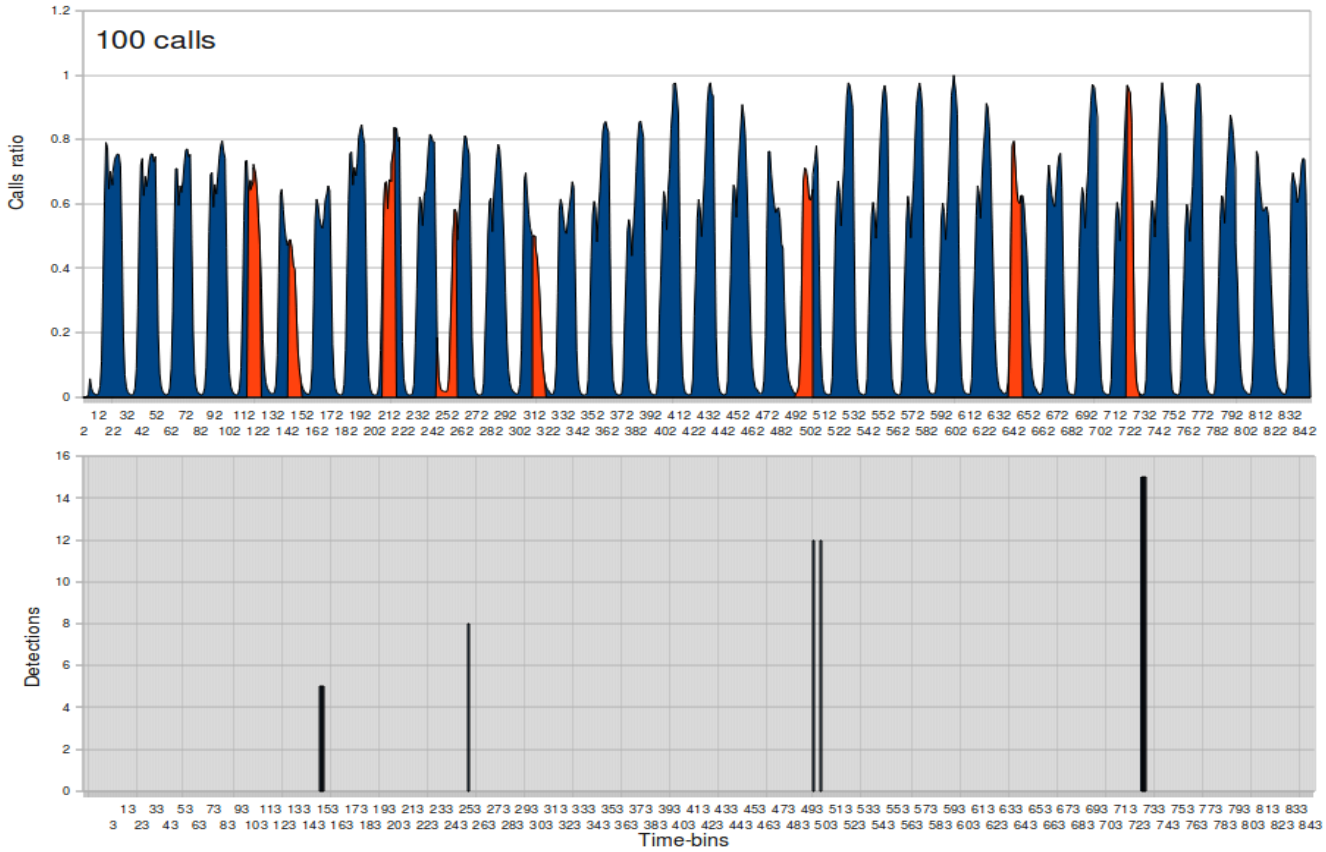


Figure F.4

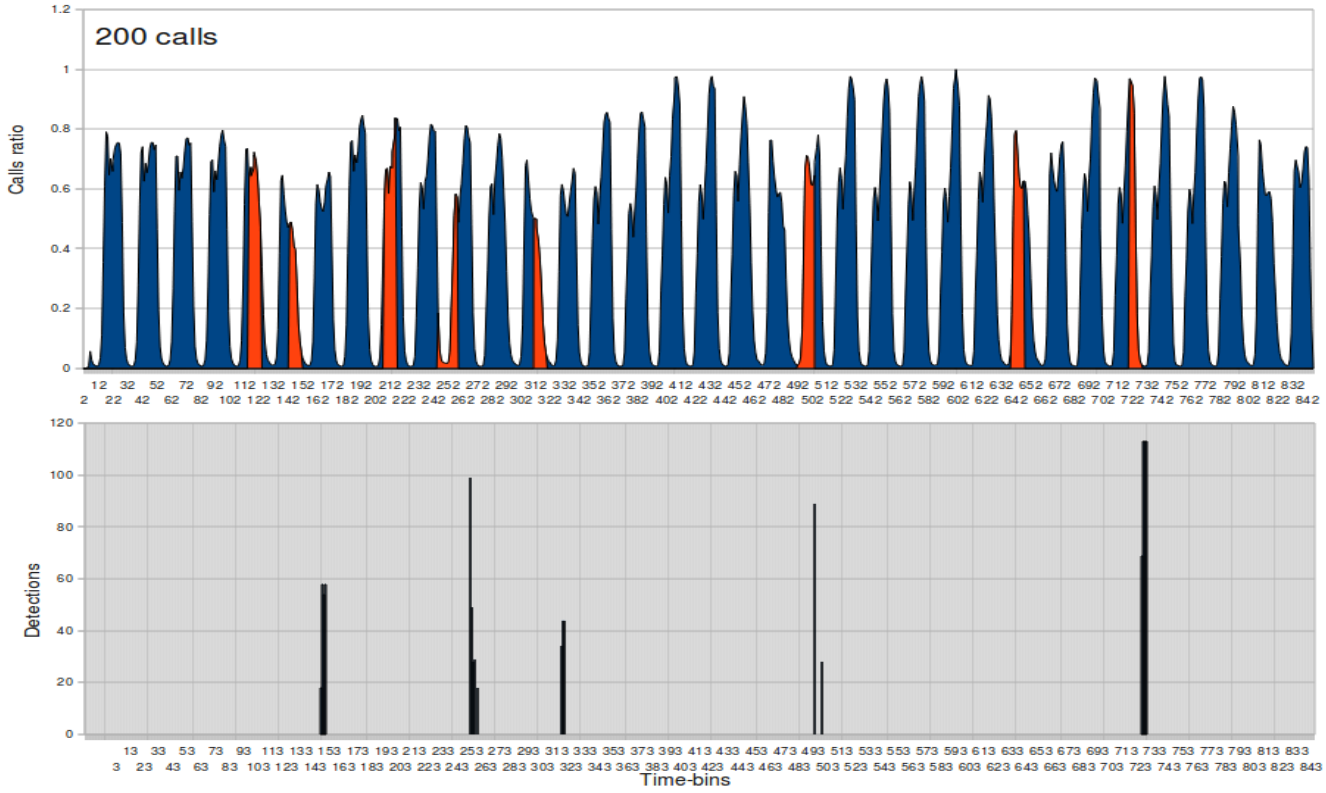


Figure F.5

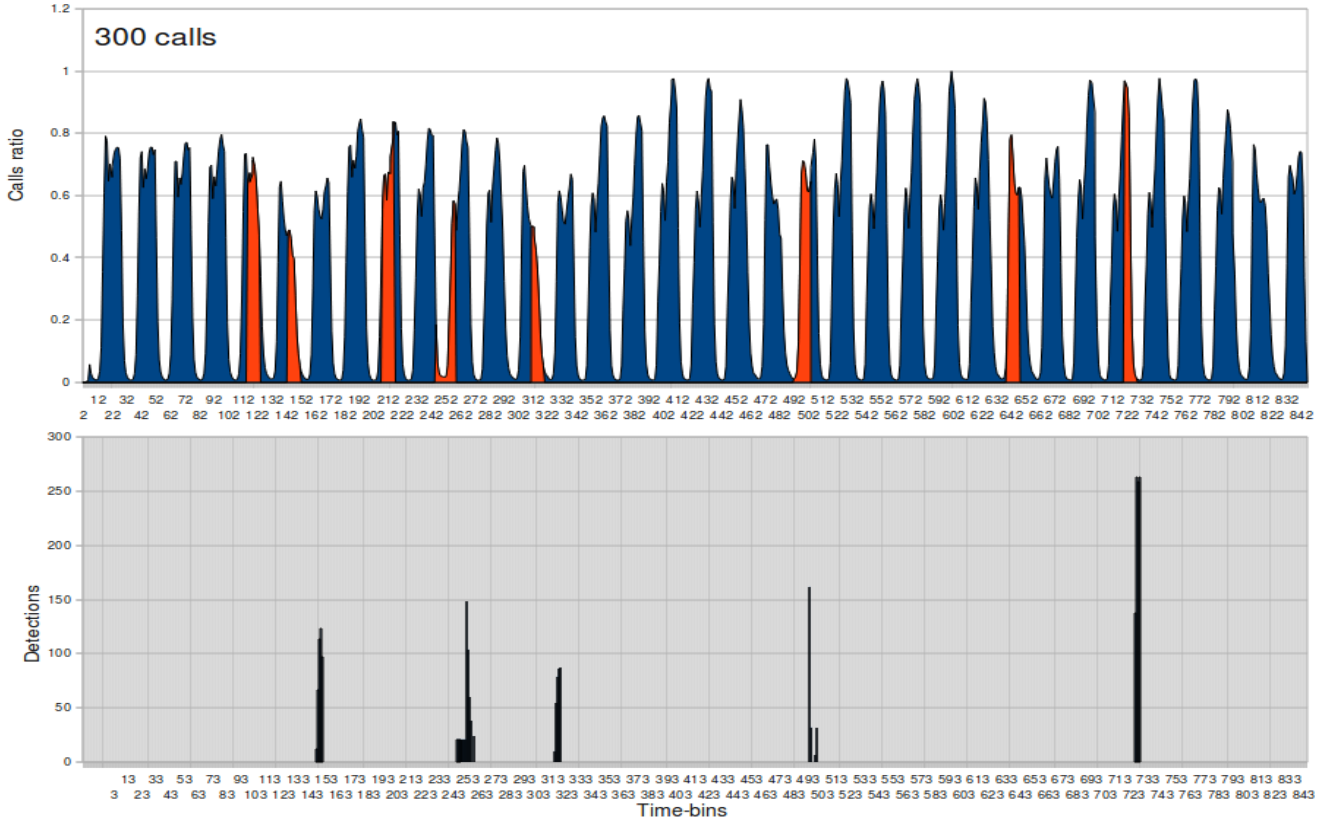


Figure F.6

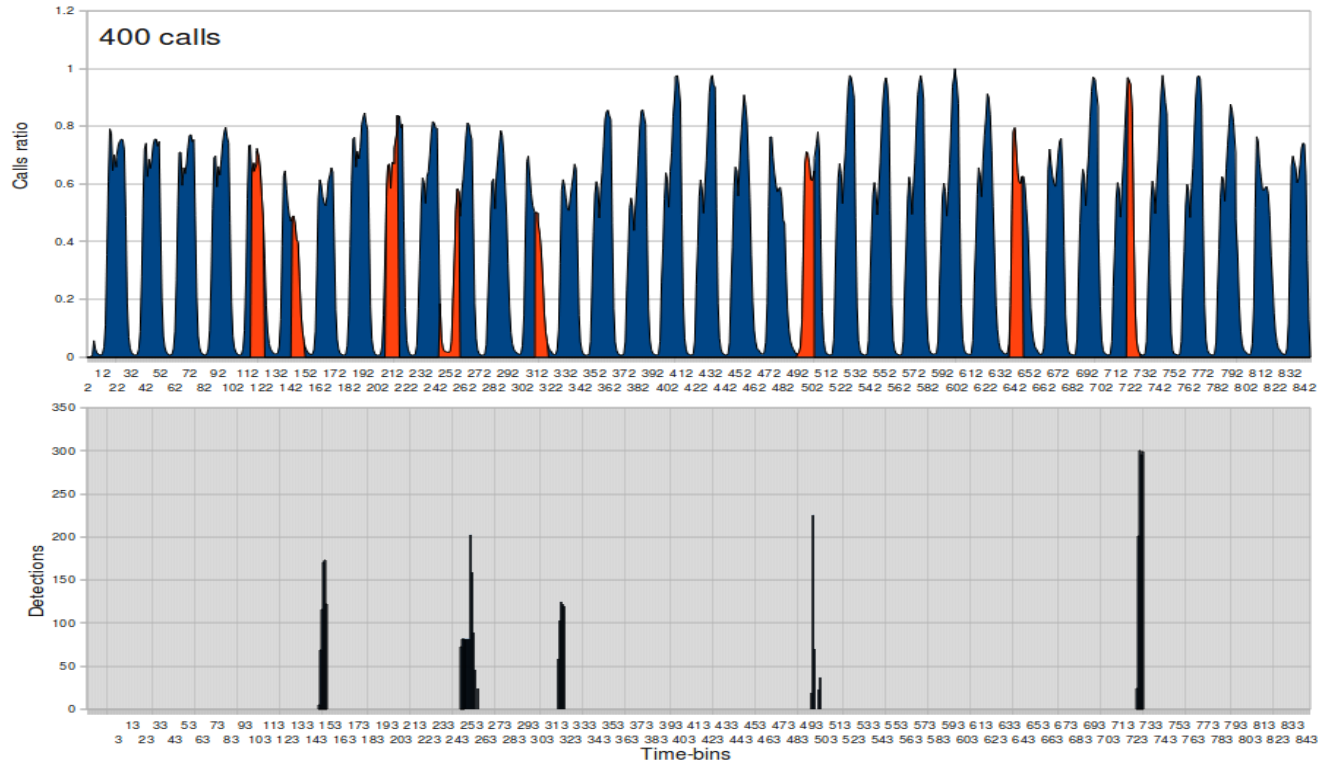


Figure F.7

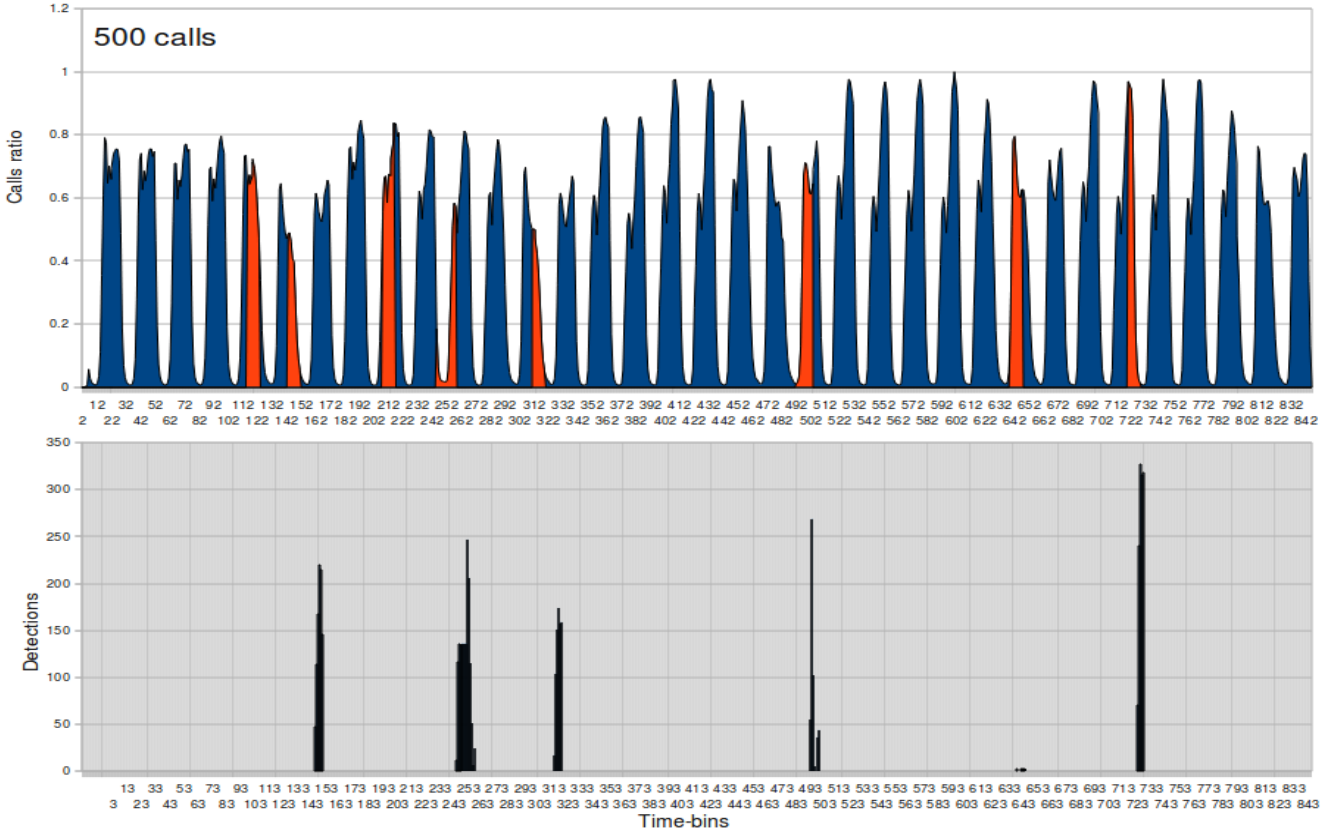


Figure F.8

