Graph-aware

Anomalous Network Agent Detection

Ahmet Gercekcioglu

TU Delft

Graph-aware

Anomalous Network Agent Detection

by

Ahmet Gercekcioglu

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday January 11, 2021 at 10:00 AM.

Student number:	4042700
Faculty:	EEMCS
Master Program:	Electrical Engineering
Track:	Signals and Systems
Thesis committee:	Geert Leus
	Huijuan Wang
	Mario Coutiño

TU Delft

Abstract

Networks with a large number of participants and a highly dynamic data exchange are better off using a distributed networking system due to network failures in centralized networks. However, with the increase in distributed networking, security problems arise in distributed processes. Injection of malicious data, for example, must be dealt with by using the tools provided by detection theory. The detection probability P_d can be taken as the performance metric that we aim to optimize. In order to achieve this, one must first define a hypothesis testing problem and derive an optimization problem for P_d that is dependent on which nodes are assumed to be compromised by these malicious agents that inject data.
For the injected data, there are three different models taken into consideration for the change in values over time and nodes. For every model, we assume that the outlier data can be injected with two different attack modes. These attack modes enforce different network topology related constraints on the set of compromised nodes due to different motivations. Furthermore, additional constraints are assumed due to the limited resources of the agents.
Additionally, with the given framework, we can also derive an optimization problem that can be solved with the help of the well-known linear regression method, i.e., Lasso. The problem that arise with this method is the difficulty of implementing the network topology related constraints into this optimization problem.

In order to solve these optimization problems, several methods are combined for the relaxation and solution of the optimization problems.

From numerical evaluation, it can be observed that our empirical performance is non-negligibly lower than the theoretical performance for all three models and both attacking modes. This can be linked to the dependence of the empirical distribution to the derived subset of compromised nodes, these subsets are chosen such that the cost function is optimized. Hence, we observe that the empirical values are much higher than it is theoretically assumed, in case that the network is 'clean'.
A second factor for performance evaluation is the number of wrongly indexed nodes, it can be observed how this factor is dependent on the distribution of the energy of the outlier data over the nodes and time.

Overall, this study shows that the provided framework shows an increasing performance for an increasing outlier-to-noise energy ratio. For an energy ratio higher than 0.5, the empirical and theoretical ROC-curves are nearly perfectly saturated for all models. The number of wrongly indexed nodes for our methods is generally speaking lower compared with the Lasso-method.

Preface

I would like to thank my supervisors Geert and Mario, whose doors were always open for me, with their patience and guidance during my thesis.
Also, my parents, who supported me in all aspects possible throughout my life, deserve a special gratitude from me.

Ahmet Gercekcioglu
Delft, December 2020

Contents

1

Introduction

Networks are designed to exchange data between entities (nodes) such as network access points, sensors or agents. Networks with a large number of participants and where a highly dynamic data exchange is the norm can be prone to network failures. In a centralized network where 'a' wants to send data to 'b', there is only a single path of connection from 'a' to 'b'. If there is a connectivity problem this can lead to the loss of data. Distributed networks are more robust to failure since that the failure of a single path is backed up by the other existing paths. However, with the increasing interest in distributed processing over networks, there is also an increasing concern for the security of such processes [1]. The algorithms that are used in order to distribute the data efficiently in these networks rely heavily upon the exchange of data on these internodal paths. Protection against injection of malicious data, during the exchange on these paths, is nowadays a very relevant problem in signal processing over networks. In a smart grid for example, this can also mean the subtraction or addition of power into the network, see figure 1.1

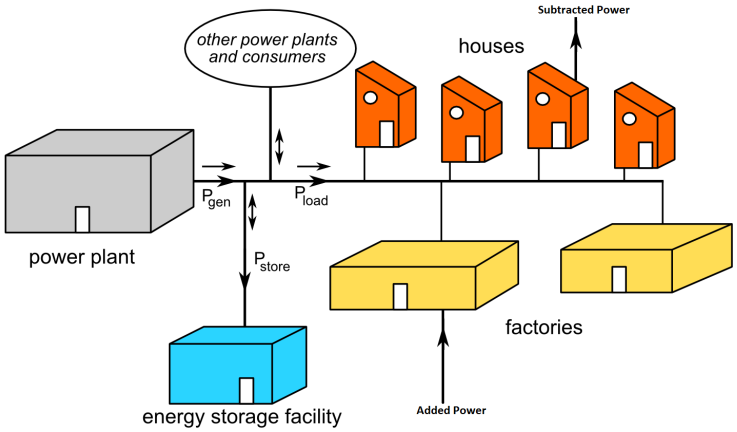


Figure 1.1: The arrows indicate the flow of the power, hence the power flowing away from the house indicates power subtraction and the power flowing into the fabric indicates power addition

Another example: with the help of controlled data injection by malicious agents it is shown that it is possible to steer the end result of a network process to a state [2] beneficial for the malicious agents. An example of such a behavior occurs in social networking, for instance, where the general opinion is usually formed through consensus dynamics where the opinion of the network participants are metricized and taken as a variable [3]. In this setting, there can be the so-called 'stubborn agents' that aim to influence their neighbors but always stick to their initial opinion in order to skew the general opinion, for example, to bias the outcome of an election or in a viral marketing campaign.
Recently, several methods have been put forth to cope with malicious data injection on a node level. In those approaches, either the nodes are screened individually to identify them as malicious agents or

using the information of their direct neighbours [4, 5]. Although the detection of a malicious agent can be done in a network through collaborative screening, i.e., each node auditing its neighboring nodes, there are instances where there are centralized observations available. Examples of such cases are smart grid networks for monitoring the power distribution [6, 7] or centralized routing for communications [8].
Regardless of whether the screening of nodes is on a global or local level, there exists an assumed difference in the behavior of the malicious agents with respect to the nodes that are considered to be honest agents, i.e., nodes that operate normally. In the case of global screening, the problem has been tackled before from an outlier detection point of view where the injected data is treated as additive outliers to the node signal values. The node signal values are then separated from the outliers with the help of an estimation process that measures the alignment of the signal values with the underlying structure of the network. Properties such as smoothness, i.e. variation of signal values across connected nodes, are exploited in order to measure the alignments of the nodal values [9]. The hypothesis testing problem for this kind of outlier detection problems is formulated as:

$$\begin{aligned} \mathcal{H}_0 &: \text{ observed data consists of noise and nodal data} \\ \mathcal{H}_1 &: \text{ observed data consists of noise, nodal data and outlier (malicious) data} \end{aligned} \tag{1.1}$$

Apart from similarity measures for the node signal to be estimated, there is also made use of the sparsity for the estimation of outliers in [9]. Thus a sparsity-based method is used for the identification of malicious nodes. However, in the classical outlier detection methods, the information from the network is not leveraged. Particularly, information that is related to the connectivity measures of the sparse outlier vector can not be imposed directly. So these approaches neglect the fact that the connectivity of the malicious agents is, in many cases, relevant to describe their behavior. For example, the malicious nodes can be arranged such that they are weakly connected to the rest of the network or they can be managed and coordinated by the agents without having a connection between them. In this thesis, we will thus focus on the detection of outliers with a structure that is dependent on the topology of the network. It is therefore required to set up a framework that identifies the anomalous nodes in the network from historical network data. To do so, we will make use of the performance metrics provided by classical detection theory along with tools of graph theory to find a mathematical expression to identify outliers with structural restrictions.

The remaining part of the thesis looks as follows. In chapter 2, we give the preliminaries about graph theory tools that are used in this work and along with these we also provide the required background of classical detection theory.

In chapter 3, the problem statement is provided and complemented with the data model. After this, we introduce three different models for the behavior of the malicious agents (outlying values). The last part in chapter 3 is dedicated to the explanation of the topology-based constraints that model the connectivity of the malicious (or compromised) nodes.

In chapter 4, we explain the mathematical derivation for the hypothesis testing problem related to the anomaly detection over networks. For all three models, there is a different hypothesis testing problem defined and thus also a different test statistic (or threshold function) must be derived. From the test statistic, we devise an optimization problem which aims to optimize the target performance metrics, which are dependent on the candidate malicious nodes. In addition, we add a discussion on why the sparsity based Lasso-method is not directly applicable, in terms of constraints, for our setting. Nevertheless, for comparison purposes, the test detector for the Lasso-based detector is also derived. A derivation for the optimization of the fitting error is also discussed in the last part.

Chapter 5 is dedicated to the convex relaxation of the optimization problems given in chapter 4. For all three models the relaxed problems are different, hence we provide three different algorithms that optimize the respective performance metrics.

In chapter 6, we will evaluate the performance of the relaxed optimization problems for all three models. Before doing this, the parameters affecting the performance are explained as well as the decisions that are taken for the variation of these parameters.

In chapter 7, future research directions are provided. Finally, in chapter 8, an overall conclusion is provided for the thesis.

2

Background

This chapter explains the concepts of graph signal processing, more specifically how network information diffusion occurs and how we can measure some basic connectivity properties. Furthermore, the concepts of detection theory and its performance metrics are explained.

2.1. Graph Theory

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of an edge set $\mathcal{E}$ and a node set $\mathcal{V}$. The edge set is the set that contains the links between the nodes. The links are assumed to be used for exchange of information. Every element in the edge set consists of two nodes that are connected. For undirected graphs, it is assumed that an edge is free of orientation. This means that the edge from node i to node j is identical to the edge from node j to node i. The edge set can also be represented with the help of an adjacency matrix or a Laplacian matrix. An unweighted adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is defined as the matrix that has a 1 at entry $[\boldsymbol{A}]_{i,j}$ only when node i and j are connected. For an unweighted adjacency matrix, these non-zero elements are equal to 1. The unweighted Laplacian matrix $\boldsymbol{L} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is defined by the following equation:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}, \tag{2.1}$$

where the matrix $\boldsymbol{D}$ is a diagonal matrix containing the degrees of the nodes, i.e., the number of links that are coupled to each node in the graph. Hence, the entry $[\boldsymbol{D}]_{ii}$ is equal to the degree of node i. The nodeset is defined as the set that contains the nodes. The cardinality of the nodeset will be indicated as N for future reference, i.e., $N = |\mathcal{V}|$. The time varying signal values for each node are represented in the vector $\boldsymbol{x}_t \in \mathbb{R}^N$ whose element values are assigned to the nodes. The subscript t indicates that the signal values for time t are given in the vector. In order to diffuse information, i.e., spread out the information across the network, a graph shift operator must be defined. A graph shift operator $\boldsymbol{\Phi}$ allows us to define the notion of information flow over a graph by replacing the signal values of each node with a linear combination of the neighbours [10]. Alternatively, we can also add a weighted value of the current signal to the linear combination in order to define the signal values for the next timeframe. Information diffusion occurs then as follows:

$$\boldsymbol{x}_{t+1} = \boldsymbol{\Phi}_{t+1}\boldsymbol{x}_t. \tag{2.2}$$

At time $t+1$, the signal values are calculated by multiplying the vector containing signal values at time t with the graph shift operator of time $t+1$, i.e. $\boldsymbol{\Phi}_{t+1}$. Possible candidates for our graph shift operator are a weighted adjacency matrix or a weighted Laplacian matrix. A well-known example, that will be used for evaluation purposes in this thesis, is the matrix that calculates the weighted average of the values from the neighbouring nodes at every timeframe. This weighted average is then assigned as the new value of the node signal value. For the calculation of the weighted average, the elements of $\boldsymbol{\Phi}_{t+1}$ consist of positive values smaller than 1 that will sum up to 1 in every row of $\boldsymbol{\Phi}$ for all t. Hence, the element values are then given as:

$$\begin{cases} 0 \leq [\boldsymbol{\Phi}_{t+1}]_{i,j} \leq 1 & \text{if } i = j \text{ or } \{i,j\} \in \mathcal{E}, \\ [\boldsymbol{\Phi}_{t+1}]_{i,j} = 0 & \text{otherwise} \end{cases} \tag{2.3}$$

and

$$\sum_{j=1}^{N} [\mathbf{\Phi}_{t+1}]_{i,j} = 1 : \forall \quad i \in \{1, ...N\}. \tag{2.4}$$

A matrix with these properties is also called a right stochastic matrix [11].

Furthermore, in order to exploit the structure of the graph, we will make use of graph connectivity measures. These connectivity measures are based on the cut of a subset of the node set $\mathcal{V}$. The cut of a subset is defined as the summation of the (weighted) edges 'leaving' that subset. Let us define a subset of the node set as $\mathcal{W} \subset \mathcal{V}$ and the support of the subset as

$$[\mathbf{1}_{\mathcal{W}}]_i = \begin{cases} 1, & \text{if } i \in \mathcal{W} \\ 0, & \text{if } i \notin \mathcal{W} \end{cases} \tag{2.5}$$

Mathematically, the cut $C(\mathbf{1}_{\mathcal{W}})$ can then be formulated as follows [12]:

$$C(\mathbf{1}_{\mathcal{W}}) = \mathbf{1}_{\mathcal{W}}{}^{T} \boldsymbol{L} \mathbf{1}_{\mathcal{W}}. \tag{2.6}$$

The Laplacian matrix can be unweighted or there can be a weight assigned to the edges of the graph. In this thesis, we work with the unweighted Laplacian for the graph cut. An illustrative example is given in figure 2.1, where the edge subset that defines the cut for a given subset of nodes is indicated in red. In the example given in figure 2.1, the cut will be equal to 4. .

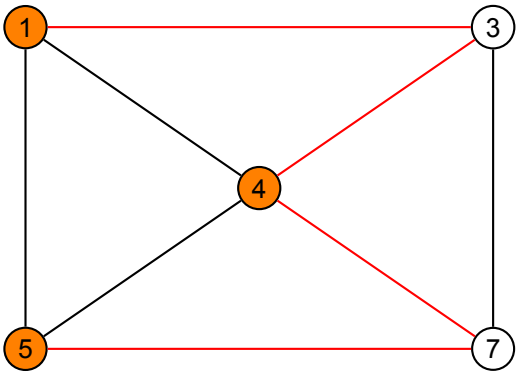


Figure 2.1: For the subset of nodes that are filled in orange, the subset of edges that belong to the cut are colored in red

2.2. Detection Theory

Detection theory lies at the core of signal processing. The main concern of detection theory is deciding whether a certain event of interest occurs or not. Mathematically, this event is then denoted as an hypothesis. A well-known detection problem is to detect whether a certain signal $\boldsymbol{s}$ is present in observed data that is embedded in noise $\boldsymbol{n}$. This problem can then be mathematically denoted as a binary hypothesis problem where the null hypothesis $\mathcal{H}_0$ is then the absence of the signal $\boldsymbol{s}$ and thus only the presence of noise $\boldsymbol{n}$. And the alternative hypothesis is then the situation where there is noise and signal present in the observation $\boldsymbol{y}$, i.e.,

$$\begin{aligned} \mathcal{H}_0 : \quad & \boldsymbol{y} = \boldsymbol{n} \\ \mathcal{H}_1 : \quad & \boldsymbol{y} = \boldsymbol{s} + \boldsymbol{n} \end{aligned} \tag{2.7}$$

In case that we want to choose the correct hypothesis for our decision, the observed data needs to be exploited as efficient as possible. So that the rate of correct decisions is maximized for multiple realisations of the problem. We can make use of the statistical properties and model the hypotheses under certain distributions. For example, if the noise in the elements of the observed data $\boldsymbol{y}$ is zero mean while Gaussian, $\boldsymbol{n} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$, and the observed signal $\boldsymbol{s}$ is deterministic, we can then denote the hypothesis under these statistical models as:

$$\begin{aligned} \mathcal{H}_0 : \quad & \boldsymbol{y} \sim \mathcal{N}(0, \sigma^2 \mathrm{I}) \\ \mathcal{H}_1 : \quad & \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{s}, \sigma^2 \mathrm{I}) \end{aligned} \tag{2.8}$$

If we define the distribution under $\mathcal{H}_0$ as $P(\boldsymbol{y};\mathcal{H}_0)$ and the distribution under $\mathcal{H}_1$ as $P(\boldsymbol{y};\mathcal{H}_1)$. We can set up a framework for a detector dependent on the false alarm probability P_{fa} and the detection probability P_d. The false alarm probability P_{fa} is defined as the probability that $\mathcal{H}_1$ is decided when $\mathcal{H}_0$ is true, $P(\mathcal{H}_1;\mathcal{H}_0)$. And the detection probability is defined as the situation where $\mathcal{H}_1$ is decided while this is true, $P(\mathcal{H}_1;\mathcal{H}_1)$. The Neyman-Pearson (NP) theorem shows that the likelihood ratio test returns the most 'powerful' test [13]. The power of a test is defined as the detection probability P_d and the most powerful test is defined as the test that has the highest P_d for a given risk of P_{fa}. The likelihood ratio test, $L(\boldsymbol{y})$, must be compared with a detection threshold γ in order to make a decision between the two hypotheses, i.e.,

$$L(\boldsymbol{y}) = \frac{P(\boldsymbol{y};\mathcal{H}_1)}{P(\boldsymbol{y};\mathcal{H}_0)} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \gamma. \tag{2.9}$$

The detection threshold γ is then found with the help of the false alarm probability:

$$P_{fa} = \int_{\{\boldsymbol{y}:L(\boldsymbol{y})>\gamma\}} P(\boldsymbol{y};\mathcal{H}_1) \quad d\boldsymbol{y}. \tag{2.10}$$

By mathematical manipulation of the likelihood ratio, we can then derive a threshold function, which is also called as the test statistic interchangeably, $T(\boldsymbol{y})$, which allows us to derive a mathematical expression for the detection and false alarm probability as:

$$\begin{aligned} P_d &= P(T(\boldsymbol{y}) > \gamma';\mathcal{H}_1) \\ P_{fa} &= P(T(\boldsymbol{y}) < \gamma';\mathcal{H}_0) \end{aligned} \tag{2.11}$$

where γ' is the newly found threshold after simplification. In case that the distribution of $T(\boldsymbol{y})$ is known under both hypotheses, it is simpler to derive a mathematical expression that is dependent on $\boldsymbol{y}$ for both of the probabilities. For the above given example, assuming that $\boldsymbol{s}$ is known and deterministic, a matched filter is derived for (2.7).

$$T(\boldsymbol{y}) = \boldsymbol{y}^T\boldsymbol{s}. \tag{2.12}$$

And the distribution of the test, under both hypotheses, is given as follows:

$$\begin{aligned} \mathcal{H}_0: \quad & T(\boldsymbol{y}) \sim \mathcal{N}(0, \sigma^2\boldsymbol{s}^T\boldsymbol{s}) \\ \mathcal{H}_1: \quad & T(\boldsymbol{y}) \sim \mathcal{N}(\boldsymbol{s}^T\boldsymbol{s}, \sigma^2\boldsymbol{s}^T\boldsymbol{s}) \end{aligned} \tag{2.13}$$

These expressions give us the following expressions for the detection and false alarm probability

$$P_d = Q\left(\frac{\gamma' - \boldsymbol{s}^T\boldsymbol{s}}{\sqrt{\sigma^2\boldsymbol{s}^T\boldsymbol{s}}}\right), \tag{2.14}$$

$$P_{fa} = Q\left(\frac{\gamma'}{\sqrt{\sigma^2\boldsymbol{s}^T\boldsymbol{s}}}\right). \tag{2.15}$$

In (2.14) and (2.15) the Q-function is defined as the right tail distribution function of the standard normal distribution. The problem gets more complicated when the statistical models used for the hypotheses contain unknown parameters. In these cases, what must be done first is to find an estimate for these unknown parameters. To this end, we can make use of the generalized likelihood ratio test which replaces the unknown parameters with their maximum likelihood estimation from the collected data, i.e.,

$$\boldsymbol{s}^* = \underset{\boldsymbol{s}}{\operatorname{argmax}}\, P(\boldsymbol{y};\boldsymbol{s},\mathcal{H}_1). \tag{2.16}$$

An overview of the performance is presented with the help of a receiver operating characteristic (ROC)-curve that plots the P_d for $P_{fa} \in [0,1]$.

The mathematical modeling used for the detection problem in this thesis will be simplified to the above mentioned formulation in (2.7) while a certain structure of the unknown signal to be detected is taken into account.
In the next chapter, we provide the problem statement together with the data models of the signal and outliers.

3

Anomaly Detection in Networks

The goal of this chapter is to explain the problem that will be dealt with in this thesis. A formal statement of the problem is provided first. Secondly, the data model that will be used is explained in section 3.2. Then, we introduce three different models for the outliers. And in the last part, we will explain how certain connectivity measures for the graph are motivated and taken as constraints for the support set of the compromised node set.

3.1. Problem Statement

Let us consider a network modeled by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each element in the node set $\mathcal{V}$ represents a network element and $\mathcal{E}$ is the edge set that represents the relationship between the elements. In this thesis, we will focus on distributed networking where $\{i, j\} \in \mathcal{E}$ indicates that nodes i and j exchange data, in other words there is a communication between the elements. Despite the distributed nature of the network, an active surveillance in the form of a centralized observation will be available for security purposes. In figure 3.1 an illustration is given for the situation under study. An attack can be carried out by an unknown third party that may cause anomalies in the data or data

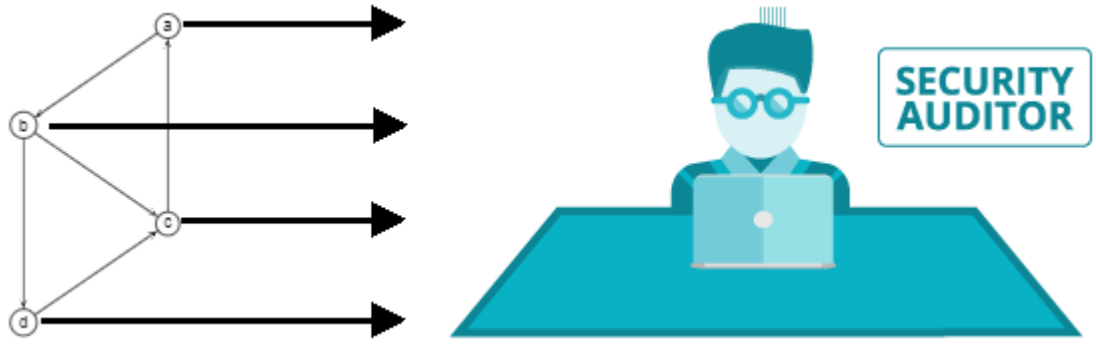


Figure 3.1: An illustration of an auditor observing the data

observation. An anomaly occurs by a malicious injection that results in data outliers. A case example is given in [14] for cyber-physical systems, where sensor readings are altered. A more specific example could be the injection of false data during the observation of the state in microgrids [15]. Thereby manipulating the state estimation procedure. In these two examples, the false data injection alters the observation $\boldsymbol{y}_t$ of the data at time t by adding outliers $\boldsymbol{o}_t$ to the observation, i.e.,

$$\boldsymbol{y}_t = \boldsymbol{x}_t + \boldsymbol{o}_t + \boldsymbol{e}_t. \tag{3.1}$$

In equation (3.1), $\boldsymbol{x}_t$ is defined as the vector containing the signal values of the network at time t and $\boldsymbol{e}_t$ is the observation noise.

Apart from false data injection during observation, it can also occur that the signal values are directly influenced by a third party during state transition, i.e.,

$$\boldsymbol{x}_t = \boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{o}_t + \boldsymbol{n}_t. \tag{3.2}$$

Here, $\boldsymbol{A}_t$ is defined as the state transition matrix at time t and $\boldsymbol{n}_t$ is the added noise during transmission. In [16], it is explained that analyzing the signal values of consensus-based distributed estimation during direct false data injection can prevent future attacks by designing an effective defensive measure. When talking about false data injection, outliers or anomalies, the model in (3.2) will be assumed in the continuation of the thesis. It is the task of the auditor to detect the anomaly in order to prevent further influence of the third party malicious agents to the network. In case that we assume a noise-free observation and noisy transmission of signal values, the hypothesis test for the detection problem can then be formulated as:

$$\begin{aligned} \mathcal{H}_0 : \quad & \boldsymbol{y}_t = \boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{n}_t \\ \mathcal{H}_1 : \quad & \boldsymbol{y}_t = \boldsymbol{A}_t \boldsymbol{x}_{t-1} + \boldsymbol{o}_t + \boldsymbol{n}_t. \end{aligned} \tag{3.3}$$

For $T+1$ consecutive observed timeframes that start at time t, the hypothesis test is then formulated as:

$$\begin{aligned} \mathcal{H}_0 : \quad & \boldsymbol{y}_{t+i} = \boldsymbol{A}_{t+i} \boldsymbol{x}_{t+i-1} + \boldsymbol{n}_{t+i} : \forall \quad i \in \{0, \ldots . T\} \\ \mathcal{H}_1 : \quad & \boldsymbol{y}_{t+i} = \boldsymbol{A}_{t+i} \boldsymbol{x}_{t+i-1} + \boldsymbol{o}_{t+i} + \boldsymbol{n}_{t+i} : \forall \quad i \in \{0, \ldots . T\}. \end{aligned} \tag{3.4}$$

The challenge is to set up a framework that identifies the compromised nodes and estimates the outliers. The support of vector $\boldsymbol{o}_t$ is defined as the set $\mathcal{A}$ that consists of the indices corresponding to nonzero entries in $\boldsymbol{o}_t$. Hence $\mathcal{A}$ is a subset of $\mathcal{V}$ containing the nodes that are compromised. Sparsity of the outliers has been exploited in the past [9] [17] in order to estimate the outlying values and thus also $\mathcal{A}$. In addition to the sparsity of $\mathcal{A}$, in this thesis, we will exploit prior information on the connectivity of the malicious agents leading to structural constraints for $\mathcal{A}$, along with assumptions about the behavior over time of $\boldsymbol{o}_t$.

Considering these, a framework is required that must identify the support set $\mathcal{A}$ based on metrics used in detection theory, while graph-related constraints are enforced on the candidate set of compromised nodes. In other words, the task is to estimate $\mathcal{A}$ that optimizes a given detection performance metric, while the new constraints are taken into account. In section 2.2, it is explained that the likelihood ratio test is derived such that the true positive $P(\mathcal{H}_1; \mathcal{H}_1)$ is maximized for a given risk of false positive $P(\mathcal{H}_1; \mathcal{H}_0)$. The reason that we define the compromised state as $\mathcal{H}_1$ and not as $\mathcal{H}_0$ is explained by the consequences that arise as a result of the detection or misdetection of this compromised state of the network. The consequences for an undetected anomaly, are a bigger concern than the minor consequences that arise by a falsely claimed anomaly, i.e., unnecessary temporal network process inhibition. Thus, we would like to increase our detection probability of anomaly for a given risk of falsely claimed state of anomaly. Otherwise, if we would have preferred to decrease the chances of a falsely claimed anomaly and thus increase the chances of detecting the 'normal' state, then we would have defined $\mathcal{H}_1$ as the 'normal' state. This is the reason that the alternative hypothesis $\mathcal{H}_1$ is defined as the case where an anomaly occurs.
Therefore, we need to derive an optimization algorithm that solves the following:

$$\begin{aligned} \max_{\mathbf{1}_{\tilde{\mathcal{A}}}} \quad & f_{P_d}(\mathbf{1}_{\tilde{\mathcal{A}}}) \\ \text{subject to} \quad & \mathbf{1}_{\tilde{\mathcal{A}}} \in \mathcal{C} \end{aligned} \tag{3.5}$$

The vector $\mathbf{1}_{\mathcal{A}t}$ is defined as the support vector of $\boldsymbol{o}_t$, i.e.,

$$[\mathbf{1}_{\mathcal{A}}]_i = \begin{cases} 1, & \text{if } i \in \mathcal{A} \\ 0, & \text{if } i \notin \mathcal{A} \end{cases}. \tag{3.6}$$

The vector, $\mathbf{1}_{\tilde{\mathcal{A}}}$ is the stacking of $\mathbf{1}_{\mathcal{A}t}$ for multiple values of t.
In the optimization problem of (4.24) f_{P_d} is the cost function that must be maximized in order to optimize the detection probability P_d. Furthermore, $\mathcal{C}$ is the constraint set that is defined by the sparsity and the graph-structure related constraints of $\mathcal{A}$.

3.2. Networked Data Model

Let us recall the data model for the signal values as:

$$\boldsymbol{x}_t = \boldsymbol{\Phi}_t \boldsymbol{x}_{t-1} \tag{3.7}$$

The matrix $\boldsymbol{\Phi}_t$ was previously identified in section 2.1 as the graph shift operator. In section 3.1, $\boldsymbol{\Phi}_t$ is defined as the state transition matrix $\boldsymbol{A}_t$ and we will assume that it is known at all times. Due to uncertainties in the network it is assumed that there is noise added to the signal, i.e.,

$$\boldsymbol{x}_t = \boldsymbol{\Phi}_t \boldsymbol{x}_{t-1} + \boldsymbol{n}_t \tag{3.8}$$

where $\boldsymbol{n}_t$ is assumed to be additive white Gaussian noise, $\boldsymbol{n}_t \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$, that is added to the signal due to its uncertainties in the network at time t. In case that there is a compromised node, let's say the i'th node, $[\boldsymbol{x}_t]_i$ will be altered by a malicious agent during state transition, causing an abnormal behavior in $[\boldsymbol{x}_t]_i$. These assumptions result in the following expression for the signal:

$$\boldsymbol{x}_t = \boldsymbol{\Phi}_t \boldsymbol{x}_{t-1} + \boldsymbol{o}_t + \boldsymbol{n}_t. \tag{3.9}$$

In (3.9), the vector $\boldsymbol{o}_t$ corresponds to the outlying values of $\boldsymbol{x}_t$. Hence an entry in $[\boldsymbol{o}_t]_i$ is non-zero only if $i \in \mathcal{A}$. For the observation, we assume that there is not any noise added to the signal values, hence we write the observation as:

$$\boldsymbol{y}_t = \boldsymbol{x}_t. \tag{3.10}$$

The state transition matrix $\boldsymbol{\Phi}_{(t+i,t)}$ between time t and $t+i$ is defined as:

$$\boldsymbol{\Phi}_{(t+i,t)} = \prod_{\tau=0}^{i} \boldsymbol{\Phi}_{t+i-\tau}. \tag{3.11}$$

In order to simplify the notations where t is included, without loss of generality, we shift the time to make it $t = 0$ and omit it from the notation.
If the first observation is made at time t, all observations $\boldsymbol{y}_i$ for $i \in \{1, 2, 3, .., T\}$ can be stacked in a single vector:

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}_1 \\ \boldsymbol{\Phi}_{(2,1)} \\ \vdots \\ \boldsymbol{\Phi}_{(T,1)} \end{bmatrix} \boldsymbol{x}_0 + \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \dots \\ \boldsymbol{\Phi}_2 & \boldsymbol{I} & \boldsymbol{0} & \dots \\ \vdots & \vdots & \ddots & \\ \boldsymbol{\Phi}_{(T,2)} & \boldsymbol{\Phi}_{(T,3)} & \dots & \boldsymbol{I} \end{bmatrix} \left(\begin{bmatrix} \boldsymbol{o}_1 \\ \boldsymbol{o}_2 \\ \vdots \\ \boldsymbol{o}_T \end{bmatrix} + \begin{bmatrix} \boldsymbol{n}_1 \\ \boldsymbol{n}_2 \\ \vdots \\ \boldsymbol{n}_T \end{bmatrix} \right) \tag{3.12}$$

$$\boldsymbol{y} = \boldsymbol{\Phi} \boldsymbol{x}_0 + \boldsymbol{\Omega}(\boldsymbol{o} + \boldsymbol{n}) \tag{3.13}$$

The absence of subscripts in vectors indicate a stacking of vectors starting from timeframe 1 until the last observed timeframe T. The same notation will be used throughout the thesis for other vectors. The matrix $\boldsymbol{\Phi}$ indicates a stacking of the power of state transition matrices from $\boldsymbol{\Phi}_1$ to $\boldsymbol{\Phi}_{(T,1)}$. And the matrix $\boldsymbol{\Omega}$ refers to the lower triangular matrix in equation (3.12).

3.3. Anomalous Action Model

We introduce the following three different models for the manner in which malicious agents inject outlying values into the network. We will show that for all three models, the hypothesis test for the detection problem will be different. As a result, due to the different nature of the problems arising from the respective performance metrics, the derived optimization problems will also differ.

3.3.1. Time-invariant attacks

In the past, it has been considered how to approach the problem of a time-invariant attack on a network [18]. For example, constant data injection is dealt with in [19]. In [14], the time invariant attack is referred to as static attack. In these works, a motivation is not provided for the use of time-invariant attacks. Here, we will motivate this kind of network attack by listing two advantages that a time-invariant attack provides for the malicious agents.

1. Constant and time-invariant injection of false data does not require an observation of the network by the malicious agents. Outlying values that are dependent on the signal values require a constant observation of the network, thus resulting in a higher usage of resources by the malicious agents.
2. Apart from saving on resources, a constant and time-invariant injection results in a constant power ratio of injected-to-noise data. A non-increasing power ratio will increase the chances of being undetected. This will be true for the framework for detection that is provided in this thesis due to the fact that we do not have to deal with the power in the signal. The reason for this will be clear in the next chapter.

The structure of the outlying values for a time-invariant constant attack is given by the following equations:

$$\begin{aligned} \boldsymbol{o}_i &= c\mathbf{1}_{\mathcal{A}_i} : \forall \quad i \in \{0, ...T\} \\ \mathbf{1}_{\mathcal{A}_i} &= \mathbf{1}_{\mathcal{A}} : \forall \quad i \in \{0, ...T\}, \end{aligned} \tag{3.14}$$

where $\mathbf{1}_{\mathcal{A}_i}$ is defined as the support vector at time i.
In the continuation of the thesis, the time invariant attack will be referred to as *Model 1* or M.1,

3.3.2. Time-variant attacks

Time varying attacks are defined as attacks where the outliers vary over time, i.e.,

$$\boldsymbol{o}_i \neq \boldsymbol{o}_j, \quad : \forall \quad i \neq j. \tag{3.15}$$

Time varying attacks are effective when the malicious agents are interested in exerting a greater influence on the signal values. In [5], for example, an attacking schedule that will result in the steering of a consensus based distributed network towards a desired final state is given as

$$\boldsymbol{x}_{i+1} = g(i)\boldsymbol{\Phi}_{i+1}\boldsymbol{x}_i + \boldsymbol{\alpha}_0(1 - g(i)) : \forall \quad i \in \{0, ...T\}, \tag{3.16}$$

where $g(i) \in [0, 1]$ is a variable that is decreasing over time and $\boldsymbol{\alpha}_0$ is the desired state of the malicious agents. The outliers are given then as:

$$\boldsymbol{o}_{i+1} = (g(i) - 1)\boldsymbol{\Phi}_{i+1}\boldsymbol{x}_i + \boldsymbol{\alpha}_0(1 - g(i)) : \forall \quad i \in \{0, ...T\}. \tag{3.17}$$

In (3.17), we see how the outliers are produced depending on the previous signal values. A fraction of the signal value is subtracted and at the same time the same fraction of the desired final state is added. After every time frame is the fraction $1 - g(i)$ increasing.
A particular version of this attacking schedule will be used for evaluation purposes in this thesis. In this particular version we want the nodes in the network to converge to the average of the initial values. In case the goal of the agents is to steer the state to the global average, it is not mandatory to alter the nodal values of the complete network, i.e., all the nodes. Instead, steering only the nodes that are selected by the agents to the desired average will also force the rest of the network to this desired global average. Hence, we can denote the outliers and node values then respectively as:

$$\boldsymbol{o}_{i+1} = \big((g(i) - 1)\boldsymbol{\Phi}_{i+1}\boldsymbol{x}_i + \boldsymbol{\alpha}_0(1 - g(i))\big) \odot \mathbf{1}_{\mathcal{A}_i} : \forall \quad i \in \{0, ...T\}, \tag{3.18}$$

$$\boldsymbol{x}_{i+1} = \boldsymbol{\Phi}_{i+1}\boldsymbol{x}_i + \big((g(i) - 1)\boldsymbol{\Phi}_{i+1}\boldsymbol{x}_i + \boldsymbol{\alpha}_0(1 - g(i))\big) \odot \mathbf{1}_{\mathcal{A}_i} : \forall \quad i \in \{0, ...T\}. \tag{3.19}$$

The notation $\odot$ refers to a pointwise multiplication of vectors. Also, $\boldsymbol{\alpha}_0$ is now an all-ones vector with a scalar value which we will call α_0.
In section 3.4, an example is given for the visualisation of the nodal values with this attacking schedule for both attacking modes. The two attacking modes will also be explained in section 3.4.

Fixed anomalous nodes

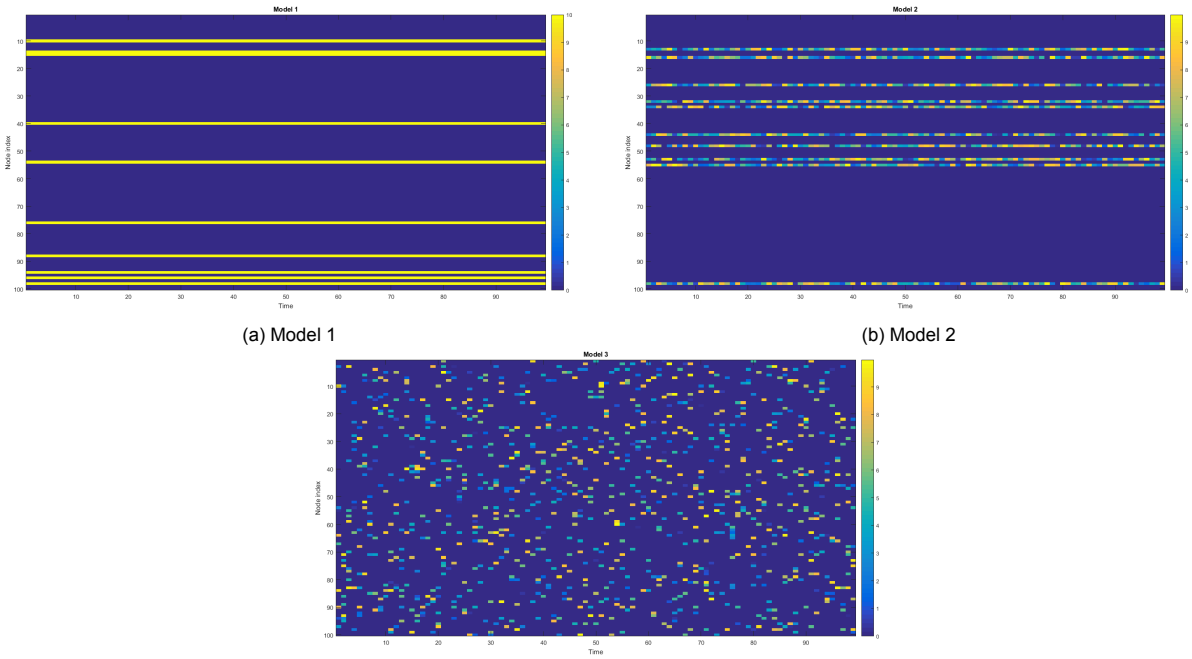
Time varying attacks can be seperated in two different kinds. If the support of the outliers is fixed over time, only the values are then varying. This attack will be called *Model 2* or M.2 in the continuation of the thesis. $\mathbf{1}_{\mathcal{A}}$ is then said to be constant for all observed timeframes i, i.e.,

$$\mathbf{1}_{\mathcal{A}_i} = \mathbf{1}_{\mathcal{A}} : \forall \quad i \in \{0, ...T\}. \tag{3.20}$$

Time-varying anomalous nodes
The final attack mode is the time-varying attack where we assume that the support can be, although not necessarily, varying over time, i.e.,

$$\mathbf{1}_{\mathcal{A}_i} \neq \mathbf{1}_{\mathcal{A}_j} \quad \text{for some} \quad i \neq j. \tag{3.21}$$

This attack mode will be called *Model 3* or $\mathrm{M}.3$. In case that the structure of the network is also time-varying, this kind of attack becomes necessary. In figure 4.3, the different models are illustrated with a sparsity of 10%, i.e., 10% of the nodes are assumed to be compromised for time i for $i \in \{1, 2, \ldots 100\}$.