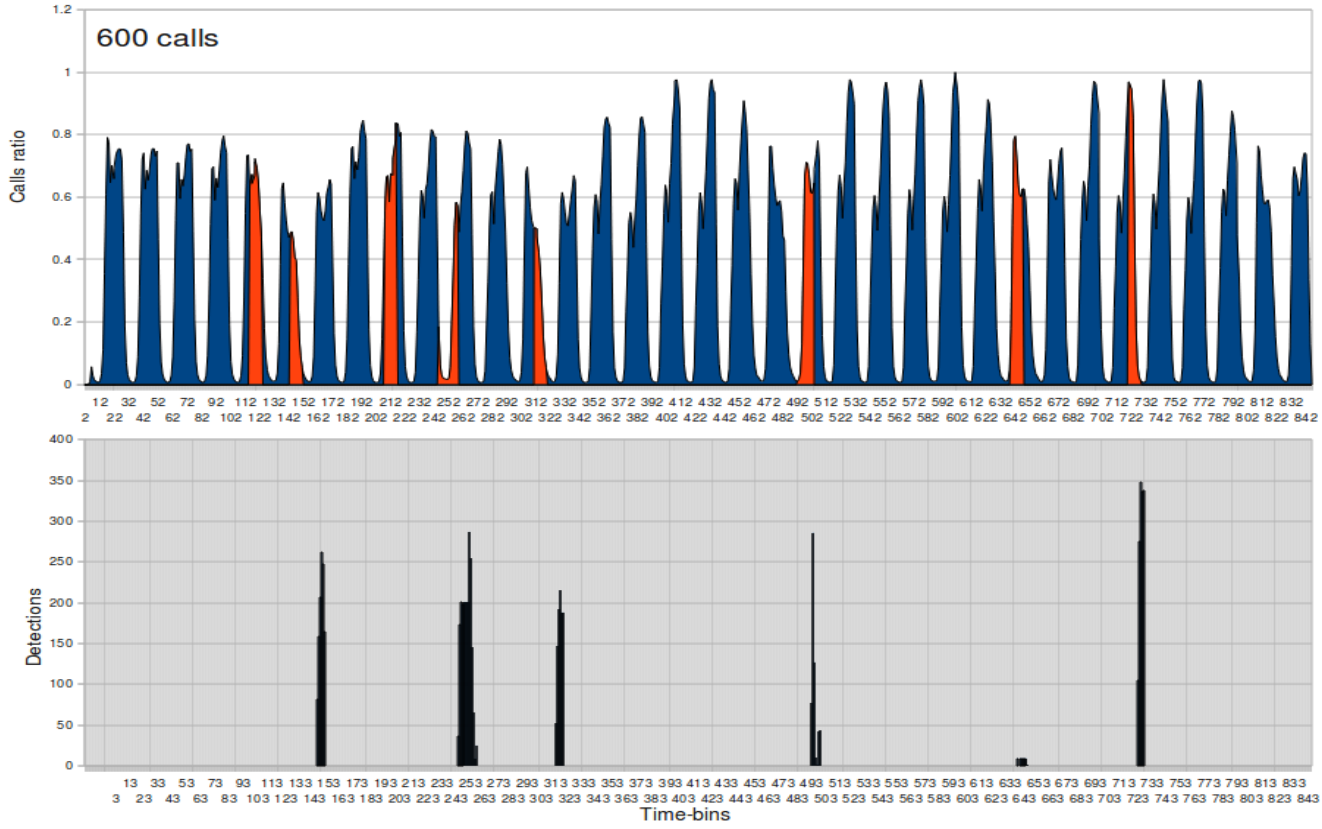


Figure F.9

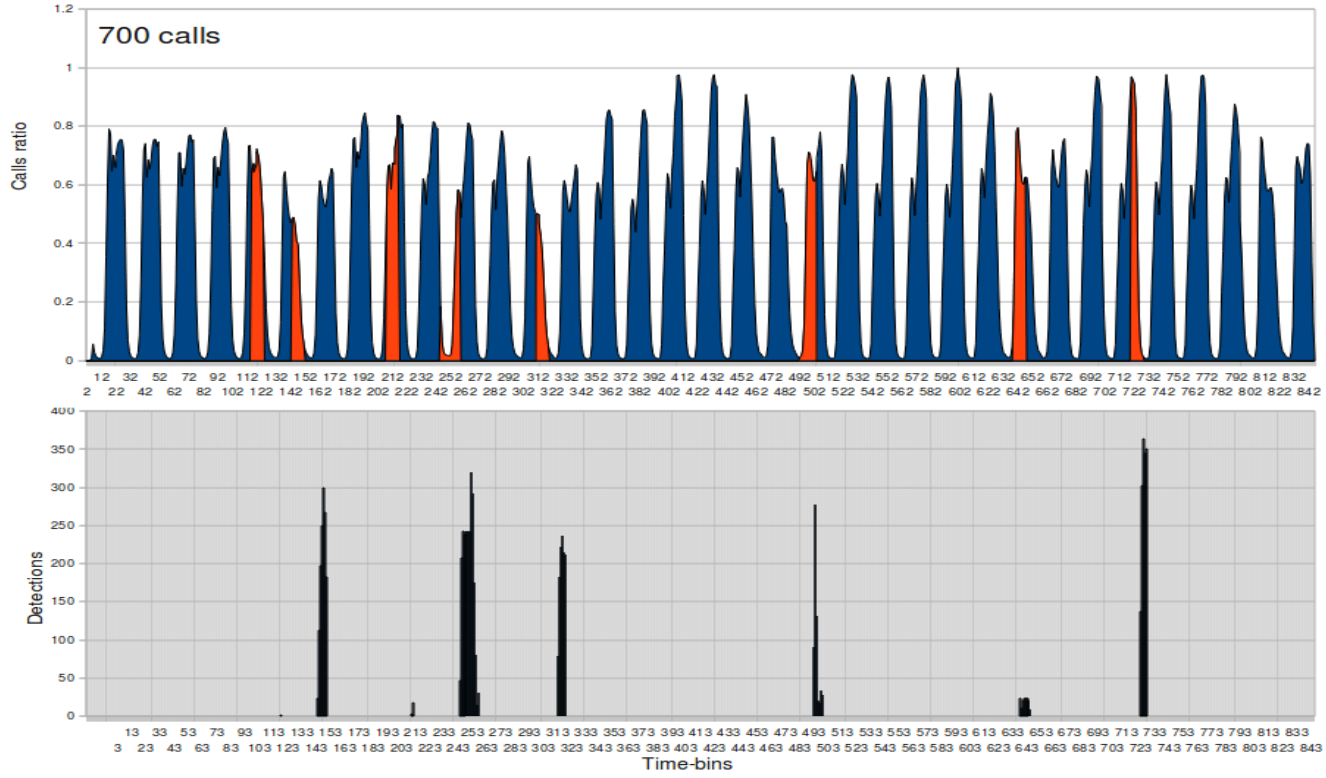


Figure F.10

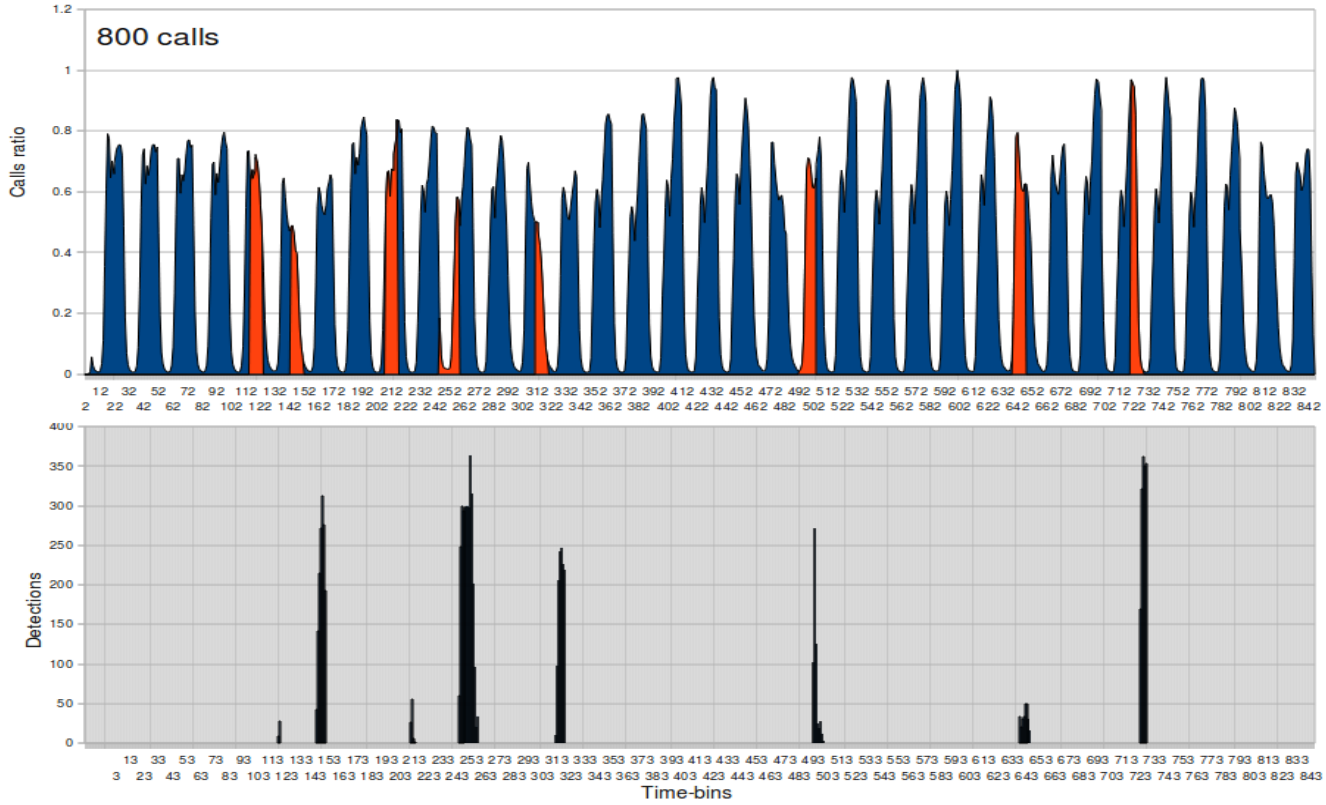


Figure F.11

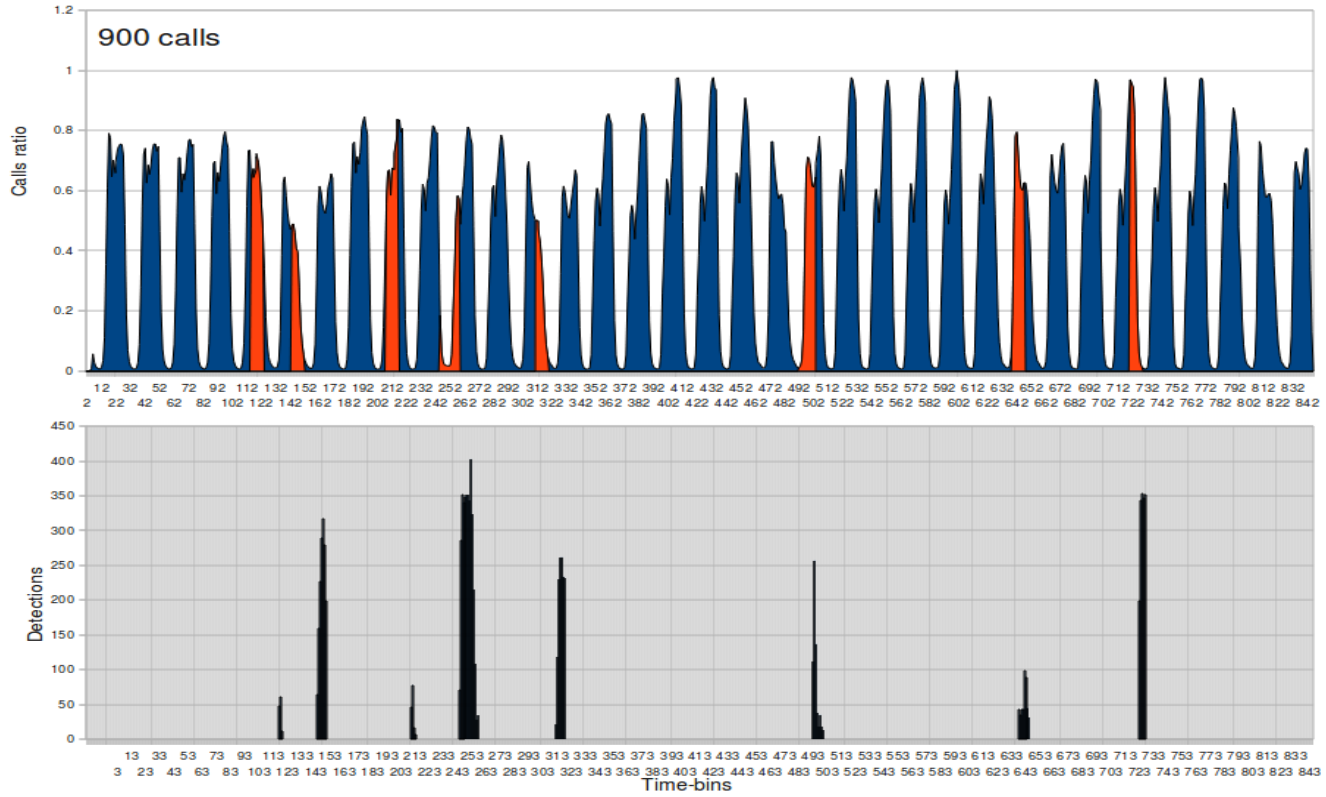
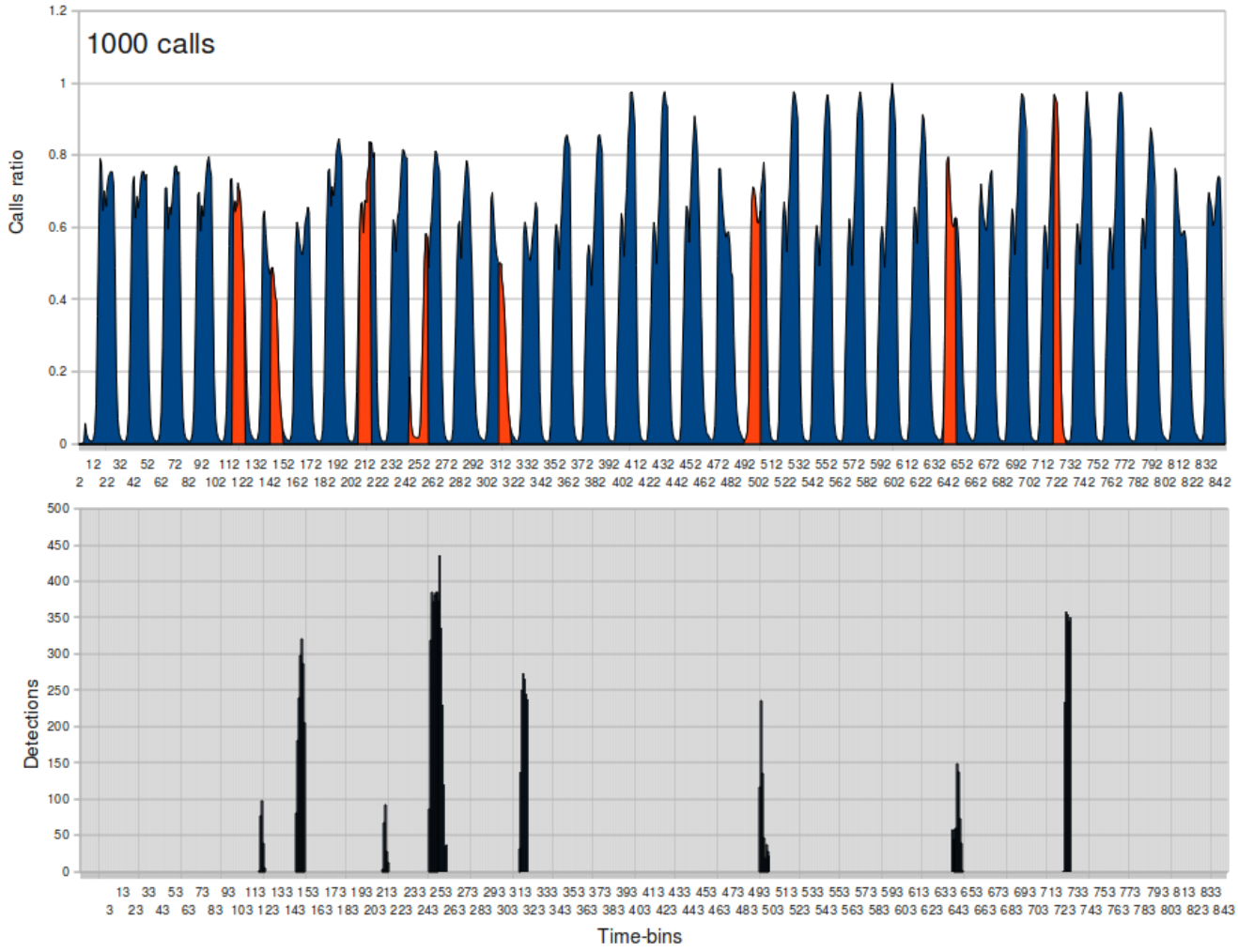


Figure F.12


1000 calls
Calls ratio
1.2
1
0.8
0.6
0.4
0.2
0
12 32 52 72 92 112 132 152 172 192 212 232 252 272 292 312 332 352 372 392 412 432 452 472 492 512 532 552 572 592 612 632 652 672 692 712 732 752 772 792 812 832
2 22 42 62 82 102 122 142 162 182 202 222 242 262 282 302 322 342 362 382 402 422 442 462 482 502 522 542 562 582 602 622 642 662 682 702 722 742 762 782 802 822 842
Detections
500
450
400
350
300
250
200
150
100
50
0
13 33 53 73 93 113 133 153 173 193 213 233 253 273 293 313 333 353 373 393 413 433 453 473 493 513 533 553 573 593 613 633 653 673 693 713 733 753 773 793 813 833
3 23 43 63 83 103 123 143 163 183 203 223 243 263 283 303 323 343 363 383 403 423 443 463 483 503 523 543 563 583 603 623 643 663 683 703 723 743 763 783 803 823 843
Time-bins

Ratio of anomalous bins detected and responsible URIs recognized (upon the total of them) with “single threshold-single number of PCs” analysis (anomaly height: 1000 calls)