(a) Model 1

(b) Model 2

(c) Model 3

Figure 3.2: The images show the outlying values $\boldsymbol{o}_i$ for $i \in \{1, 2, \ldots 100\}$ and 100 nodes, for all three models. Every column represents a different timeframe and every row is a different node.

3.4. Anomalous Agents Network Constraints

Apart from the models that are proposed for the variety in time of the outliers $\boldsymbol{o}$ in the previous chapter, there can also be constraints ascribed to the support set $\mathcal{A}$, for example, due to the limitations in resources of the agents. These constraints can also be dependent on the topology of the network. In case that the malicious set of agents are limited in their resources, the compromised nodes will be also limited in their amount. Hence our first constraint is the limited cardinality of $\mathcal{A}$, i.e.,

$$K(\mathbf{1}_{\mathcal{A}_i}) = \left\|\mathbf{1}_{\mathcal{A}_i}\right\|_0 \leq \beta : \forall \quad i \in \{0, \ldots . T\}, \tag{3.22}$$

for some $\beta \in \mathbb{R}^+$.

For an increased effectivity of network influence for the agents, it is preferred to compromise high degree nodes. However, highly connected nodes are fundamental for the operation of the network and therefore they, most of the time, are highly secured. Therefore, this translates to a large expenditure of resources for a malicious agent if it tries to compromise such nodes. Thus, the malicious agents will limit their direct reach, i.e., to a one-hop neighbourhood, leading to the following (total degree) constraint

$$D(\mathbf{1}_{\mathcal{A}_i}) = \boldsymbol{d}_i^T \mathbf{1}_{\mathcal{A}_i} \leq \alpha : \forall \quad i \in \{0, \ldots . T\}. \tag{3.23}$$

Thus, $D(\mathbf{1}_{\mathcal{A}_i})$ is defined as the total degree or density of all the selected nodes. A degree of a node is defined as the total number of direct neighbours of that node. The vector $\boldsymbol{d}_i$ is the vector which contains the (possibly time-dependent) degrees of the nodes of graph $\mathcal{G}$.

Due to these two limitations, the malicious agents must provide a graph-related structure to the set of compromised nodes in order to optimize the influence on the network. The relation to the graph is based on the connectivity measures between the nodes that are and are not compromised. An increasing amount of edges between these two subsets of nodes results in an increasing influence of the malicious agents on the network. A way to measure this connectivity is by the cut function. In classical literature, the cut function of a node set is defined as the (un)weighted amount of outgoing edges of that nodeset, see section 2.1. In this thesis, the cut is defined as the unweighted amount of edges between the compromised nodeset and the 'clean' nodeset. The cut function can then be formulated as:

$$C(\mathbf{1}_{\mathcal{A}_i}) = \mathbf{1}_{\mathcal{A}_i}^T \boldsymbol{L} \mathbf{1}_{\mathcal{A}_i} : \forall \quad i \in \{0, ...T\} \tag{3.24}$$

Alternatively, we can also minimize the cardinality of the dependent edge set of $\mathcal{A}$. The dependent edge set of $\mathcal{A}$, is defined as $\mathcal{F}$, where

$$(i,j) \in \mathcal{F} \quad \text{iff} \quad i,j \in \mathcal{A} : \forall \quad (i,j) \in \mathcal{E}. \tag{3.25}$$

An example of the edges that belong to $\mathcal{F}$ is given in figure 3.3.

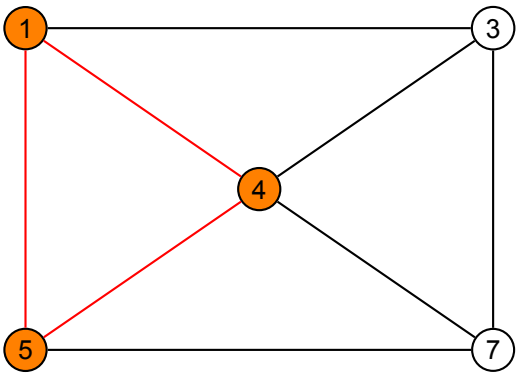


Figure 3.3: For the identical graph and set of malicious nodes as in figure 2.1, now, the set of edges that belong to the dependent edge set are given in red

Every element in the dependent edge set can be seen as an unnecessary edge that does not contribute to the influence on the network by the malicious agents. The influence on the 'clean' nodes are limited to indirect altering. Hence, in order to maximize the influence on these nodes, one can chose $\mathcal{A}$ such that $|\mathcal{F}|$ is minimized. The relationship between the cut, $\mathcal{F}$ and density can be explained with the following equation:

$$D(\mathbf{1}_{\mathcal{A}_i}) = C(\mathbf{1}_{\mathcal{A}_i}) + 2|\mathcal{F}_i| : \forall \quad i \in \{0, ...T\}. \tag{3.26}$$

where $\mathcal{F}_i$ is the dependent edge set of $\mathcal{A}_i$.
As it is stated above, we can describe how the maximization of the cut will increase the influence in the network by minimizing the difference between the degree and the cut, i.e.,

$$|\mathcal{F}_i| = \frac{D(\mathbf{1}_{\mathcal{A}_i}) - C(\mathbf{1}_{\mathcal{A}_i})}{2} : \forall \quad i \in \{0, ...T\}. \tag{3.27}$$

Hence, our third constraint is directly linked to to the cardinality of the dependent edge set:

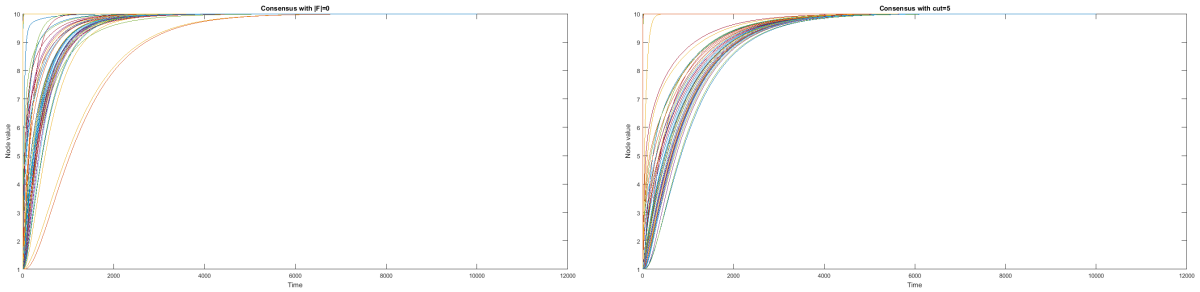
$$B(\mathbf{1}_{\mathcal{A}_i}) = |\mathcal{F}_i| \leq \rho : \forall \quad i \in \{0, ...T\}, \tag{3.28}$$

for some $\rho \in \mathbb{R}^+$.
In case the malicious agents are interested in only a part of the network, they will focus their resources on compromising nodes that belong to the part of the network in question. Instead of imposing a constraint on the dependent edge set, there will be a constraint on the cut of the compromised subset of nodes. A constraint on the cut will limit the influence of the agents on the rest of the network and

optimize the influence on the part of the network in question. Hence, the constraint given in equation (3.28) may be replaced by a new constraint depending on the intention of the agents:

$$C(\mathbf{1}_{\mathcal{A}_i}) \leq \rho : \forall \quad i \in \{0, ...T\} \tag{3.29}$$

In order to understand the concept of influence on the network, there are two plots given in figure 3.4 that show how the signal values in a network are affected by the attack schemes given in (3.17). For each plot a different compromised set $\mathcal{A}$ is given, depending on the intention of the malicious agents. Hence, $\mathcal{A}$ is dependent on the constraints given in (3.28) and (3.29), respectively for each plot.



(a) Consensus with $|\mathcal{F}| = 0$ constraint

(b) Consensus with cut=5 constraint

Figure 3.4: In the figures, we have plotted the signal values for a varying time. In figure 3.4a, $\mathcal{A}$ is chosen such that $|\mathcal{F}| = 0$ and in figure 3.4b $\mathcal{A}$ is chosen such that $C(\mathbf{1}_{\mathcal{A}}) = 5$ at all times. For both plots, identical graphs are used and all node values have an initial value of 1. The desired value is assumed to be 10 for all nodes

It can be noticed how most of the nodes in figure 3.4a converge in a quicker pace to the desired state compared with the node values in 3.4b, hence the influence is optimized for the complete network. In figure 3.4b, the compromised nodes converge quicker to the desired state in comparison to the rest of the nodes, which means that the influence is optimized for the selected subset of nodes. The attack that is visualized in figure 3.4a will be called attack a and in figure 3.4b will be called attack b.

In the next chapter, we will provide the hypothesis testing problems together with the performance metrics with the help of the provided data model of this chapter. Also, the metrics for the Lasso-based method will be treated.

4

Anomalous Network Agent Detection

This chapter discusses the relation between the classical outlier detection and the provided problem in chapter 3. Secondly, the limitations of the classical Lasso-based approach are discussed. After this, we mathematically derive the performance metrics that are based on the maximum likelihood estimators and compare this framework with the classical matched filter. And also we derive the optimization problem for the Lasso-based method. Finally, an alternative performance metric is discussed in the last part of this chapter.

4.1. Anomalies as Outlying Values

In anomaly detection problems the task is to detect whether observations diverge from a pattern and decide if the observations are normal or not. Outlier detection is a form of anomaly detection where outlier data ought to be separated from the normal signal data. In the previous section, we have defined the outlier $\boldsymbol{o}_t$ as an addition to the signal that is to be observed at time t:

$$\boldsymbol{y}_t = \boldsymbol{s}_t + \boldsymbol{o}_t + \boldsymbol{n}_t \tag{4.1}$$

By using the formulation of our data model in (3.13), it is possible to write the hypothesis for our problem as an outlier detection problem

$$\begin{aligned} \mathcal{H}_0 &: \quad \boldsymbol{y} = \boldsymbol{\Phi x}_0 + \boldsymbol{\Omega n} \\ \mathcal{H}_1 &: \quad \boldsymbol{y} = \boldsymbol{\Phi x}_0 + \boldsymbol{\Omega}(\boldsymbol{o} + \boldsymbol{n}) \end{aligned} \tag{4.2}$$

By making use of the fact that $\boldsymbol{\Phi}$ is known, see section 3.2, it is possible to reformulate the hypothesis in (4.2). First we introduce a new variable $\boldsymbol{z}$, that is defined with the help of the following equation:

$$\boldsymbol{z} = \boldsymbol{\Omega}^{-1}(\boldsymbol{y} - \boldsymbol{\Phi x}_0) \tag{4.3}$$

Since $\boldsymbol{\Phi}$ is known, we can subtract $\boldsymbol{\Phi x}_0$ from our observed data $\boldsymbol{y}$, given that the observation starts at time $t = 0$. Secondly, the matrix $\boldsymbol{\Omega}$ is always invertible regardless of $\boldsymbol{\Phi}$. This is due to the fact that the set of eigenvalues of a lower triangular matrix are equal to the set of the diagonal values [20]. Since the diagonal values of $\boldsymbol{\Omega}$ are always equal to one, it will be the case that $\boldsymbol{\Omega}$ is always invertible. It can be said now that $\boldsymbol{z}$ can be calculated by the auditor of the network, provided that there are at least two observed timeframes. The equation in (3.13) is then simplified as:

$$\boldsymbol{z} = \boldsymbol{o} + \boldsymbol{n} \tag{4.4}$$

Giving rise to a new detection problem:

$$\begin{aligned} \mathcal{H}_0 &: \quad \boldsymbol{z} = \boldsymbol{n} \\ \mathcal{H}_1 &: \quad \boldsymbol{z} = \boldsymbol{o} + \boldsymbol{n} \end{aligned} \tag{4.5}$$

The problem in (4.5) can be seen as a signal detection problem where the signal ought to be unknown but subjected to a certain structure explained in sections 3.3 and 3.4. The support of $\boldsymbol{o}$ is then also subjected to constraints that determine a subset within a larger powerset with cardinality $2^{|\mathcal{V}|}$ where the support set can be chosen from, see section 3.4. The powerset is defined as the set that contains all the subsets of $\mathcal{V}$.

4.2. Pitfalls of Vanilla Sparse Regression

Before a detection test can be carried out, an estimation must be done for the unknown parameters. Omitting the structural contraints for the malicious nodes, the estimation that can be carried out is the maximum likelihood estimator [21].

$$\boldsymbol{o}^* = \underset{\boldsymbol{o}}{\operatorname{argmax}}\, p(\boldsymbol{z}|\boldsymbol{o}) \tag{4.6}$$

The maximum likelihood estimation (MLE) of a signal with additive white Gaussian noise is equal to the least squares error estimation [21]:

$$\boldsymbol{o}^* = \underset{\boldsymbol{o}}{\operatorname{argmin}} \|\boldsymbol{z} - \boldsymbol{o}\|_2^2 \tag{4.7}$$

Which will give a trivial solution as $\boldsymbol{o}^* = \boldsymbol{o}$ if no constraints are assumed. From chapter 3, we know that not all of the nodes are compromised but only a few. If we model the outliers as a Laplacian prior, we can then observe that this leads to an l1-norm problem [22]. This is alternatively interpreted as a regularized linear regression problem. The scaling parameter λ of the Laplacian distribution is then used for the tuning of the variable sparsity:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{y} - \mathbf{B}\boldsymbol{x}\|^2 + \lambda \|\boldsymbol{x}\|_1 \tag{4.8}$$

In (4.8), $\boldsymbol{y}$ is assumed to follow a linear observation model with additive Gaussian noise, i.e., $\boldsymbol{y} = \mathbf{B}\boldsymbol{x} + \boldsymbol{n}$. Sparse regression is a collection of methods that is used for optimization problems, where the sparsity of the variable is exploited. In sparse regression methods a new variable or a set of variables is introduced that can be tuned in order to increase or decrease the sparsity of a variable. A well-known example for this is the lasso method [23], which is explained above. We can also penalize the l2-norm of the variable, which is known as the ridge regression method [24]. The l1- or l2-norm can be seen as a relaxation of the cardinality $\|\boldsymbol{o}\|_0$. If we combine the l1- and l2-norm as penalty terms, a new sparse regression method can be defined: elastic net regularization [24].
In the original paper for the lasso method [23], results show how the lasso-based least squares method, c.f. (4.8), outperforms other methods, like the non-negative garrotte method or the classical least squares method, in terms of the mean square error, i.e., the average squared difference between the estimated values and the actual value of the variable $\boldsymbol{x}$. These results are calculated for the situation where $\mathbf{B}$ is a matrix with dimensions $1 \times N$, i.e., a row vector that is correlated with $\boldsymbol{x}$ over different noise realisations and for different values of $\mathbf{B}$.
In case that $\mathbf{B}$ has orthogonal columns, i.e., $\mathbf{B}^T\mathbf{B} = \boldsymbol{I}$, we can make use of the special form for the closed form solution of $\boldsymbol{x}$ in (4.8), see [23]. The closed form solution for the Lasso method for our case, i.e.,

$$\boldsymbol{o}_i^* = \underset{\boldsymbol{o}_i}{\operatorname{argmin}} \|\boldsymbol{z}_i - \boldsymbol{o}_i\|_2^2 + \lambda_i \|\boldsymbol{o}_i\|_1 : \forall \quad i \in \{1, \ldots T\}, \tag{4.9}$$

is then denoted as:

$$\boldsymbol{o}_i^* = sgn(\boldsymbol{z}_i)(|\boldsymbol{z}_i| - \boldsymbol{\lambda}_i)_+ : \forall \quad i \in \{1, \ldots T\}, \tag{4.10}$$

where $sgn(\boldsymbol{z}_i)$ returns the sign vector of $\boldsymbol{z}_i$ and the $(.)_+$ operator turns all negative values to zero in the vector of $(|\boldsymbol{z}_i| - \boldsymbol{\lambda}_i)$. Also the newly defined vector $\boldsymbol{\lambda}_i$ is the multiplication of an all-ones vector with λ_i. Hence, the method returns a linearly regressed $\boldsymbol{z}_i$ as the solution for $\boldsymbol{o}_i$. This is done for all timestamps $i \in \{1, \ldots T\}$ separately, i.e., every timestamp has a different optimal value for λ, i.e., λ_i .

However, apart from the sparsity there are other constraints that must be included in the estimation. The constraints discussed in section 3.4 are related to the support set $\mathcal{A}$ of the outliers. Relaxing the constraints with a dependency on $\boldsymbol{o}_i$ could be possible, similar as for the cardinality:

$$\left\|\mathbf{1}_{\mathcal{A}i}\right\|_0 = \|\boldsymbol{o}_i\|_0 \longrightarrow \|\boldsymbol{o}_i\|_1 : \forall \quad i \in \{1, \ldots T\} \tag{4.11}$$

However, multiple relaxed constraints where the support vector $\mathbf{1}_{\mathcal{A}}$ is substituted by the outliers $\boldsymbol{o}$ will result in a very complicated optimization problem. Multiple regularization parameters as λ are then required. For example if we would have to include the cut $C(\mathbf{1}_{\mathcal{A}})$ and the density $D(\mathbf{1}_{\mathcal{A}})$ as constraints in the estimation, we will need to have the following optimization problem:

$$\boldsymbol{o}_i^* = \underset{\boldsymbol{o}_i}{\operatorname{argmin}} \|\boldsymbol{z}_i - \boldsymbol{o}_i\|_2^2 + \lambda_1 \|\boldsymbol{o}_i\|_0 + \lambda_2 C_1(\boldsymbol{o}_i) + \lambda_3 D_1(\boldsymbol{o}_i) : \forall \quad i \in \{1, \ldots T\} \tag{4.12}$$

where $C_1(\boldsymbol{o}_i)$ and $D_1(\boldsymbol{o}_i)$ are relaxations of the cut and density, respectively. Regularizing 3 different parameters gives a lot of uncertainties about how the penalty terms will act and if there can be found a solution that meets the constraints. Apart from that, even if it is possible to find a relaxation in terms of $\boldsymbol{o}_i$, it is not guaranteed that the derived relaxations of the constraints in section 3.4 are a good relaxation. Finding a proper convex relaxation of the constraints in section 3.4 is difficult in terms of $\boldsymbol{o}_i$. For example, a relaxation of the density constraint $D(\mathbf{1}_{\mathcal{A}_i})$ can be achieved with the help of the weighted node degrees, i.e.,

$$D_1(\boldsymbol{o}_i) = \frac{\boldsymbol{d}_i^T|\boldsymbol{o}_i|}{\|\boldsymbol{o}_i\|_1} \leq \alpha : \forall \quad i \in \{1, ...T\}. \tag{4.13}$$

This formulation can be manipulated into a convex relaxation only when all the elements in $\boldsymbol{o}_i$ are positive, i.e., $\boldsymbol{d}_i^T|\boldsymbol{o}_i| = \boldsymbol{d}_i^T\boldsymbol{o}_i$. The relaxation then looks as

$$\boldsymbol{d}_i^T|\boldsymbol{o}_i| - \alpha\mathbf{1}^T\boldsymbol{d}_i \leq 0 : \forall \quad i \in \{1, ...T\}. \tag{4.14}$$

Even when all the outlying values are positive, an accurate relaxation is not guaranteed. If the non-zero values in $\boldsymbol{o}_i$ differ from each other greatly, the value of the relaxation is also going to deviate greatly from the true value. Hence, the solutions derived for $\boldsymbol{o}_i$ will be inaccurate due to the multiple weak relaxations like the above given example. Instead, we will look at the given models in section 3.3 and derive a maximum likelihood estimation based on subset selection. And for evaluation purposes of the lasso-based method, we will only take the cardinality constraint into consideration. This way, we can use the closed form solution for the lasso-based method in (4.10) as a benchmark.

4.3. Maximum Likelihood-based Metrics

The subset selection method evaluates all the possible subsets for the support of $\boldsymbol{o}_t$. Instead of a linear regression until a desired cardinality is reached, the subset will search exhaustively for the subset with the optimal value for the problem. Consequently, the variable elements $[\boldsymbol{o}_t]_i$ are then forced to zero if $i \notin \mathcal{A}$. As stated earlier in section 3.1, the task is to find the support for $\boldsymbol{o}$ that gives the highest detection probability for a fixed false alarm probability. Instead of a direct estimation for $\boldsymbol{o}$, an expression for the estimation will be found for $\boldsymbol{o}$ that is dependent on $\mathbf{1}_{\tilde{\mathcal{A}}}$. The vector $\mathbf{1}_{\tilde{\mathcal{A}}}$ is defined as the stacking of the support vectors for timeframes $i \in \{1, ...T\}$. Hence, $\tilde{\mathcal{A}}$ is defined as the collection of the support sets for timeframes $i \in \{1, ...T\}$, i.e.,

$$\tilde{\mathcal{A}} = \{(\mathcal{A}_1, ..., \mathcal{A}_T)\}. \tag{4.15}$$

Consequently, we can derive a likelihood ratio that is also dependent on $\mathbf{1}_{\tilde{\mathcal{A}}}$:

$$T(\boldsymbol{z}, \mathbf{1}_{\tilde{\mathcal{A}}}) = \frac{P(\boldsymbol{z}; \mathcal{H}_1, \mathbf{1}_{\tilde{\mathcal{A}}})}{P(\boldsymbol{z}; \mathcal{H}_0)} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \gamma, \tag{4.16}$$

since the likelihood function for $\boldsymbol{z}$ under $\mathcal{H}_1$ is dependent on $\mathbf{1}_{\tilde{\mathcal{A}}}$. For a signal detection problem denoted as in (4.5), where the noise $\boldsymbol{n}$ is assumed to be white Gaussian, the standard expression for the log-likelihood ratio is derived as [25]:

$$T(\boldsymbol{z}, \mathbf{1}_{\tilde{\mathcal{A}}}) = \frac{1}{\sigma^2}\boldsymbol{o}^T\boldsymbol{z} - \frac{1}{2\sigma^2}\boldsymbol{o}^T\boldsymbol{o} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \gamma \tag{4.17}$$

Thus, the likelihood functions for $T(\boldsymbol{z}, \mathbf{1}_{\tilde{\mathcal{A}}})$ will also be dependent on $\mathbf{1}_{\tilde{\mathcal{A}}}$:

$$\begin{aligned} \mathcal{H}_0 : \quad & P(T(\boldsymbol{z}, \mathbf{1}_{\tilde{\mathcal{A}}}); \mathcal{H}_0, \mathbf{1}_{\tilde{\mathcal{A}}}) \\ \mathcal{H}_1 : \quad & P(T(\boldsymbol{z}, \mathbf{1}_{\tilde{\mathcal{A}}}); \mathcal{H}_1, \mathbf{1}_{\tilde{\mathcal{A}}}) \end{aligned} \tag{4.18}$$

Mathematically, the expression for the detection probability is then given as:

$$P_d = P(\mathcal{H}_1; \mathcal{H}_1) = P(T(\boldsymbol{z}, \mathbf{1}_{\tilde{\mathcal{A}}}) > \gamma'; \mathcal{H}_1, \mathbf{1}_{\tilde{\mathcal{A}}}) \tag{4.19}$$

And for the false alarm probability, the expression is denoted as:

$$P_{fa} = P(\mathcal{H}_1; \mathcal{H}_0) = P(T(\boldsymbol{z}, \mathbf{1}_{\tilde{\mathcal{A}}}) < \gamma'; \mathcal{H}_0, \mathbf{1}_{\tilde{\mathcal{A}}}) \tag{4.20}$$

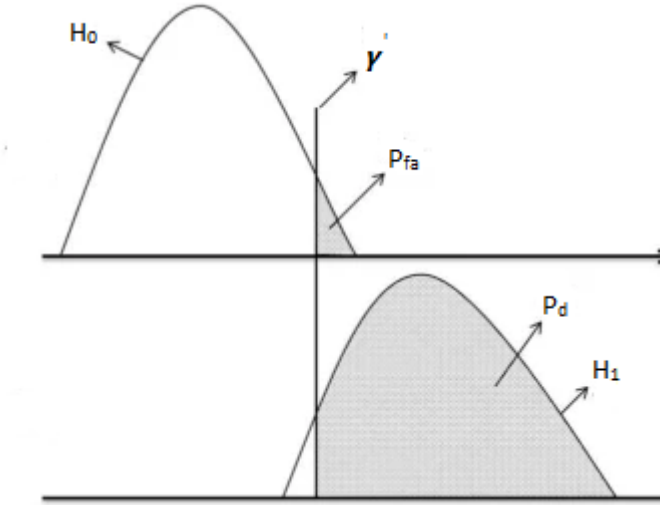


Figure 4.1: For a calculated $\mathbf{1}_{\tilde{\mathcal{A}}}$ a certain distribution is derived for the test statistic under both hypotheses. The detection and false alarm probabilities are visualized for a given threshold value

The Q-function, which is defined as the tail distribution function of $T(\boldsymbol{z}, \mathbf{1}_{\tilde{\mathcal{A}}})$ under $\mathcal{H}_0$ will yield the false alarm probability

$$Q_{T(\boldsymbol{z},\mathbf{1}_{\tilde{\mathcal{A}}});\mathcal{H}_1}(\gamma') = P_{fa} \tag{4.21}$$

The threshold value can then be calculated with the help of the inverse Q-function,

$$Q^{-1}_{T(\boldsymbol{z},\mathbf{1}_{\tilde{\mathcal{A}}});\mathcal{H}_1}(P_{fa}) = \gamma' \tag{4.22}$$

By the substitution of γ' with the expression given in (4.22), we can then, for a given P_{fa}, find the support vector that gives the optimal detection probability.

$$P_d = P(T(\boldsymbol{z}, \mathbf{1}_{\tilde{\mathcal{A}}}) > Q^{-1}_{T(\boldsymbol{z},\mathbf{1}_{\tilde{\mathcal{A}}});\mathcal{H}_1}(P_{fa}); \mathcal{H}_1, \mathbf{1}_{\tilde{\mathcal{A}}}) \tag{4.23}$$

In order to simplify notation, we will not denote the dependency of the threshold $T(\boldsymbol{z})$ and likelihood functions $P(T(\boldsymbol{z}))$ on $\mathbf{1}_{\tilde{\mathcal{A}}}$, hereafter.
If we have to visualize what the equation in (4.23) will mean for the distributions of $T(\boldsymbol{z})$ under both hypotheses we can look at figure 4.1. By analyzing the expression that we have for P_d, it is now possible to evaluate the relationship between P_d and $\mathbf{1}_{\tilde{\mathcal{A}}}$. After this, we can derive a cost function f_{P_d} that must be optimized in order to yield a support vector $\mathbf{1}_{\tilde{\mathcal{A}}}$ from which the maximum P_d can be calculated. Given the properties for the support set in section 3.4, an optimization problem can be formulated now for the models M.1a, M.2a and M.3a as:

$$\begin{aligned} \max_{\mathbf{1}_{\tilde{\mathcal{A}}}} \quad & f_{P_d}(\mathbf{1}_{\tilde{\mathcal{A}}}) \\ \text{subject to} \quad & B(\mathbf{1}_{\mathcal{A}_i}) \leq \rho : \forall \quad i \in \{1, \ldots T\}, \\ & K(\mathbf{1}_{\mathcal{A}_i}) \leq \beta : \forall \quad i \in \{1, \ldots T\}, \\ & D(\mathbf{1}_{\mathcal{A}_i}) \leq \alpha : \forall \quad i \in \{1, \ldots T\} \end{aligned} \tag{4.24}$$

And for the models M.1b, M.2b and M.3b the optimization problem can be formulated as:

$$\begin{aligned} \max_{\mathbf{1}_{\tilde{\mathcal{A}}}} \quad & f_{P_d}(\mathbf{1}_{\tilde{\mathcal{A}}}) \\ \text{subject to} \quad & C(\mathbf{1}_{\mathcal{A}_i}) \leq \rho : \forall \quad i \in \{1, \ldots T\}, \\ & K(\mathbf{1}_{\mathcal{A}_i}) \leq \beta : \forall \quad i \in \{1, \ldots T\}, \\ & D(\mathbf{1}_{\mathcal{A}_i}) \leq \alpha : \forall \quad i \in \{1, \ldots T\} \end{aligned} \tag{4.25}$$

For M.1 and M.2 we include the additional constraint:

$$\mathbf{1}_{\mathcal{A}_i} = \mathbf{1}_{\mathcal{A}} : \forall \quad i \in \{1, \ldots T\} \tag{4.26}$$

In order to derive a unique optimization problem for all three different models, we will derive a maximum likelihood estimation and cost function for time-invariant and time-varying models seperately.

4.3.1. Time-invariant attacks

With the given framework above, we can derive a threshold function that is specific for the time invariant attack. To do so, we rewrite the hypothesis for the time-invariant attack M.1 as:

$$\begin{aligned} \mathcal{H}_0: &\quad \boldsymbol{z} = \boldsymbol{n} \\ \mathcal{H}_1: &\quad \boldsymbol{z} = c\mathbf{1}_{\tilde{\mathcal{A}}} + \boldsymbol{n} \end{aligned} \tag{4.27}$$

In case we assume that $\mathbf{1}_{\tilde{\mathcal{A}}}$ is known, we can set up an optimization problem for the MLE:

$$\min_{c} \left\| \boldsymbol{z} - c\mathbf{1}_{\tilde{\mathcal{A}}} \right\|_2^2 \tag{4.28}$$

From which we can derive an expression for c as:

$$c = \frac{\mathbf{1}_{\tilde{\mathcal{A}}}{}^T \boldsymbol{z}}{|\mathcal{A}|T} \tag{4.29}$$

Where $\mathcal{A}$ is the constant support set for all timeframes $i \in \{1, \ldots T\}$.
Hence, c is the average of all the selected nodes over all the T timeframes. By substituting the outliers $\boldsymbol{o}$ with $c\mathbf{1}_{\tilde{\mathcal{A}}}$ in (4.17), the log-likelihood ratio test can then be formulated as and simplified to:

$$T(\boldsymbol{z}) = \frac{|\boldsymbol{z}^T \mathbf{1}_{\tilde{\mathcal{A}}}|}{\sqrt{T|\mathcal{A}|}} \underset{H_0}{\overset{H_1}{\gtrless}} \gamma' \tag{4.30}$$

In appendix A, the necessary steps are given for the derivation of c and $T(\boldsymbol{z})$. The distribution of the test statistic under both hypotheses is denoted as

$$\begin{aligned} \mathcal{H}_0: &\quad T(\boldsymbol{z}) \sim \begin{cases} 2\mathcal{N}(0, \sigma^2), & \text{if } T(\boldsymbol{z}) \geq 0, \\ 0, & \text{if } T(\boldsymbol{z}) < 0, \end{cases} \\ \mathcal{H}_1: &\quad T(\boldsymbol{z}) \sim \begin{cases} \mathcal{N}(A, \sigma^2) + \mathcal{N}(-A, \sigma^2), & \text{if } T(\boldsymbol{z}) \geq 0, \\ 0, & \text{if } T(\boldsymbol{z}) < 0, \end{cases} \end{aligned} \tag{4.31}$$

where A, which we will call the amplitude of the total outlier energy, is substituted for

$$A = \frac{\mathbf{1}_{\tilde{\mathcal{A}}}^T \boldsymbol{z}}{\sqrt{T|\mathcal{A}|}}. \tag{4.32}$$

The Q-function of a Gaussian distribution that is folded around 0, which is the case for the distributions in (4.31), is equal to the summation of two Gaussian Q-functions that are shifted left and right with the mean of that Gaussian distribution [26]. Hence, for M.1, P_d can be formulated as the summation of two Q-functions that are shifted positively and negatively with the term A under $\mathcal{H}_1$:

$$P_d = Q\left(\frac{\gamma' - A}{\sqrt{\sigma^2}}\right) + Q\left(\frac{\gamma' + A}{\sqrt{\sigma^2}}\right). \tag{4.33}$$

The false alarm probability is, like the detection probability, expressed using the Gaussian Q-function

$$P_{fa} = 2Q\left(\frac{\gamma'}{\sqrt{\sigma^2}}\right) \tag{4.34}$$

By substitution, the following dependency is provided between P_{fa} and P_d

$$P_d = Q\left(Q^{-1}(P_{fa}/2) - \frac{A}{\sqrt{\sigma^2}}\right) + Q\left(Q^{-1}(P_{fa}/2) + \frac{A}{\sqrt{\sigma^2}}\right) \tag{4.35}$$

The term that is dependent on $\mathcal{A}$ and that must be analyzed is the amplitude A. The function for P_d is symmetric around zero for a varying amplitude A, furthermore the Gaussian Q-function $Q(x)$ is a

monotonic decreasing function for x. Thus, if we would like to increase the detection probability, the magnitude of the amplitude must be increased. Hence our cost function for M.1 is:

$$f_{P_{d1}}(\mathbf{1}_{\tilde{\mathcal{A}}}) = \frac{|\boldsymbol{z}^T \mathbf{1}_{\tilde{\mathcal{A}}}|}{\sqrt{|\mathcal{A}|}} \tag{4.36}$$

It is also possible to use the performance metric that is derived for the matched filter, see section 2.2. We can make use of this metric by not assuming any model for $\boldsymbol{o}$ during the derivation of the P_d. This will mean that the likelihood functions are not derived as an expression of $\mathcal{A}$. With the result that the expression derived for P_d will be standard, no matter the model of $\boldsymbol{o}$. The standard expression for P_d in the matched filter is formulated as:

$$P_d = Q(Q^{-1}(P_{fa}) - \sqrt{\frac{\boldsymbol{o}^T \boldsymbol{o}}{\sigma^2}}) \tag{4.37}$$

The expression in (4.37) is maximized by maximizing the square root of the energy term for the outliers, $\sqrt{\boldsymbol{o}^T \boldsymbol{o}}$. Which is mathematically equivalent to maximizing $|A|$. This is clear when we substitute $\boldsymbol{o}$ for $c\mathbf{1}_{\tilde{\mathcal{A}}}$ and c for the expression given in (4.29).
For comparison purposes of the performance metrics between the two detector tests, P_d is plotted for a constant $P_{fa} = 0.5$ and varying values of $|A|$.

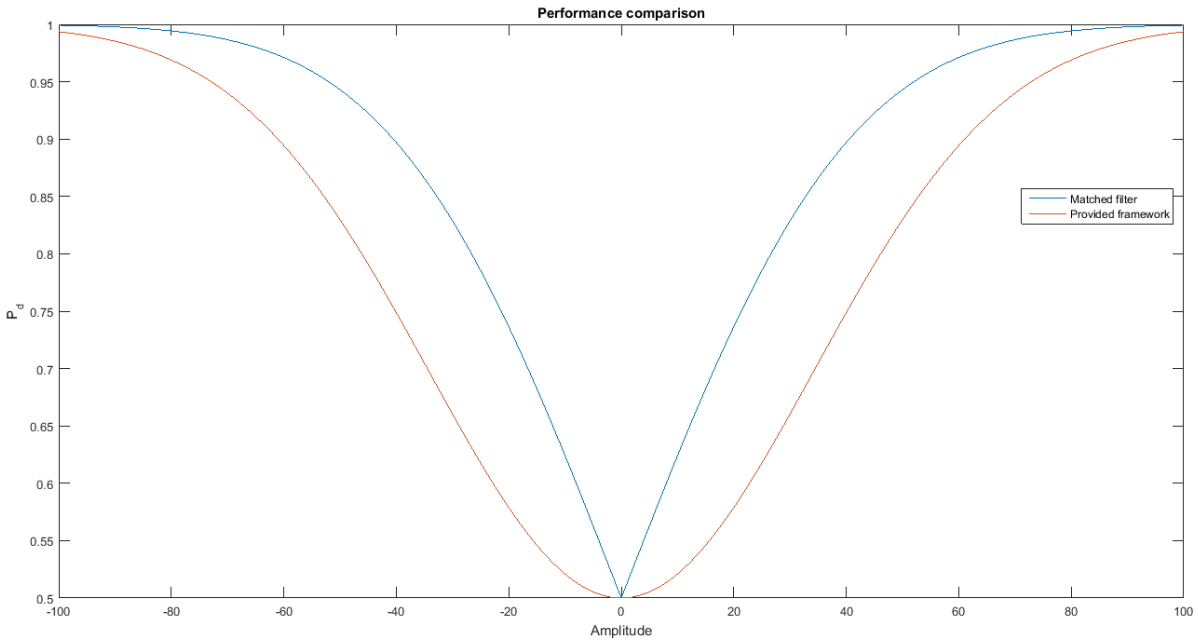


Figure 4.2: P_d is plotted for a varying $|A|$ with noise variance $\sigma^2 = 10$ and $P_{fa} = 0.5$

Despite the lower performance that can be seen in figure 4.2, a numerical evaluation for the proposed framework will be provided. It can be said that the metrics for the provided framework are more realistic of nature, since the structure of $\boldsymbol{o}$ is taken into consideration for the derivation of the probability distributions of $T(\boldsymbol{z})$. Adding to what is already mentioned, the matched filter can thus be assumed to be too optimistic for the auditor.