Figure F.13

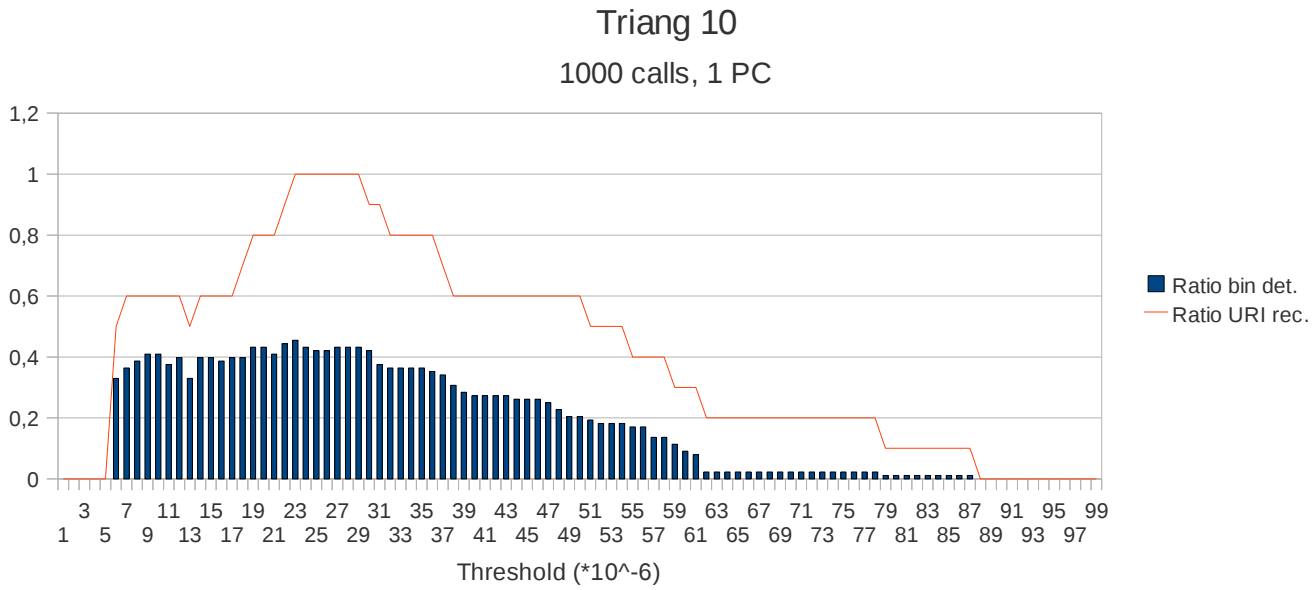


Figure F.14

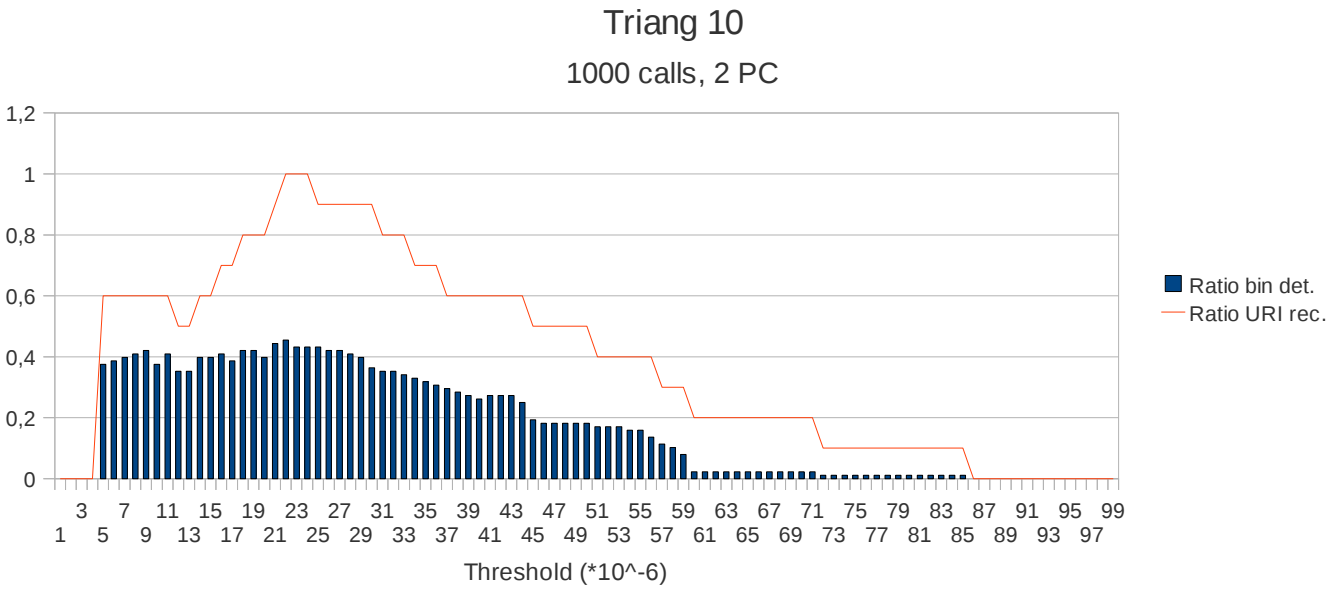


Figure F.15

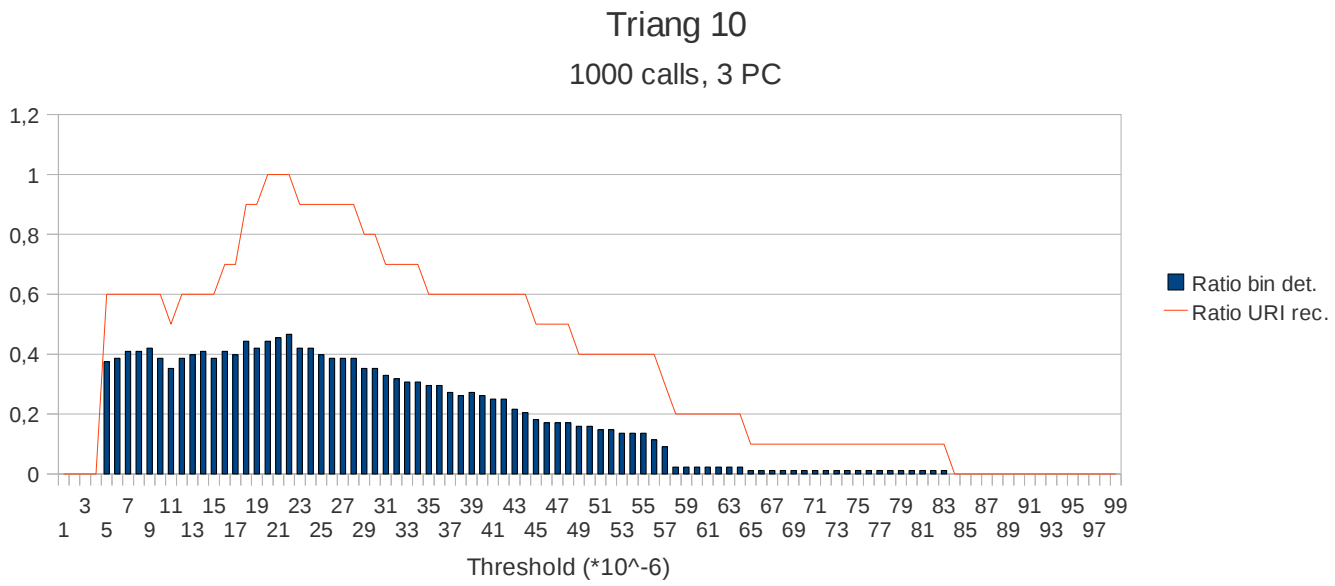