4.3.2. Time-variant attacks

For M.2 and M.3, it is also possible to derive a threshold function specific for these time-varying attacks. First, we reformulate the hypothesis as

$$\begin{aligned} \mathcal{H}_0 &: \quad \boldsymbol{z} = \boldsymbol{n} \\ \mathcal{H}_1 &: \quad \boldsymbol{z} = \boldsymbol{o} + \boldsymbol{n} \end{aligned} \tag{4.38}$$

The solution for the MLE is derived by assuming that $\mathbf{1}_{\tilde{\mathcal{A}}}$, which is the support of $\boldsymbol{o}$, is known:

$$\min_{\boldsymbol{o}} \|\boldsymbol{z} - \boldsymbol{o}\|_2^2 \tag{4.39}$$

Hence we can formulate our estimation for $\boldsymbol{o}$ as:

$$\boldsymbol{o} = diag(\mathbf{1}_{\tilde{\mathcal{A}}})\boldsymbol{z} \tag{4.40}$$

where the diag operator returns a diagonal matrix with the elements of $\mathbf{1}_{\tilde{\mathcal{A}}}$ on the diagonal. The threshold function can then be found with the log-likelihood ratio test and is simplified to:

$$T(\boldsymbol{z}) = \boldsymbol{z}^T diag(\mathbf{1}_{\tilde{\mathcal{A}}})\boldsymbol{z} \underset{H_0}{\overset{H_1}{\gtrless}} \gamma' \tag{4.41}$$

The distributions for both hypotheses are denoted as:

$$\begin{aligned} \mathcal{H}_0 : \quad & T(\boldsymbol{z}) \sim \chi^2_{|\tilde{\mathcal{A}}|} \\ \mathcal{H}_1 : \quad & T(\boldsymbol{z}) \sim \chi^2_{|\tilde{\mathcal{A}}|}(\mu) \end{aligned} \tag{4.42}$$

Where $|\tilde{\mathcal{A}}|$ is defined as the total cardinality. For M.2 this is defined as:

$$|\tilde{\mathcal{A}}| = T|\mathcal{A}| \tag{4.43}$$

and for M.3:

$$|\tilde{\mathcal{A}}| = \sum_{i=1}^{T} \mathcal{A}_i. \tag{4.44}$$

The χ^2-distribution yields the distribution of the sum of $|\tilde{\mathcal{A}}|$ squared independent normal variables with zero-mean if it is centralized, as in the case of $\mathcal{H}_0$. A non-central χ^2-distribution returns the distribution of the sum of $|\tilde{\mathcal{A}}|$ squared independent normal distributed variables, where the non-centrality parameter is then defined as the squared summation of the means of the independent normal distributed variables. For $\mathcal{H}_1$, the non-centrality parameter is defined as μ:

$$\mu = \frac{\boldsymbol{z}^T diag(\mathbf{1}_{\tilde{\mathcal{A}}})\boldsymbol{z}}{\sigma^2} \tag{4.45}$$

In both cases, the degrees of freedom (*dof*) are then equal to $|\tilde{\mathcal{A}}|$.
The detection probability can then be expressed in terms of the Q-function of a non-central χ^2-distribution.

$$P_d = Q(\gamma')_{\chi^2_{|\tilde{\mathcal{A}}|}(\mu)} \tag{4.46}$$

And the false alarm probability as the Q-function of a centralized χ^2-distribution.

$$P_{fa} = Q(\gamma')_{\chi^2_{|\tilde{\mathcal{A}}|}} \tag{4.47}$$

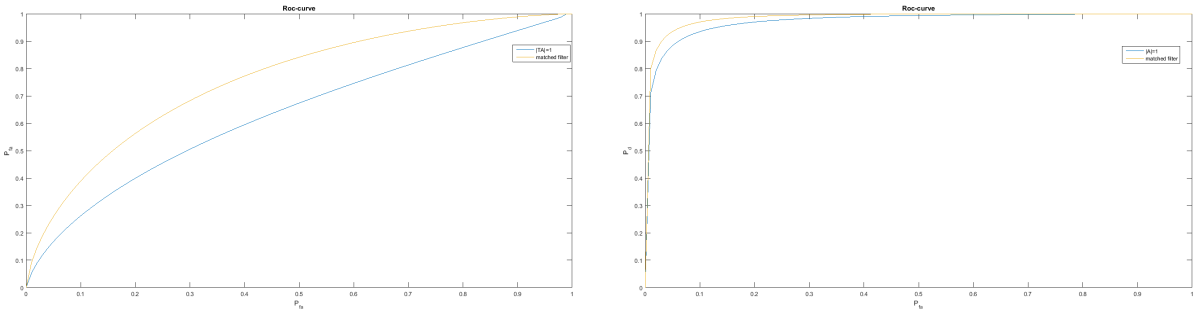
Substitution gives the following formulation for P_d:

$$P_d = Q(Q^{-1}(P_{fa})_{\chi^2_{|\tilde{\mathcal{A}}|}})_{\chi^2_{|\tilde{\mathcal{A}}|}(\mu)} \tag{4.48}$$

An increasing μ results in an increasing P_d. To understand why this is happening, we have to look at how the χ^2-distribution is defined. Increasing μ implies that the means of the squared and summed normal distributions are increasing under $\mathcal{H}_1$. As a result, the distributions under both hypotheses will be more separated. In case that μ is constant and $|\tilde{\mathcal{A}}|$ is increasing, the realisations of the distributions under both hypotheses will be increasing in similarity. This will result in an increasing of the overlap between the two hypothesis distributions. Hence, we observe contradictory trends for μ and $|\tilde{\mathcal{A}}|$. A way to tackle this problem is to take μ as the cost function, and optimize the problem for different constraint values β for the cardinality.

$$f_{P_{d2}} = f_{P_{d3}}(\boldsymbol{z}) = \boldsymbol{z}^T diag(\mathbf{1}_{\tilde{\mathcal{A}}})\boldsymbol{z} \tag{4.49}$$

Contrary to M.1, in case that we would use the matched filter approach for finding a cost function, we can now observe a difference in the optimization problem. The matched filter returns the same cost function as in equation (4.49). But, it does not take $|\tilde{\mathcal{A}}|$ into consideration. This is because the detection probability under the matched filter is increased by increasing the energy signal-to-noise ratio of the signal that is to be detected. Whereas the detection probability denoted in (4.48), is also dependent on the cardinality of $\tilde{\mathcal{A}}$. In figure 4.3b we observe once again that the provided framework shows a lower performance compared to the metrics provided by the matched filter.



(a) ncp=1 (b) ncp=10

Figure 4.3: The ROC-curve is plotted for a matched filter, and minimum degree-of-freedom of 1. For the matched filter $\sqrt{\frac{\boldsymbol{o}^{(1)^T}\boldsymbol{o}^{(1)}}{\sigma^2}}$ is equal to the square root of the non-centrality parameter, $\sqrt{\mu}$. It can be seen that even for a minimum number of *dof* and an increasing μ, the provided framework performs suboptimal.

However, the provided framework can be more selective in identifying the compromised nodes. For example, in case that only a single node is compromised. The framework that is used provides a higher accuracy of estimating the sets of $\mathcal{A}_1$, compared with the matched filter. The optimization problem for the matched filter will then most probably return an estimation for $\mathcal{A}_1$ with a cardinality that reaches the upperbound for the constraint. Thereby selecting nodes that are erroneously labeled as compromised. The last claim is backed up by the fact that the optimization problem for the matched filter is aimed at maximizing the total energy, regardless of how the energy is distributed over the nodes. Maximization of the energy is reached by maximizing the cardinality, while the provided framework is also dependent on $\{|\mathcal{A}_1\}|$ and not only on the total energy.

4.3.3. Lasso Estimator

From the expression in (4.17), we can derive the threshold function by substituting $\boldsymbol{o}$ with the expression in (4.10) as:

$$T(\boldsymbol{z}) = \frac{1}{\sigma^2}\sum_{i=1}^{nT}([\boldsymbol{z}]_i^2 - [\boldsymbol{\lambda}]_i^2)_+ \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \gamma' \tag{4.50}$$

In the equation above $\boldsymbol{\lambda}$ is defined as the vector that contains stacking of T the vectors $\boldsymbol{\lambda}_t$ for all $t \in \{1, 2..T\}$.
At first glance, the distribution of $T(\boldsymbol{z})$ resembles that of a χ^2 distribution that is shifted. However, the $(.)_+$ operator makes the distribution more complicated. Given that the probability density function (pdf) of a (non-central) chi-squared distribution for a **single** degree of freedom is defined as $f_{\chi^2(\mu)}(x)$ where μ is the non-centrality parameter and x is the random variable that is the squared of a random normal variable; and that $F_{\chi^2(\mu)}(x)$ denotes the cumulative distribution function (cdf) of that same distribution,

we can denote the distribution under both hypotheses for a single element z as follows:

$$\begin{aligned}\mathcal{H}_0: \quad T(z) &\sim \begin{cases} f_{\chi^2}(T(z)+\frac{\lambda^2}{\sigma^2}), & \text{if } T(z)>0 \\ F_{\chi^2}(\frac{\lambda^2}{\sigma^2})+f_{\chi^2}(\frac{\lambda^2}{\sigma^2}), & \text{if } T(z)=0 \\ 0, & \text{otherwise} \end{cases} \\ \mathcal{H}_1: \quad T(z) &\sim \begin{cases} f_{\chi^2(\mu)}(T(z)+\frac{\lambda^2}{\sigma^2}), & \text{if } T(z)>0 \\ F_{\chi^2(\mu)}(\frac{\lambda^2}{\sigma^2})+f_{\chi^2(\mu)}(\frac{\lambda^2}{\sigma^2}), & \text{if } T(z)=0 \\ 0, & \text{otherwise} \end{cases}\end{aligned} \tag{4.51}$$

In (4.53), λ refers to the single elemental value for $\boldsymbol{\lambda}$ and μ is defined as

$$\mu = \frac{(|z|-\lambda)_+^T(|z|-\lambda)_+}{\sigma^2} \tag{4.52}$$

In (4.53), the distributions are valid only when $\boldsymbol{z}$ is a scalar, i.e., z. The explanation for the distributions is given in appendix C.
With an increasing number of elements for $\boldsymbol{z}$ and different values of λ it can be observed that deriving a theoretical distribution for the threshold function becomes mathematically challenging. Idem ditto for deriving an expression for the detection and false alarm probability.
Due to the mathematically challenging nature of finding an expression for the distribution of (4.50), we can resort to a simpler expression for the threshold function that will be used as proxy for the Lasso-based method. Namely that of the matched filter. As it is shown in section 2.2, the matched filter is derived by not making any substitutions for $\boldsymbol{o}$ when deriving the threshold function. If we assume $\boldsymbol{o}$ to be known, we will get the following distributions for the matched filter under both hypotheses:

$$\begin{aligned}\mathcal{H}_0: \quad T(\boldsymbol{z}) &\sim \mathcal{N}(0, \sigma^2\boldsymbol{o}^T\boldsymbol{o}) \\ \mathcal{H}_1: \quad T(\boldsymbol{z}) &\sim \mathcal{N}(\boldsymbol{o}^T\boldsymbol{o}, \sigma^2\boldsymbol{o}^T\boldsymbol{o})\end{aligned} \tag{4.53}$$

With the help of the expressions given for P_d and P_{fa} in section 2.2, we can derive the following expression for P_d:

$$P_d = Q(Q^{-1}(P_{fa}) - \sqrt{\frac{\boldsymbol{o}^T\boldsymbol{o}}{\sigma^2}}) \tag{4.54}$$

The energy term $\boldsymbol{o}^T\boldsymbol{o}$ can be substituted with the following term that is derived with the help of (4.10):

$$\boldsymbol{o}^T\boldsymbol{o} = (|\boldsymbol{z}|-\boldsymbol{\lambda})_+^T(|\boldsymbol{z}|-\boldsymbol{\lambda})_+ \tag{4.55}$$

Since we know that the detection probability for the matched filter is increased by increasing the energy term, we can derive the following expression for the cost function:

$$f_{P_{d\,lasso}}(\boldsymbol{\lambda}) = (|\boldsymbol{z}|-\boldsymbol{\lambda})_+^T(|\boldsymbol{z}|-\boldsymbol{\lambda})_+ \tag{4.56}$$

4.4. Alternative Fitting Error Metrics

As stated in chapter 3, the scope of this thesis is to optimize the detection probability. An alternative metric to optimize could be the fitting of our estimated $\boldsymbol{o}$ to our observed data $\boldsymbol{y}$. This can be, for example, measured with the help of the mean squared error. The value that we retrieve is called the least squares error. The optimization problem will then look as follows:

$$\min_{\boldsymbol{o}} \|\boldsymbol{z}-\boldsymbol{o}\|_2^2 \tag{4.57}$$

In the previous section, we only derived an expression for $\boldsymbol{o}$ that is based on the maximum likelihood estimation (and thus also least squares estimation). We can rewrite the problem above as:

$$\min_{\boldsymbol{o}} \quad \boldsymbol{z}^T\boldsymbol{z} + \boldsymbol{o}^T\boldsymbol{o} - 2\boldsymbol{o}^T\boldsymbol{z} \tag{4.58}$$

For M.1, the problem is then denoted as

$$\min_{c} \quad \boldsymbol{z}^T\boldsymbol{z} + c\mathbf{1}_{\tilde{\mathcal{A}}}^T c\mathbf{1}_{\tilde{\mathcal{A}}} - 2c\mathbf{1}_{\tilde{\mathcal{A}}}^T\boldsymbol{z} \tag{4.59}$$

Since we already derived an expression for c (see appendix A), we can now substitute this expression for c in (4.59) and observe that we get the identical optimization problem as in the previous section:

$$\min_{\mathbf{1}_{\tilde{\mathcal{A}}}} \quad \boldsymbol{z}^T\boldsymbol{z} + \left(\frac{\mathbf{1}_{\tilde{\mathcal{A}}}{}^T\boldsymbol{z}}{|\mathcal{A}|T}\right)^2 |\mathcal{A}T| - 2\left(\frac{\mathbf{1}_{\tilde{\mathcal{A}}}{}^T\boldsymbol{z}}{|\mathcal{A}|T}\right)\mathbf{1}_{\tilde{\mathcal{A}}}^T\boldsymbol{z} \tag{4.60}$$

Upon simplification, we get the following problem

$$\max_{\mathbf{1}_{\tilde{\mathcal{A}}}} \quad \frac{(\mathbf{1}_{\tilde{\mathcal{A}}}{}^T\boldsymbol{z})^2}{|\mathcal{A}|} \tag{4.61}$$

In the next chapter, it is explained how this cost function and the cost function given in (4.36) gives the same estimated value for $\mathbf{1}_{\tilde{\mathcal{A}}}$ since that one is the squared version of the other.

In case that we apply the same steps for M.2 and M.3, i.e.,

$$\min_{\mathbf{1}_{\tilde{\mathcal{A}}}} \quad \boldsymbol{z}^T\boldsymbol{z} - \boldsymbol{z}^T diag(\mathbf{1}_{\tilde{\mathcal{A}}})\boldsymbol{z} \tag{4.62}$$

which can be simplified to

$$\max_{\mathbf{1}_{\tilde{\mathcal{A}}}} \quad \boldsymbol{z}^T diag(\mathbf{1}_{\tilde{\mathcal{A}}})\boldsymbol{z}, \tag{4.63}$$

we observe that the problem is identical to the problem derived with the matched filter in the previous section despite that the metric to be optimized is different. However, in this thesis our main concern is the detection of malicious nodes. Hence, our framework is based on the optimization of detection probability. If our problem was based on the estimation of outlying values, given that the network is (guaranteed) compromised, then we can use the framework that is given in this section. This would not differ greatly in terms of the derived optimization problem.

In the next chapter, we will look into how we can solve these optimization problems that we have derived with the help of methods that are used to convert non-convex optimization problems to convex optimization problems.

5

Convexification of Detection Problem

This chapter treats the methods that are used for the convexification of the optimization problems that were derived in the previous section and how to solve the new convexified formulation of those problems. Methods such as binary relaxation, semidefinite programming and the Dinkelbach algorithm will be used for achieving this goal. A relaxation for the constraints will be firstly given. Secondly, we will explain the steps that are going to be used to solve the optimization problem for M.1 using the convex optimization machinery. And also provide the simple binary relaxed formulation for the cost functions of M.2 and M.3. Hereafter, the remaining steps are explained and we then formulate the complete methods for all three models in an algorithmic scheme. The last part gives the derived solution for the Lasso method.

5.1. Constraints relaxations

For the cardinality and the density constraints, relaxations are easy to derive with the binary relaxation method, i.e., a new variable vector $\boldsymbol{x}$ is introduced that relaxes the entries of $\mathbf{1}_{\tilde{\mathcal{A}}}$ to the box $[0,1]$. This relaxation can now be expressed as

$$\mathbf{1}_{\tilde{\mathcal{A}}} \in \{0,1\}^{NT} \longrightarrow \boldsymbol{x} \in [0,1]^{NT}. \tag{5.1}$$

Using this relaxation, the cardinality and density constraints are relaxed to convex constraints. The newly relaxed constraints are then given as

$$K_1(\boldsymbol{x}_i) = \|\boldsymbol{x}_i\|_1 \leq \beta : \forall \quad i \in \{1, ...T\}, \tag{5.2}$$

$$D_1(\boldsymbol{x}_i) = \boldsymbol{d}_i^T \boldsymbol{x}_i \leq \alpha : \forall \quad i \in \{1, ...T\}. \tag{5.3}$$

Note that the cardinality constraint can be assumed to be linear if we use the fact that $\boldsymbol{x}$ has only non-negative entries, i.e., $K_1(\boldsymbol{x}_i) = \mathbf{1}_N^T \boldsymbol{x}_i$.
In order to relax the remaining two cut-dependent constraints in (3.28) and (3.29), a convex and concave relaxation for the cut function in (3.24) must be found.

5.1.1. Convex Cut Relaxation

For the convex relaxation of the cut function, we can make use of the properties of submodular functions [27]. To do so, first let us consider the following. For every modular set function $f(\mathcal{A})$, we can define the so-called Lovasz extension $\hat{f}(\boldsymbol{x})$, which extends the original function through the variable $\boldsymbol{x} \in [0,1]^{|\mathcal{A}|}$ from a powerset $2^{|\mathcal{A}|}$ to a unit cube. The Lovasz extensions has nice properties that can be used to find a convex relaxation for the cut function. These properties are listed as follows:

- if a set function $f(\mathcal{A})$ is submodular, the Lovasz extension of that function $\hat{f}(\boldsymbol{x})$ is convex
- if $f(\mathcal{A})$ is submodular, then $\min_{\mathcal{A}} f(\mathcal{A}) = \min_{\boldsymbol{x}} \hat{f}(\boldsymbol{x})$
- $f(\mathcal{A}) = \hat{f}(\boldsymbol{x}), \quad$ if $\quad \mathbf{1}_{\mathcal{A}} = \boldsymbol{x}$

The properties are very well suited for taking the Lovasz extended function as a convex relaxation for the cut. For the way that we have defined the cut, we need to find the Lovasz extended function of an unweighted and undirected cut. This gives us the well-known relaxation for the cut function:

$$C_1(\boldsymbol{x}_i) = \boldsymbol{x}_i^T \boldsymbol{L} \boldsymbol{x}_i : \forall \quad i \in \{1, ...T\}. \tag{5.4}$$

5.1.2. Concave Cut Relaxation

In order to find a relaxation for the constraint function in (3.28). We need to have a concave relaxation of the cut function. The cut function, as we have defined it to be, can be formulated in different ways. A formulation is already given in (3.24) as a quadratic function. In order to derive a concave relaxation, we will propose an alternative formulation for the cut function. Before giving the formulation, we will first explain the steps that are used in order to derive this formulation.
Firstly, the incidence matrix $\boldsymbol{B}_i$ must be explained. The matrix $\boldsymbol{B}_i$ is defined as a matrix with the dimensions $N \times |\mathcal{E}|$ at time i. Each column of this incidence matrix contains a pair of 1's, the 1's indicate for a given edge which two nodes this edge connects, e.g., figure 5.1:

$$\boldsymbol{B}_i = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccccc} a & b & c & d & e & f & g & h \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \Rightarrow$$

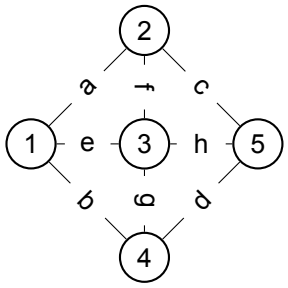


Figure 5.1: An example of an incidence matrix with the corresponding graph. The nodes are indicated with numbers and the edges are indicated with a letter

Thus if the l'th edge connects the nodes j and k then $[\boldsymbol{B}_i]_{j,l}$ and $[\boldsymbol{B}_i]_{k,l}$ are equal to 1 and the rest of the column elements of $[\boldsymbol{B}_i^T]_{*l}$ are equal to zero. The term $*l$ as a subscript for a matrix is a general term for the l'th column. Multiplying this matrix with $\mathbf{1}_{\mathcal{A}}$ will give us a vector that looks as follows:

$$\boldsymbol{B}_i^T \mathbf{1}_{\mathcal{A}_i} = [2\ 2....2..1\ 1...0\ 0] : \forall \quad i \in \{1, ...T\}. \tag{5.5}$$

The value of 2 occurs in $[\boldsymbol{B}_i^T \mathbf{1}_{\mathcal{A}_i}]_l$ when the l'th edge connects two nodes which are both in $\mathbf{1}_{\mathcal{A}i}$; if only one of the nodes belongs to $\mathbf{1}_{\mathcal{A}i}$ then the entry in $[\boldsymbol{B}_i^T \mathbf{1}_{\mathcal{A}_i}]_l$ yields 1; otherwise we will get 0 for the entry. Now, if we subtract an all ones vector from the above-mentioned vector, we will get a vector that looks as follows:

$$(\boldsymbol{B}_i^T \mathbf{1}_{\mathcal{A}_i}) - \mathbf{1}_{|\epsilon|}^T = [1\ 1....1..0\ 0... - 1\ -1] : \forall \quad i \in \{1, ...T\}. \tag{5.6}$$

The values of 2 are subtracted to 1, values of 1 are subtracted to 0 and values of 0 are subtracted to -1. By taking the $l1$-norm of this vector, we obtain the number of edges that does not belong to the cut set, i.e. the cut of the complement of $\mathbf{1}_{\mathcal{A}i}$

$$C(\mathbf{1}_{\mathcal{A}i}^c) = \left\| (\boldsymbol{B}_i^T \mathbf{1}_{\mathcal{A}_i}) - \mathbf{1}_{|\epsilon|}^T \right\|_1 : \forall \quad i \in \{1, ...T\}. \tag{5.7}$$

Here the superscript c stands for the complement set of $\mathbf{1}_{\mathcal{A}}$. The final step for deriving the cut of $\mathbf{1}_{\mathcal{A}i}$ is achieved by subtracting $C(\mathbf{1}_{\mathcal{A}i}^c)$ from the total number of edges $|\mathcal{E}|$

$$C(\mathbf{1}_{\mathcal{A}_i}) = |\mathcal{E}| - \left\| (\boldsymbol{B}_i^T \mathbf{1}_{\mathcal{A}_i}) - \mathbf{1}_{|\epsilon|}^T \right\|_1 : \forall \quad i \in \{1, ...T\}. \tag{5.8}$$

What can be observed is that upon binary relaxation of $\mathbf{1}_{\mathcal{A}_i}$:

$$C_2(\boldsymbol{x}_i) = |\mathcal{E}| - \left\| \boldsymbol{B}_i^T \boldsymbol{x}_i - \mathbf{1}_{|\epsilon|}^T \right\|_1 : \forall \quad i \in \{1, ...T\}, \tag{5.9}$$

we get a different function than in (5.4). In fact, it is easy to show with the help of the composition rule [28] that (5.9) is a concave function. With the help of the composition rule we can prove that

$\left\| \boldsymbol{B}_i^T \boldsymbol{x}_i - \mathbf{1}_{|\epsilon|}^T \right\|$ is convex since that it is the composition of an l1-norm and an affine function. Hence, the function in (5.9) is concave.
In (3.27), the relation between the density, cut and the dependent edge set are given. We can now exploit this relation for a convex relaxation of the dependent edge set:

$$B_1(\boldsymbol{x}_i) = \frac{1}{2}(D_1(\boldsymbol{x}_i) - C_2(\boldsymbol{x}_i)) \leq \rho : \forall \quad i \in \{1, ...T\}. \tag{5.10}$$

5.2. Relaxation of Performance Metrics

Now that the constraints are relaxed, the next step would be to find a convex relaxation for the cost functions related to the performance metrics.

5.2.1. Time-invariant attacks

First, notice that the derived cost function for M.1 in chapter 4 is a fractional function, i.e. a function that consists of a numerator $f(x)$ and denominator $g(x)$. Hence, the fractional function is then defined as $\frac{f(x)}{g(x)}$. For fractional functions there are fractional programming methods that can be used for their optimization. For instance, the Dinkelbach method [29] is a well known method that is used for the solution of fractional problems. In order to utilize the method, there are conditions that must be met. We define a variable x that contains all the variable elements that are used in the function, i.e., all the elements in the vectors and matrices that are a variable. The fractional function $\frac{f(x)}{g(x)}$ that must be minimized for x must hold the following conditions for the Dinkelbach algorithm to be applicable [29].

- $f(x)$ must be convex for $x \in \mathcal{C}$
- $g(x)$ must be concave for $x \in \mathcal{C}$
- $f(x) : \mathcal{C} \rightarrow \mathbb{R}$
- $g(x) : \mathcal{C} \rightarrow (0, +\infty)$

Here $\mathcal{C}$ is the convex constraint domain for x. Hence, $\mathcal{C}$ has the same dimension as the variable set x. Thus, $f(x) \in \mathbb{R}$ for $x \in \mathcal{C}$. And $g(x)$ maps to the set of positive values for $x \in \mathcal{C}$. Hence, we need to first find a relaxation for the nominator and the denominator that satisfies these conditions.
Let us recall the cost function

$$f_{P_{d1}}(\mathbf{1}_{\tilde{\mathcal{A}}}) = \frac{|\boldsymbol{z}^T \mathbf{1}_{\tilde{\mathcal{A}}}|}{\sqrt{|\tilde{\mathcal{A}}|}} \tag{5.11}$$

In order to convexify the nominator part of the cost function, it can be squared. The maximum argument of the squared cost function is equal to the maximum argument of the original cost function. And since the Dinkelbach algorithm is derived for minimization instead of maximization, we reformulate the cost function as follows:

$$\hat{f}_{P_{d1}}(\mathbf{1}_{\tilde{\mathcal{A}}}) = -\frac{(\mathbf{1}_{\tilde{\mathcal{A}}}{}^T \boldsymbol{z})^2}{|\tilde{\mathcal{A}}|} \tag{5.12}$$

The denominator in (5.12) is equal to the cardinality of $\tilde{\mathcal{A}}$. In chapter 5.1, we explained how the cardinality is relaxed to a linear expression. Benefiting from the convexity of a linear function, we could put the minus sign in the denominator. Unfortunately, the minus sign in the denominator, i.e., $g(x)$, will falsify the condition $g(x) : \mathcal{C} \rightarrow (0, +\infty)$. Hence, a convex relaxation must be find for the nominator $-(\mathbf{1}_{\tilde{\mathcal{A}}}{}^T \boldsymbol{z})^2$.

SDP Formulation

For dealing with non-convex quadratic functions [30], we can make use of semidefinite programing relaxation methods. For instance, the nominator in (5.12) can be reformulated as follows:

$$-\mathbf{1}_{\tilde{\mathcal{A}}}{}^T \boldsymbol{z}\boldsymbol{z}^T \mathbf{1}_{\tilde{\mathcal{A}}} = -tr(\boldsymbol{z}\boldsymbol{z}^T \mathbf{1}_{\tilde{\mathcal{A}}} \mathbf{1}_{\tilde{\mathcal{A}}}{}^T) \tag{5.13}$$

The operator $tr()$ returns the trace of a square matrix.
After applying binary relaxation, we can observe that we are dealing with a concave nominator due to the

$(-)$ sign. With the help of semidefinite relaxation, a new variable matrix for the relaxation of the 'support matrix' $\mathbf{1}_{\tilde{\mathcal{A}}}\mathbf{1}_{\tilde{\mathcal{A}}}^T$ can be introduced. If we define a new matrix $\boldsymbol{X}_B \in \mathbb{R}^{NT\times NT}$ that should be equal to $\boldsymbol{x}\boldsymbol{x}^T$, then $\boldsymbol{X}_B$ should be a positive semidefinite and rank-1 matrix. The semidefinite programming provides a relaxation for the problem by dropping the rank-1 constraint for $\boldsymbol{X}_B$ and keeping the semidefinite constraint since this yields a convex subset for matrices. If we reformulate

$$\boldsymbol{z}\boldsymbol{z}^T = \boldsymbol{Z}, \tag{5.14}$$

the relaxed formulation for (5.13) can then be denoted as

$$-\operatorname{tr}(\boldsymbol{Z}\boldsymbol{X}_B) \tag{5.15}$$

With the following constraints for $\boldsymbol{X}_B$:

$$\begin{aligned}
&\boldsymbol{X} \in \mathbb{R}^{N\times N},\\
&0 \le [\boldsymbol{X}]_{i,j} \le 1 \quad : \forall \quad i,j \in \{1,2,3\ldots N\},\\
&\boldsymbol{X} \in \mathbb{S}^N,\\
&\boldsymbol{X} \succeq 0,\\
&\boldsymbol{X}_B = \mathbb{I}_T \otimes \boldsymbol{X},
\end{aligned} \tag{5.16}$$

The symmetric constraint is given with the help of the symbol $\mathbb{S}^N$, which stands for the symmetric domain of matrices with dimensions $N \times N$. The positive semidefinite constraint is presented with the help of the symbol $\succeq$. The matrix $\boldsymbol{X}_B$ is actually the Kronecker product, $\otimes$, of a different semidefinite relaxation matrix with an all ones matrix $\mathbb{I}_T$ with dimensions $T \times T$. The reason for this is the assumed time-invariancy of the support set. In equation (5.16) the matrix $\boldsymbol{X}$ is the semidefinite relaxation matrix for the constant vector $\boldsymbol{x}_c$ where $\boldsymbol{x}_c = \boldsymbol{x}_i : \forall \quad i \in \{1,\ldots.T\}$. In order to tighten the relation between $\boldsymbol{X}_B$ and $\boldsymbol{x}$, a second semidefinite constraint can be included:

$$\boldsymbol{X}_B - \boldsymbol{x}\boldsymbol{x}^T \succeq 0. \tag{5.17}$$

This relaxation, which is called the Shor's relaxation [31], is needed because the denominator is expressed as a term of $\boldsymbol{x}$. With the help of the Schur complement, we can manipulate this relaxation into a semidefinite constraint [32]

$$\begin{bmatrix} \boldsymbol{X}_B & \boldsymbol{x} \\ \boldsymbol{x}^T & 1 \end{bmatrix} \succeq 0 \tag{5.18}$$

From experiments, it can be observed that without enforcing semidefinite contraints, we observe that the method returns good results. Therefore, we conclude that it works in practice.
Since the computational complexity of the semidefinite constraints are very high, i.e., long and unrealistic simulation times, the decision is taken to drop the semidefinite constraints. However, we must introduce new relaxations that tighten the relationship between $\boldsymbol{X}_B$ and $\boldsymbol{x}$. The final optimization problem that will be used for the Dinkelbach method is given as follows:

$$\begin{aligned}
\min_{\boldsymbol{X}_B,\boldsymbol{X},\boldsymbol{x}_c,\boldsymbol{x}} \quad & \tilde{f}_{P_{d1}} = -\frac{Tr(\boldsymbol{Z}\boldsymbol{X}_B)}{\|\boldsymbol{x}\|_1} \\
\text{subject to} \quad & B_1(\boldsymbol{x}_c) \le \rho \quad or \quad C_1(\boldsymbol{x}_c) \le \rho, \\
& K_1(\boldsymbol{x}_c) \le \beta, \\
& D_1(\boldsymbol{x}_c) \le \alpha, \\
& \boldsymbol{X} \in \mathbb{S}^N, \\
& \boldsymbol{X}_B = \mathbb{I}_T \otimes \boldsymbol{X}, \\
& \boldsymbol{x}_c = \boldsymbol{x}_i && : \forall \quad i \in \{1,\ldots.T\}, \\
& 0 \le [\boldsymbol{X}]_{i,j} \le 1 && : \forall \quad i,j \in \{1,2,3\ldots N\}, \\
& 0 \le [\boldsymbol{x}_c]_i \le 1 && : \forall \quad i \in \{1,2,3\ldots N\}, \\
& [\boldsymbol{X}]_{i,i} = [\boldsymbol{x}_c]_i && : \forall \quad i \in \{1,2,3\ldots N\}, \\
& Tr(\boldsymbol{Z}\boldsymbol{X}_B) \ge 0, \\
& \left\|[\boldsymbol{X}]_{i*}\right\|_1 \le \beta && : \forall \quad i \in \{1,2,3\ldots N\}, \\
& \|\boldsymbol{X}\|_F \le \|\boldsymbol{x}_c\|_1
\end{aligned} \tag{5.19}$$

The newly introduced constraints must be justified first. We start with

$$[\boldsymbol{X}]_{ii} = [\boldsymbol{x}_c]_i : \forall \quad i \in \{1,2,3\ldots N\} \tag{5.20}$$

This follows from the following:

$$[\mathbf{1}_{\mathcal{A}}\mathbf{1}_{\mathcal{A}}^{T}]_{i,i} = [\mathbf{1}_{\mathcal{A}}]_{i} : \forall \quad i \in \{1, 2, 3 \ldots N\} \tag{5.21}$$

Hence we enforce this also for $\boldsymbol{x}_c$ and $\boldsymbol{X}$.

The second constraint that is going to be explained is:

$$Tr(\boldsymbol{Z}\boldsymbol{X}_B) \geq 0. \tag{5.22}$$

Since we know that

$$(\mathbf{1}_{\bar{\mathcal{A}}}{}^{T}\boldsymbol{z})^2 = \mathbf{1}_{\bar{\mathcal{A}}}{}^{T}\boldsymbol{Z}\mathbf{1}_{\bar{\mathcal{A}}} \geq 0, \tag{5.23}$$

we can also enforce this constraint:

$$(\boldsymbol{x}^{T}\boldsymbol{z})^2 = \boldsymbol{x}^{T}\boldsymbol{Z}\boldsymbol{x} \geq 0. \tag{5.24}$$

From which we can also derive

$$Tr(\boldsymbol{Z}\boldsymbol{X}_B) \geq 0. \tag{5.25}$$

For the third constraint:

$$\left\|[\boldsymbol{X}]_{i*}\right\|_1 \leq \|\boldsymbol{x}_c\|_1 : \forall \quad i \in \{1, 2, 3 \ldots N\} \tag{5.26}$$

we know that the column i of matrix $\mathbf{1}_{\mathcal{A}}\mathbf{1}_{\mathcal{A}}^{T}$ is equal to $\mathbf{1}_{\mathcal{A}}$ if $[\mathbf{1}_{\mathcal{A}}]_i = 1$. With relaxation, this turns into an inequality:

$$\left\|[\boldsymbol{x}\boldsymbol{x}^{T}]_{i*}\right\|_1 \leq \|\boldsymbol{x}_c\|_1 : \forall \quad i \in \{1, 2, 3 \ldots N\} \tag{5.27}$$

Hence we can enforce for the columns of $\boldsymbol{X}$ an upperbound for the l1-norm.

The last constraint that needs to be explained is:

$$\|\boldsymbol{X}\|_F \leq \|\boldsymbol{x}_c\|_1 \tag{5.28}$$

First, note that we have the following equation:

$$\sqrt{Tr(\mathbf{1}_{\mathcal{A}}^{T}\mathbf{1}_{\mathcal{A}}\mathbf{1}_{\mathcal{A}}^{T}\mathbf{1}_{\mathcal{A}})} = \sqrt{Tr(\mathbf{1}_{\mathcal{A}}^{T}\mathbf{1}_N\mathbf{1}_{\mathcal{A}}^{T}\mathbf{1}_N)} \tag{5.29}$$

where $\mathbf{1}_N$ is the all-ones vector of size N.
In case of binary relaxation, this equality turns into the following inequality:

$$\sqrt{Tr(\boldsymbol{x}_c^{T}\boldsymbol{x}_c\boldsymbol{x}_c^{T}\boldsymbol{x})} \leq \sqrt{Tr(\boldsymbol{x}_c^{T}\mathbf{1}_N\boldsymbol{x}_c^{T}\mathbf{1}_N)} \tag{5.30}$$

In case that we write $\boldsymbol{x}_c\boldsymbol{x}_c^{T}$ as $\boldsymbol{X}$, we get:

$$\sqrt{Tr(\boldsymbol{X}\boldsymbol{X})} \leq \sqrt{Tr(\boldsymbol{x}_c^{T}\mathbf{1}_N\boldsymbol{x}_c^{T}\mathbf{1}_N)} \tag{5.31}$$

Because of symmetry we see that the left side of the inequality can be reformulated as the Frobenius norm:

$$\|\boldsymbol{X}\|_F = \sqrt{Tr(\boldsymbol{X}\boldsymbol{X}^{H})} \tag{5.32}$$

While the right side can be simplified to:

$$\sqrt{Tr(\boldsymbol{x}_c^{T}\mathbf{1}_N\boldsymbol{x}_c^{T}\mathbf{1}_N)} = \boldsymbol{x}_c^{T}\mathbf{1}_N = \|\boldsymbol{x}_c\|_1 \tag{5.33}$$

FP Formulation

The Dinkelbach algorithm can now be used to solve the newly defined problem. For the Dinkelbach algorithm we exploit the fact that if there is an optimal point λ^* for the cost function, then the following holds [29]:

$$x^* = \underset{x}{\operatorname{argmin}} f(x) - \lambda^* g(x) = \underset{x}{\operatorname{argmin}} \frac{f(x)}{g(x)} \tag{5.34}$$

Furthermore, if F is defined as $F(\lambda) = \min_x f(x) - \lambda g(x)$, then $F(\lambda) < 0$ if $\lambda > \lambda^*$ and $F(\lambda) > 0$ if $\lambda < \lambda^*$. This can be proven as follows:
If we assume that

$$\min_x f(x) - t\lambda^* g(x) < 0 \tag{5.35}$$

Any derived x_λ for $x_\lambda \in \mathcal{C}$ will give

$$\frac{f(x_\lambda)}{g(x_\lambda)} = \lambda < t\lambda^* \tag{5.36}$$

Since we know that there is not a λ for a given $x_\lambda \in \mathcal{C}$ that is lower than λ^*, we can conclude that the inequality can hold only when $t > 1$. Hence,

$$f(x_\lambda) - \lambda g(x_\lambda) < 0 \tag{5.37}$$

is true given that $\lambda = t\lambda^*$ and $t > 1$ and thus $\lambda > \lambda^*$
In the same way we can prove that $F(\lambda) > 0$ for $\lambda < \lambda^*$. The strategy of the Dinkelbach algorithm is to recursively find the matching pairs of x and λ that will eventually give a value for $F(\lambda)$ that is close enough to zero. In case there is an initially chosen lambda as

$$\lambda_0 = \frac{f(x_0)}{g(x_0)} \tag{5.38}$$

We know that $F(\lambda_0) \leq 0$ if there is a guarantee that $x_0 \in \mathcal{C}$. Hence by substitution we get

$$\min_{x_1} f(x_1) - \lambda_0 g(x_1) = \min_{x_1} f(x_1) - \frac{f(x_0)}{g(x_0)} g(x_1) \leq 0 \tag{5.39}$$

From which it is provable that

$$\frac{f(x_1)}{g(x_1)} \leq \frac{f(x_0)}{g(x_0)} \tag{5.40}$$

If we calculate the new λ_1, the same steps can be taken again to find a variable set x_2 that provides a smaller point for the optimization problem, i.e., λ_2. The goal is to continue recursively in this fashion until the algorithm iterates towards the minimum of the cost function. Where then the inequality in equation (5.40) turns into equality.