Figure F.16

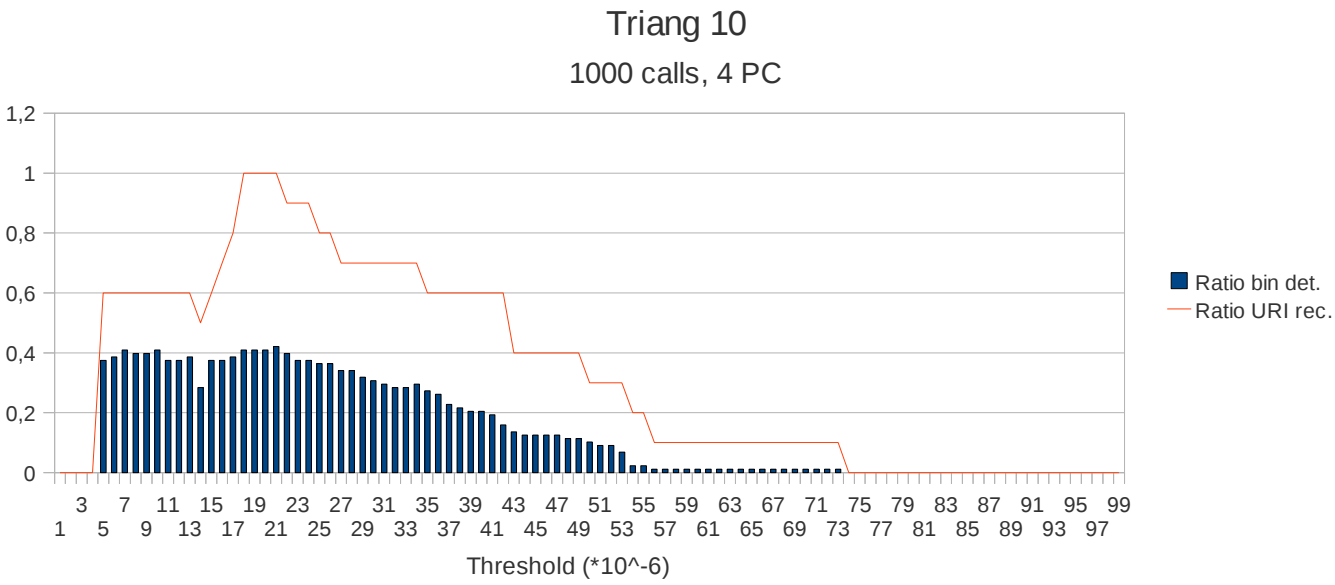


Figure F.17

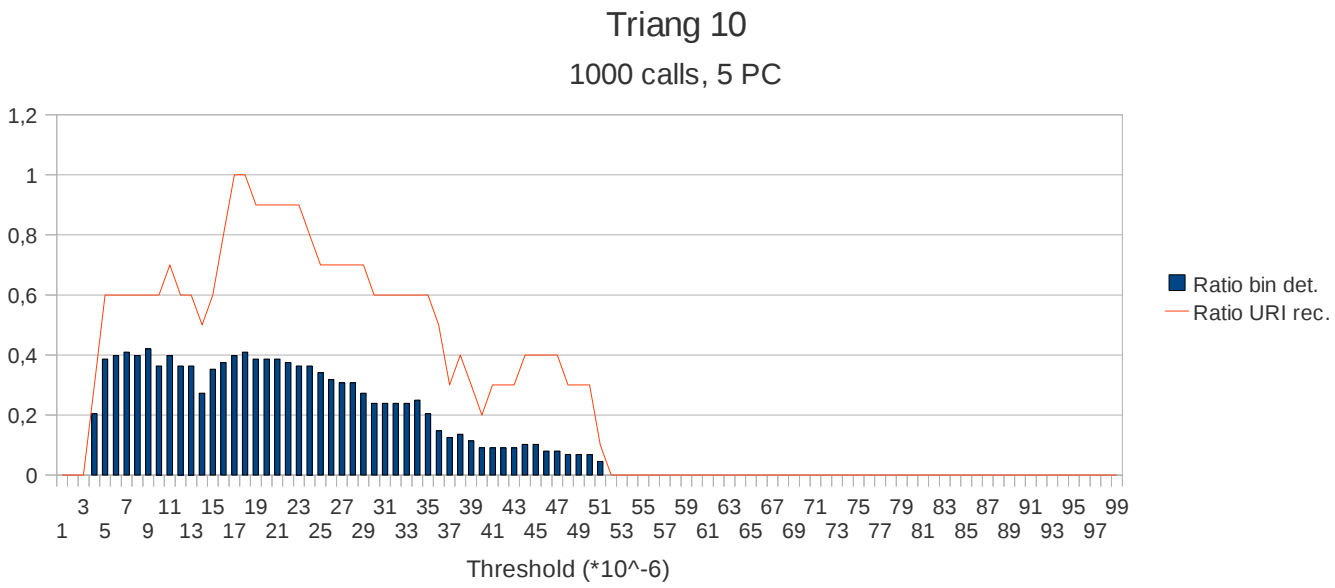


Figure F.18

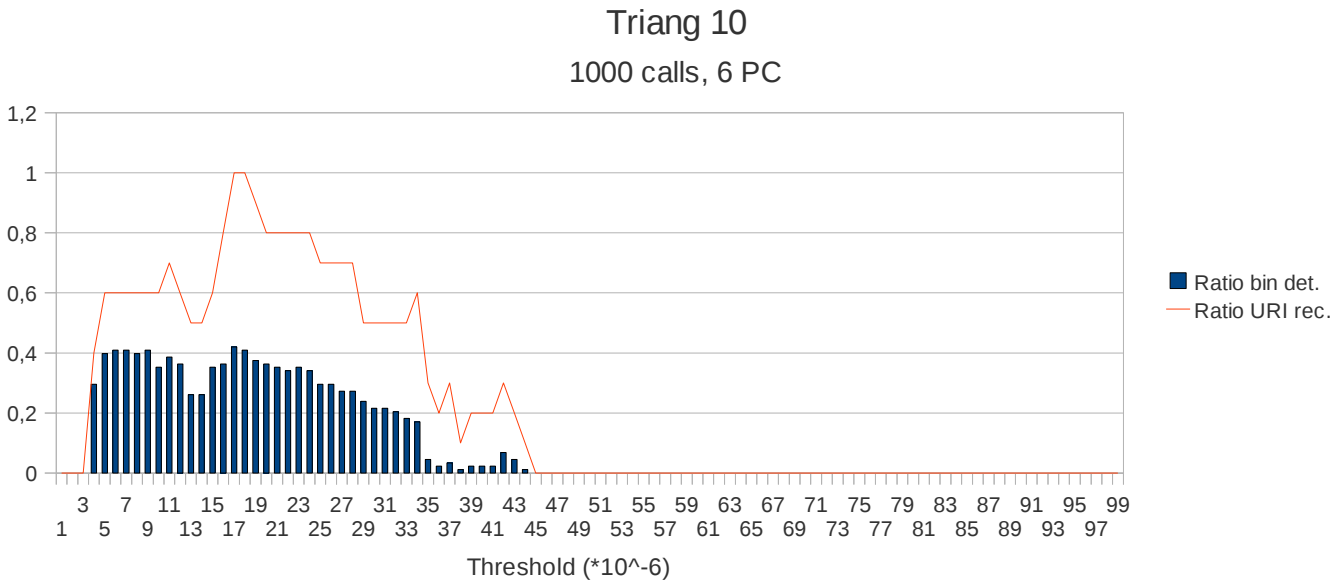


Figure F.19

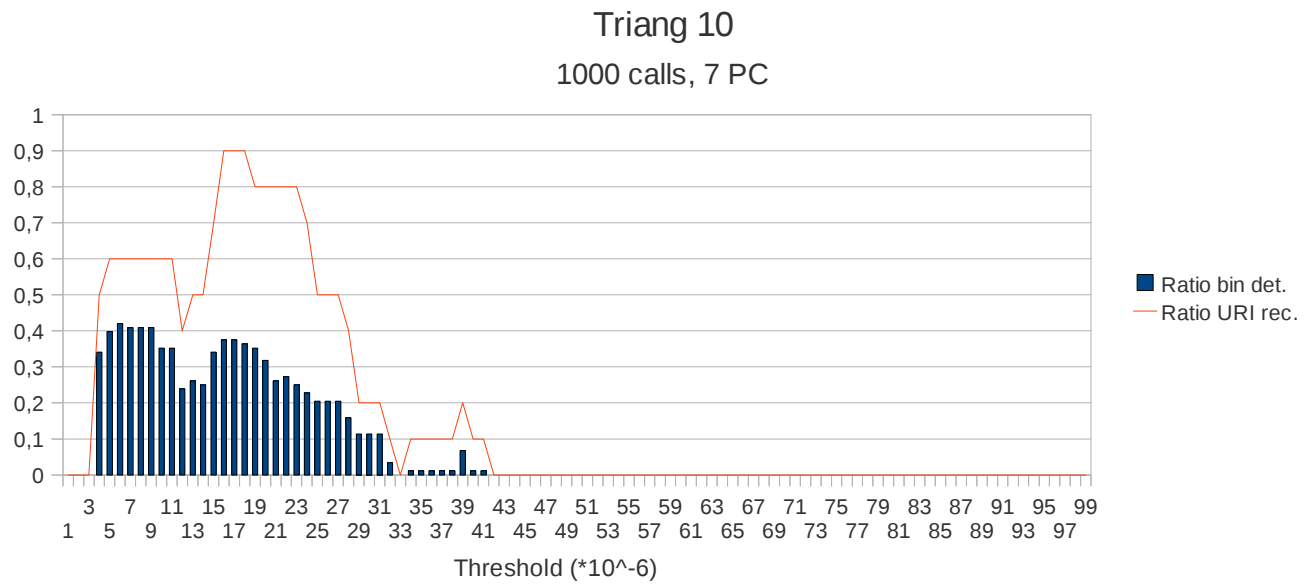


Figure F.20

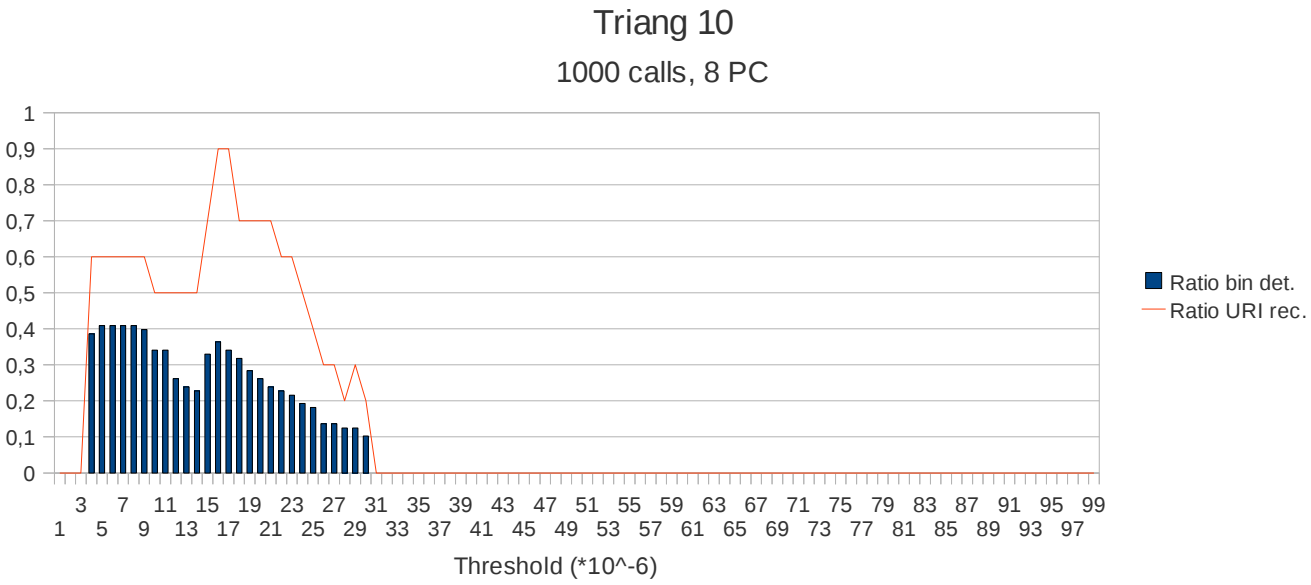


Figure F.21

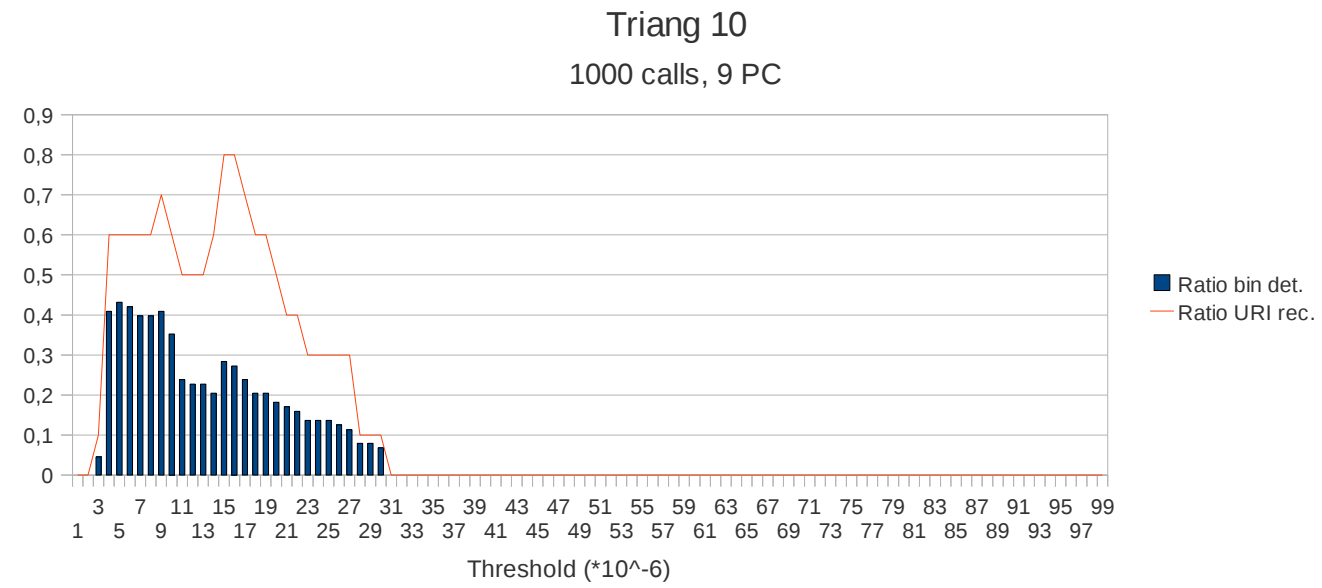


Figure F.22