5.2.2. Time-variant attacks

The relaxation for the cost functions of M.2 and M.3 are easier to derive. By exploiting its formulation, a simple binary relaxation gives a direct linear relaxation of their cost function.

$$\tilde{f}_{P_{d2}} = \tilde{f}_{P_{d3}}(\boldsymbol{x}) = \boldsymbol{z}^T diag(\boldsymbol{x})\boldsymbol{z} \tag{5.41}$$

For M.2, this will give the following relaxed optimization problem:

$$\begin{aligned} \max_{\boldsymbol{x},\boldsymbol{x}_c} \quad & \tilde{f}_{P_{d2}}(\boldsymbol{x}) \\ \text{subject to} \quad & B_1(\boldsymbol{x}_c) \leq \rho \quad or \quad C_1(\boldsymbol{x}_c) \leq \rho, \\ & K_1(\boldsymbol{x}_c) \leq \beta, \\ & D_1(\boldsymbol{x}_c) \leq \alpha, \\ & 0 \leq [\boldsymbol{x}_c]_i \leq 1 \qquad : \forall \quad i \in \{1, \ldots N\}, \\ & \boldsymbol{x}_c = \boldsymbol{x}_i \qquad : \forall \quad i \in \{1, \ldots . T\} \end{aligned} \tag{5.42}$$

And for M.3 we get:

$$\begin{aligned}
\max_{\boldsymbol{x}} \quad & \tilde{f}_{P_{d3}}(\boldsymbol{x}) \\
\text{subject to} \quad & B_1(\boldsymbol{x}_i) \leq \rho \quad or \quad C_1(\boldsymbol{x}_i) \leq \rho \quad : \forall \quad i \in \{1, \ldots . T\}, \\
& K_1(\boldsymbol{x}_i) \leq \beta \quad : \forall \quad i \in \{1, \ldots . T\}, \\
& D_1(\boldsymbol{x}_i) \leq \alpha \quad : \forall \quad i \in \{1, \ldots . T\}, \\
& 0 \leq [\boldsymbol{x}_i]_j \leq 1 \quad : \forall \quad j \in \{1, \ldots N\} \wedge i \in \{1, \ldots . T\},
\end{aligned} \tag{5.43}$$

The relaxation of the problems for M.2 and M.3 is not complete with only the optimization problems given in (5.42) and (5.43). As stated earlier in section 4.3.2, we need to solve these problems for different values of the cardinality constraints, i.e. different values of β. This way we can check for which values of β we will get the maximum value for P_d.
The same method can also be applied to M.1. The optimization problem in (5.19) is then approached in a similar fashion by 'simplifying' the cost function as

$$\begin{aligned}
\min_{\boldsymbol{X}_B, \boldsymbol{X}, \boldsymbol{x}_c, \boldsymbol{x}} \quad & -Tr(\boldsymbol{Z}\boldsymbol{X}_B) \\
\text{subject to} \quad & B_1(\boldsymbol{x}) \leq \rho \quad or \quad C_1(\boldsymbol{x}_c) \leq \rho, \\
& K_1(\boldsymbol{x}_c) \leq \beta, \\
& D_1(\boldsymbol{x}_c) \leq \alpha, \\
& \boldsymbol{X} \in \mathbb{S}^N, \\
& \boldsymbol{X}_B = \mathbb{I}_{\mathbb{T}} \otimes \boldsymbol{X}, \\
& \boldsymbol{x}_c = \boldsymbol{x}_i \quad : \forall \quad i \in \{1, \ldots . T\}, \quad , \\
& 0 \leq [\boldsymbol{X}]_{i,j} \leq 1 \quad : \forall \quad i, j \in \{1, 2, 3 \ldots N\}, \\
& 0 \leq [\boldsymbol{x}_c]_i \leq 1 \quad : \forall \quad i \in \{1, 2, 3 \ldots N\}, \\
& [\boldsymbol{X}]_{ii} = [\boldsymbol{x}_c]_i \quad : \forall \quad i \in \{1, 2, 3 \ldots N\}, \\
& Tr(\boldsymbol{Z}\boldsymbol{X}_B) \geq 0, \\
& \left\| [\boldsymbol{X}]_{i*} \right\| \leq \beta \quad : \forall \quad i \in \{1, 2, 3 \ldots N\}, \\
& \|\boldsymbol{X}\|_F \leq \|\boldsymbol{x}_c\|_1
\end{aligned} \tag{5.44}$$

and solve this optimization problem for different values of β. The gathered variable arguments for the different values of β are then put into the cost function $\tilde{f}_{P_{d1}}$, the argument that yields the optimum value for $\tilde{f}_{P_{d1}}$ is then taken as the solution for the method.
However, for an increasing number of nodes together with an increasing β, the optimization problems that must be performed will also increase. Also, since we are able to mathematically express the relationship between the cardinality of $\tilde{\mathcal{A}}$ and the term $-Tr(\boldsymbol{Z}\boldsymbol{X}_B)$ as a ratio for the optimization of P_d in M.1, it is more convenient to exploit this by using the Dinkelbach algorithm. A guaranty is not given that the above mentioned method yields the same maximum argument as the Dinkelbach method.
In the future, results can be compared between the above mentioned method and the Dinkelbach method. We can also compare other factors like the used resources and simulation time for both methods in the future.

5.2.3. Convex Formulation

Now that the intermediate steps are explained for all three models, we can now formulate the complete algorithms in an algorithmic scheme for all three models separately. Before we do this, we must first define functions that returns the argument of the optimization problems. For M.1, we first define a function

$$\begin{aligned}
F(\lambda) = \quad & \underset{\mathcal{X}}{\operatorname{argmin}} \quad -Tr(\boldsymbol{X}_B\boldsymbol{Z}) - \lambda_0 \|\boldsymbol{x}\|_1 \\
& \text{subject to} \quad \mathcal{X} \in \mathcal{C}_2
\end{aligned} \tag{5.45}$$

that returns a variable set $\mathcal{X} = (\boldsymbol{X}_B, \boldsymbol{X}, \boldsymbol{x}_c, \boldsymbol{x})$. The set $\mathcal{C}_2$ is defined as the constraint set given in (5.19). Furthermore, we define a second constraint set for $\mathbf{1}_{\tilde{\mathcal{A}}}$ as $\mathcal{C}_1$. The constraint set $\mathcal{C}_1$ is the 'hard' constraint set given in (4.24) or (4.25). The algorithm for M.1 looks as follows:

Algorithm 1 The optimization algorithm for M.1

Input: $\boldsymbol{z}$
output: $\mathbf{1}_{\mathcal{A}}^{*}$
Initialize: ϵ as positive stopping criterion close to 0 and $\mathbf{1}_{\mathcal{A}}$ such that $\mathbf{1}_{\mathcal{A}} \in \mathcal{C}_1$
$\lambda_0 := \frac{\mathbf{1}_{\bar{\mathcal{A}}}{}^T \boldsymbol{Z} \mathbf{1}_{\bar{\mathcal{A}}}}{\|\mathbf{1}_{\bar{\mathcal{A}}}\|}$
$\mathcal{X} := F(\lambda_0)$
$k := 1$
$\lambda_k := \frac{-Tr(\boldsymbol{X}_B \boldsymbol{Z})}{\|\boldsymbol{x}\|_1}$
while $| -Tr(\boldsymbol{X}_B \boldsymbol{Z}) - \lambda_{k-1} \|\boldsymbol{x}\|_1 | \geq \epsilon$ **do**
    $k := k + 1$
    $\mathcal{X} := F(\lambda_{k-1})$
    $\lambda_k := \frac{-Tr(\boldsymbol{X}_B \boldsymbol{Z})}{\|\boldsymbol{x}\|_1}$
end while
Round $\boldsymbol{x}_c$ by inserting $\boldsymbol{x}_c$ into Rounding Algorithm for M.1 and **return**: $\mathbf{1}_{\mathcal{A}}^{*}$
return $\mathbf{1}_{\mathcal{A}}^{*}$

Similarly, for M.2 we must first define a function that returns the argument as:

$$F_2(\beta) = \begin{array}{ll} \underset{\mathcal{X}}{\operatorname{argmax}} & \tilde{f}_{P_{d2}}(\boldsymbol{x}) \\ \text{subject to} & \mathcal{X} \in \mathcal{C}_3 \end{array}, \tag{5.46}$$

where $\mathcal{C}_3$ is the constraint set given in (5.42). The variable β is the cardinality constraint given in (5.42). The function in (5.46) returns a set $\mathcal{X}$ that is defined as $\mathcal{X} = (\boldsymbol{x}_c, \boldsymbol{x})$. For M.2, the convex optimization problem is solved for β_{max} different cardinality constraints, giving us β_{max} different answers for $\mathcal{X}$ that will be denoted as $\mathcal{X}_i = (\boldsymbol{x}_c, \boldsymbol{x})$ for $i = \{1, 2..\beta_{max}\}$. From these answers, we need to choose the one that gives us the highest value for P_d.

Algorithm 2 The optimization algorithm for M.2

Input: $\boldsymbol{z}$
output: $\mathbf{1}_{\mathcal{A}}^{*}$
for $i \leftarrow 1$ **to** β_{max} **do**
    $\beta := i$
    $\mathcal{X}_i := F_2(\beta)$
    insert $(\boldsymbol{x}, i) : \boldsymbol{x}_c \in \mathcal{X}_i$ into Rounding Algorithm for M.2 and return: $\mathbf{1}_{\mathcal{A}}{}^{i*}$
end for
Choose $\mathbf{1}_{\mathcal{A}}{}^{i*}$ from $i \in \{1, \ldots \beta_{max}\}$ as $\mathbf{1}_{\mathcal{A}}^{*}$ that gives the highest values for :
$P_d = Q(Q^{-1}(P_{fa})_{\chi^2_{|\{\mathcal{A}_1\}|}})_{\chi^2_{|\{\mathcal{A}_1\}|}(\mu)}$
Return: $\mathbf{1}_{\mathcal{A}}^{*}$

And for M.3 there are β_{max}^T different answers possible, for T observed timeframes. This is because every $\boldsymbol{x}_i$ for a given timeframe i has β_{max} different answers. The final estimate is found by combining all the possible combinations for the given timeframes to find the optimal P_d.
We first define a function $F_3(\beta)$ as:

$$F_3(\beta) = \begin{array}{ll} \underset{\mathcal{X}}{\operatorname{argmax}} & \tilde{f}_{P_{d3}}(\boldsymbol{x}) \\ \text{subject to} & \mathcal{X} \in \mathcal{C}_4 \end{array}, \tag{5.47}$$

where $\mathcal{C}_4$ is the constraint set given in (5.43). The variable β is the cardinality constraint given in (5.43). The function in (5.47) returns a set $\mathcal{X}$ that is defined as $\mathcal{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_T)$.

Algorithm 3 The optimization algorithm for M.3

Input: $\boldsymbol{z}$
output: $\mathbf{1}^*_{\tilde{\mathcal{A}}}$
for $i \leftarrow 1$ **to** β_{max} **do**
  $\beta := i$
  $\mathcal{X}_i := F_3(\beta)$
end for
for $i \leftarrow 1$ **to** β_{max} **do**
  for $j \leftarrow 1$ **to** T **do**
    insert $(\boldsymbol{x}_j, i, j) : \boldsymbol{x}_j \in \mathcal{X}_i$ into Rounding Algorithm for M.3 and return: $\mathbf{1}_{\mathcal{A}}{}^{i*}_{j}$
  end for
end for
Define for all ${\beta_{max}}^T$ possible combinations a stacked support vector $\mathbf{1}^{(1)i*}_{\mathcal{A}}$ where $i \in \{1, 2, \ldots {\beta_{max}}^T\}$
Choose: $\mathbf{1}^{(1)i*}_{\mathcal{A}}$ from $i \in \{1, 2, \ldots {\beta_{max}}^T\}$ as $\mathbf{1}^*_{\tilde{\mathcal{A}}}$ that gives the highest values for:
$P_d = Q(Q^{-1}(P_{fa})_{\chi^2_{|\{\mathcal{A}_1\}|}})_{\chi^2_{|\{\mathcal{A}_1\}|}(\mu)}$
return $\mathbf{1}^*_{\tilde{\mathcal{A}}}$

The rounding algorithm for all three models is given in Appendix B. All three of the rounding algorithms are based on Bernoulli processes that have the relaxed solution vector $\boldsymbol{x}$ as input. In case there are c Bernoulli processes performed for $\boldsymbol{x}$, there are c different vectors where the final answer $\mathbf{1}^*_{\mathcal{A}}$ can be chosen out of. The rounding algorithm returns the vector of Bernoulli processes for $\boldsymbol{x}$ that gives the optimal value for f_{P_d}, together with an acknowledgment for the meeting of the constraint set. Otherwise, it looks at the answer that gives the second best value for f_{P_d}, given that the constraints are met. It will continue in this fashion until we have a match with the contraint set.

5.3. Lasso-based Detection

For the Lasso technique, a relaxation for the cost function is not necessary in order to find the optimum answer. Since we know that the regression happens linearly, i.e., $(|\boldsymbol{z}_i| - \boldsymbol{\lambda}_i)_+$, we need to increase the regularization parameter $\boldsymbol{\lambda}_i$ until the cardinality constraint

$$K(\boldsymbol{\lambda}_i) = \left\| sgn(\boldsymbol{z}_i)(|\boldsymbol{z}_i| - \boldsymbol{\lambda}_i)_+ \right\|_0 \leq \beta : \forall \quad i \in \{1, 2, ..T\} \tag{5.48}$$

is met for all $i \in \{1, 2 \ldots T\}$. The regularization parameter $\boldsymbol{\lambda}_i$ that gives the highest value for the cost function while the constraints are met will be equal to the magnitude of the element of $\boldsymbol{z}_i$ with the $(\beta+1)$'th largest magnitude. That is the magnitude of the element with the $N-\beta$'th smallest magnitude. Thus, if we would like to solve the optimization problem:

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & f_{P_{dlasso}}(\boldsymbol{\lambda}) \\ \text{subject to} \quad & K(\boldsymbol{\lambda}_i) \leq \beta : \forall \quad i \in \{1, 2, ..T\}, \end{aligned}' \tag{5.49}$$

we first sort the magnitudes of the elements in $\boldsymbol{z}_i$ in descending order for all $i \in \{1, 2 \ldots T\}$. And then we take the constant value in the vector $\boldsymbol{\lambda}_i$ as the $(\beta+1)$'th element of the sorted vector, this process is repeated for all $i \in \{1, 2 \ldots T\}$. After this, we get an estimation for $\boldsymbol{o}$ as

$$\boldsymbol{o}^*_i = sgn(\boldsymbol{z}_i)(|\boldsymbol{z}_i| - \boldsymbol{\lambda}^*_i)_+ : \forall \quad i \in \{1, \ldots . T\}, \tag{5.50}$$

In the next chapter, we will numerically evaluate the methods that are derived in this chapter.

6

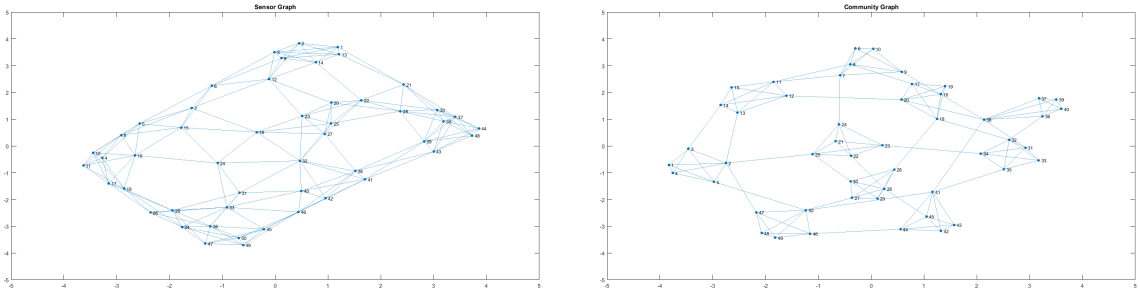
Numerical Validation

For the numerical evaluation of our relaxed problem, we will now provide numerical experiments. Before we do that, we will first explain the graph models that are used along with the values that are ascribed for the parameters. Lastly, an explanation will be given for the ROC-curves together with the average error of wrongly indexed nodes.

6.1. Graph Models

We consider a bistochastic symmetric adjacency matrix $\mathbf{\Phi}$ for our matrix that defines the flow of information. Furthermore, the elements in this matrix are between 0 and 1. In a bistochastic matrix, all the columns and rows sum to 1. It is assumed that the network is a distributed averaging network, i.e., the signal values convert to an average. In [33], a proof is given for the convergence of such a matrix towards an average.

The matrix is generated as $\mathbf{\Phi} = I - \frac{1}{2N}\boldsymbol{S} + \frac{1}{2N}(\boldsymbol{P} + \boldsymbol{P}^T)$ where $\boldsymbol{P}$ is a matrix that has random values for entries i, j if and only if the nodes i and j share an edge, i.e. $\boldsymbol{A}_{i,j} = 1$, otherwise $\boldsymbol{P}_{i,j} = 0$. Furthermore, the matrix $\boldsymbol{S}$ is considered to be a diagonal matrix consisting of the column sum of $\boldsymbol{P} + \boldsymbol{P}^T$ and N is defined as the number of nodes in the graph, i.e., $N = |\mathcal{V}|$. The matrix $\boldsymbol{A}$ is derived from a sensor graph for attack a and for attack b a community graph is considered. These graphs are generated with the help of the toolbox in [34].



(a) a sensor graph (b) a community graph

Figure 6.1: An example of a sensor graph and community graph

The initial assigned values to the nodes are integers equal to the number of the node. For every node there is assigned an initial integer value that is equal to its node number. Node 1 has a value of 1, node 2 has a value of 2 etc. The community graph is generated in such a way that there are exactly 10 communities consisting of 5 nodes. Every community has a maximum cut equal to 5. The purpose of this is to make every community a possible candidate for the compromised set. For all three problems we assume that the observation is done for 5 timeframes, hence $T = 4$. In table 6.1 there is an overview given for the constraint values and other attributions.

Table 6.1: Constraint values and number of nodes

	attack a	attack b
Timeframe observations T+1	5	5
Number of nodes N	50	50
Chosen cardinality for $\mathbf{1}_{\mathcal{A}}$	5	5
Cut constraint ρ	5	-
Dependent edge set constraint ρ	-	1
Cardinality constraint β	5	5
Density constraint α	-	-

The density constraint would be interesting to observe for very large graph networks, i.e., 1000 nodes. However putting a density constraint for a network of 50 nodes will make the subset of feasible node combinations very small. Hence it is decided not to include it during the performance evaluation.

6.2. Value Attack Models

Before we determine the outlying values, $\tilde{\mathcal{A}}$ must be determined first. The set $\tilde{\mathcal{A}}$ is selected such that the constraints in section 4.3 are met. Hereafter, we can now calculate the outlying values.
For M.1, there is a standard value of $c = 1$ for the outliers, i.e.,

$$\boldsymbol{o} = c\mathbf{1}_{\tilde{\mathcal{A}}} \tag{6.1}$$

For M.2 & M.3 we generate the outliers similar as in (3.18). The proposed attack scheme generates the outlying values depending on the previous observed data values at time i as

$$\boldsymbol{o}_{i+1} = \big((g(i)-1)\boldsymbol{\Phi}_{i+1}\boldsymbol{x}_i + \boldsymbol{\alpha}_0(1-g(i))\big) \odot \mathbf{1}_{\mathcal{A}_i} : \forall \quad i \in \{1, \ldots T\}. \tag{6.2}$$

where $g(i) \in [0,1]$ is an over-time decreasing function. Also, $\boldsymbol{\alpha}_0$ is the desired state for the malicious agents that is now constant for all elements. The values for state $t+1$ become then

$$\boldsymbol{x}_{i+1} = \boldsymbol{\Phi}_{i+1}\boldsymbol{x}_i + \big((g(i)-1)\boldsymbol{\Phi}_{i+1}\boldsymbol{x}_i + \boldsymbol{\alpha}_0(1-g(i))\big) \odot \mathbf{1}_{\mathcal{A}_i} : \forall \quad i \in \{1, \ldots T\}. \tag{6.3}$$

Since it is not possible to predict the outliers at time t for timestamps $t+2$ or higher, we will make an assumption that the malicious agents determine the outlying values based on the first observed state value $\boldsymbol{x}_t$. In other words, the agents calculate the outlying values given as in (6.2) by assuming that there is not any noise added during state transition.

$$\boldsymbol{x}_{t+1} = \boldsymbol{\Phi}_{t+1}\boldsymbol{x}_t + \boldsymbol{o}_{t+1}. \tag{6.4}$$

This is purely done so that the outlier-to-noise ratio for a given observed timeframe set can be calculated in case the noise variance is constant and known. For evaluation purposes it is useful to analyze the performance for different outlier-to-noise ratio values. Furthermore, for M.2 & M.3 we assume that we add a value of 1 to the average for the desired state. This means that

$$\boldsymbol{\alpha}_0 = \alpha_0 \mathbf{1}_N \tag{6.5}$$

and

$$\alpha_0 = \Big(\frac{1}{50}\sum_{N=1}^{50} N\Big) + 1 = 26.5. \tag{6.6}$$

Also, $g(0) = 0.9$ and $g(i) = g(i-1) * 0.8 : \forall i \in 1, \ldots T$. The attack scheme is applied for both the attacks described in 3.4.

6.3. Evaluation

For all three models and both attack modes, the optimization problems are simulated for different values of outlier-to-noise ratio (ONR). The ONR is calculated as:

$$ONR = \frac{\|\boldsymbol{o}\|_2^2}{NT\sigma^2}. \tag{6.7}$$

For every ONR, we perform 1000 different noise simulations. We must take into account that for every noise realisation the graph structure and the outlying values $\boldsymbol{o}$ must be identical for a fair evaluation of the threshold function $T(\boldsymbol{z})$. Since we want to look at how the threshold function is behaving for a varying ONR, we need to be sure that the other elements are not changing throughout the realisations. In other words, the variation in our data $\boldsymbol{z}$ must come only from the noise element $\boldsymbol{n}$. Otherwise, the variation of our estimated $\mathbf{1}_{\hat{\mathcal{A}}}$ and consequently $T(\boldsymbol{z})$ will be dependent on different factors. In addition to this consideration, the graph structure and the distribution of the outlier energy $\boldsymbol{o}^T\boldsymbol{o}$ may have an effect on the outcome. This is due to the fact that the estimation of $\mathbf{1}_{\hat{\mathcal{A}}}$ is dependent on these factors. Hence, by keeping the graph structure and outliers constant for all 1000 noise realisations we can analyze the influence of σ^2 on the performance. An overview of the process is given in figure 6.2.

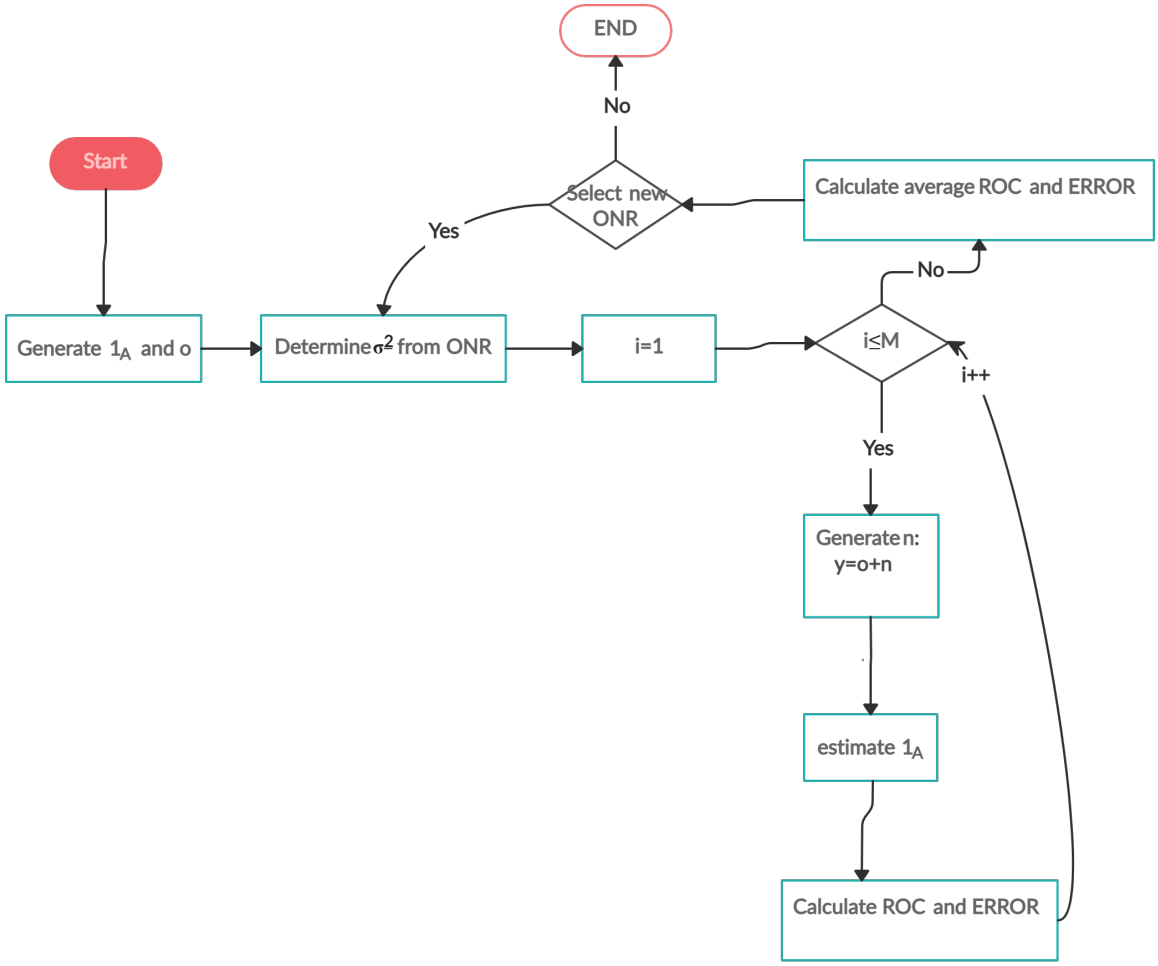


Figure 6.2: In this flowchart it is shown how results are generated without changing the network properties or the outlying values. Hence only σ^2 is adjusted to the given ONR. M is denoted as the number of noise realisations per ONR. We will also look into the error related to the identification of malicious nodes.

Apart from the theoretical ROC-curve that can be derived from the detection and false alarm probability. We can also derive an empirical ROC-curve for the experiments. The empirical ROC-curve is created by simulating values for the threshold function under both hypotheses and equal σ^2. Then, the detection and false alarm probabilities are calculated empirically. The threshold value γ' for a given empirical P_{fa} is calculated by calculating i out of $P_{fa} = \frac{i}{N}$. N is the total number of noise realisations per hypothesis. After this, γ' is equal to the average of the i'th and $i-1$'th largest value for the realisations of $\mathcal{H}_0$. The corresponding P_d is then equal to the number of noise realisations that are bigger then γ' normalized by N. For a visualisation of these results we can look at figure 6.3.

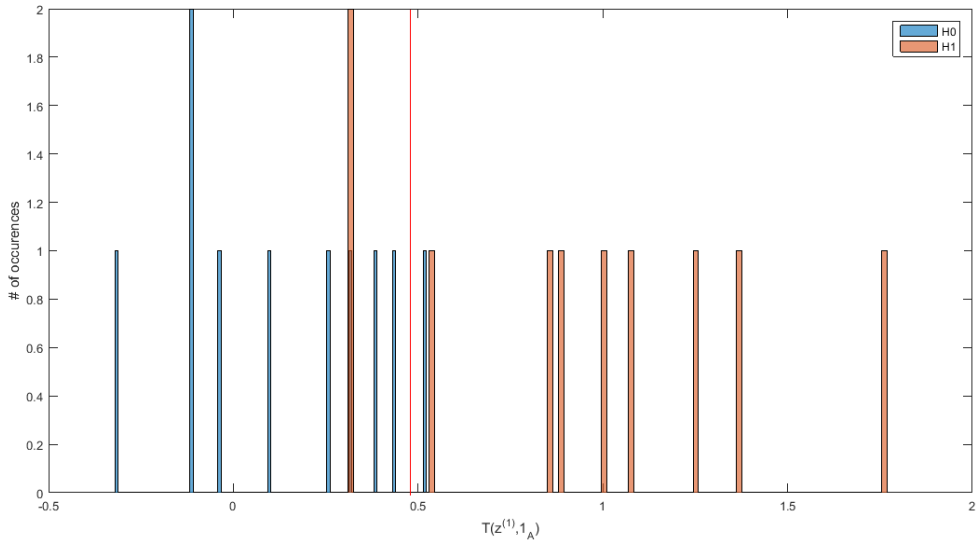


Figure 6.3: In this example, the number of realisations is taken as 10. For a P_{fa} of $\frac{1}{10}$, we can observe that the P_d is equal to $\frac{8}{10}$. The red line corresponds to γ' for $P_{fa} = \frac{1}{10}$.

The average of the theoretical ROC-curve is calculated by taking the average of the detection probability P_d for a given P_{fa} for all 1000 noise realisations:

$$\overline{P_d} = \frac{1}{1000} \sum_{i=1}^{1000} P_d^i. \tag{6.8}$$

We will refer to this average theoretical ROC-curve for $P_d^i : \forall i \in \{1, 2, ...1000\}$ as $\overline{ROC}$. Apart from the ROC-curve, an important factor for evaluating the performance is also the identification of the compromised nodes. The average number of wrongly indexed nodes **per timeframe** is calculated as follows:

$$\overline{Error} = \frac{1}{1000(T-1)} \sum_{i=1}^{1000} \left\| \mathbf{1}_{\tilde{\mathcal{A}}} - \mathbf{1}_{\tilde{\mathcal{A}}}^{i*} \right\|_2^2, \tag{6.9}$$

where $\mathbf{1}_{\tilde{\mathcal{A}}}^{i*}$ is the final estimation for the i'th noise realisation and $\mathbf{1}_{\tilde{\mathcal{A}}}$ refers to the true value. The equation calculates the number of wrongly indexed nodes per noise realisation per timeframe.

6.4. Results

We will now show the $\overline{ROC}$ and empirical ROC-curves along with the $\overline{Error}$. The results are derived for the following ONR values: 0.01, 0.1, 0.2 and 0.5. For ONR values higher than 0.5 a full saturation of the $\overline{ROC}$-curves are observed for all models along with almost all the empirical ROC-curve. This means that the P_d is almost directly increased to 1 at a very low P_{fa}. It is therefore interesting to analyze the empirical ROC and theoretical $\overline{ROC}$-curves for ONR values lower then 0.5, since the behavior is predictable for values higher than 0.5, i.e., a full saturation of both empirical ROC- and theoretical $\overline{ROC}$-curves. The P_d for all three models are in a way dependent on the ratio of the estimated outlier energy and the noise energy. An increasing ONR will result in an estimated outlier energy that is closer to the true outlier energy while the noise energy decreases. This means that the P_d will only increase for a given P_{fa} and increasing ONR.

6.4.1. Time-invariant attacks

We will now provide the empirical ROC and theoretical $\overline{ROC}$ for M.1.

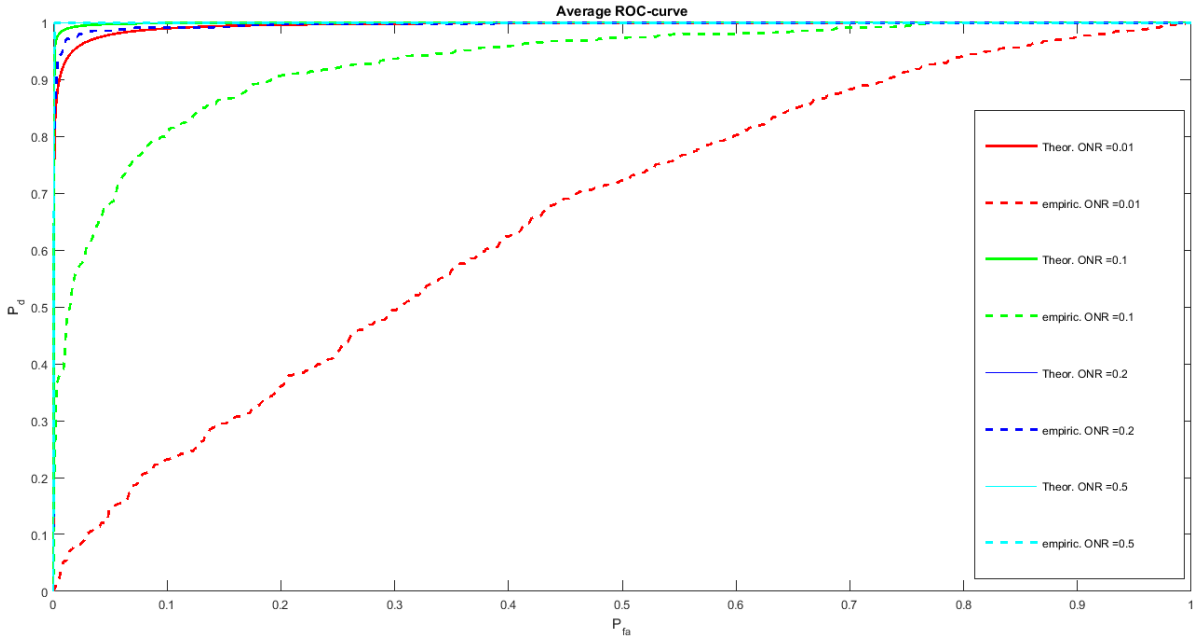


Figure 6.4: Empirical ROC and Theoretical $\overline{ROC}$-curve for M1.a

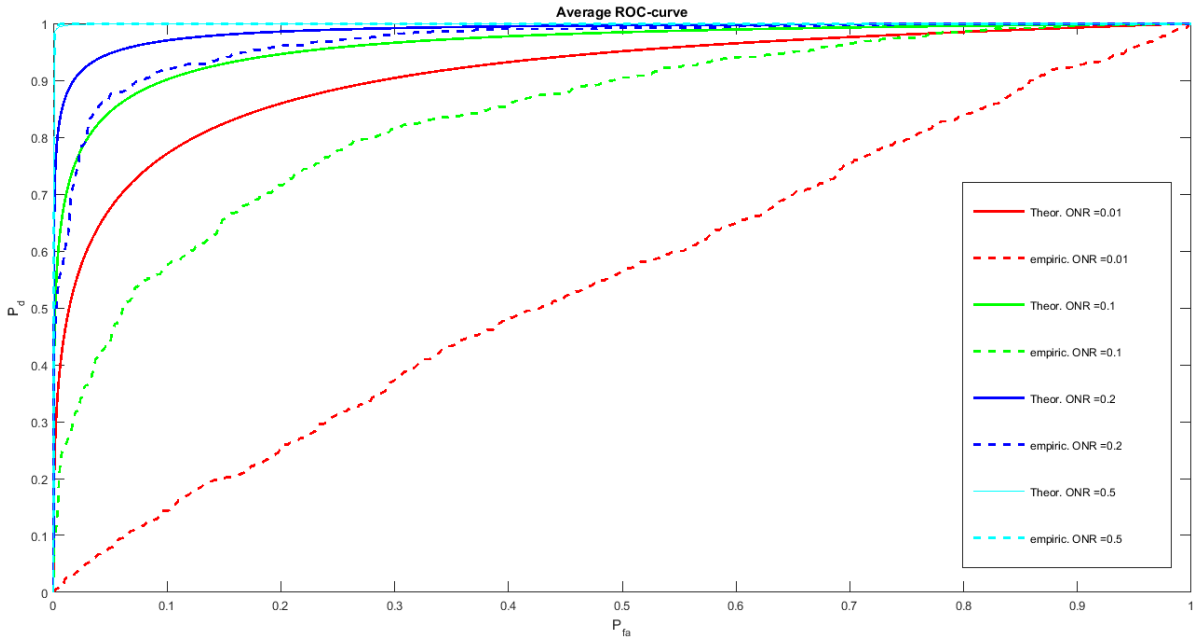


Figure 6.5: Empirical ROC and Theoretical $\overline{ROC}$-curve for M1.b

Attack \ ONR	0.01	0.1	0.2	0.5
a	8.43	4.18	2.07	0.21
b	5.61	4.37	3.81	1.51

Table 6.2: $\overline{Error}$ for M.1

The $\overline{ROC}$-curve for M.1 is calculated as follows:

$$\overline{P_d}(P_{fa}) = \frac{1}{1000}\sum_{i=1}^{1000} Q(Q^{-1}(P_{fa}/2) - \frac{A_i}{\sqrt{\sigma^2}}) + Q(Q^{-1}(P_{fa}/2) + \frac{A_i}{\sqrt{\sigma^2}}) \tag{6.10}$$

where A_i is defined as in section 4.3.

$$A_i = \frac{\mathbf{1}_{\tilde{\mathcal{A}}}^{i*T}\boldsymbol{z}}{\sqrt{T|\mathcal{A}_{i*}|}} \tag{6.11}$$

and $\mathcal{A}_{i*}$ is denoted as the estimated set for the i'th realisation $\forall\, i \in \{1, 2, ..1000\}$.

The first noticeable observation is the decreasing similarity of the empirical and theoretical $\overline{ROC}$-curves for a decreasing ONR. This can be directly attributed to the estimation of the support vectors $\mathbf{1}_{\tilde{\mathcal{A}}}$. Higher ONR yields a lower $\overline{Error}$. Consequently, a lower $\overline{Error}$ gives a more stable and less varying values for $\mathbf{1}_{\tilde{\mathcal{A}}}$. Which in turn results in an increasing resemblance of a normal distribution mentioned in (4.31) for $\mathcal{H}_1$ and for $\mathcal{H}_0$. An increase in the noise variance gives a higher deviation from the mean for elements in $\boldsymbol{z}$, under $\mathcal{H}_0$ Consequently, the optimization algorithm returns the elements with the highest deviation from zero as an estimation for $\mathbf{1}_{\tilde{\mathcal{A}}}$. In other words, the optimization problem returns values for $\mathbf{1}_{\tilde{\mathcal{A}}}$ that favors selection from the 'tail' of the normally distributed elements in $\boldsymbol{z}$, this can also be seen in figure 6.6. In chapter 4.3, we explained how the amplitude term A must deviate as much as possible from 0 in order to increase the P_d. This is achieved with the above mentioned selection from the 'tail'.

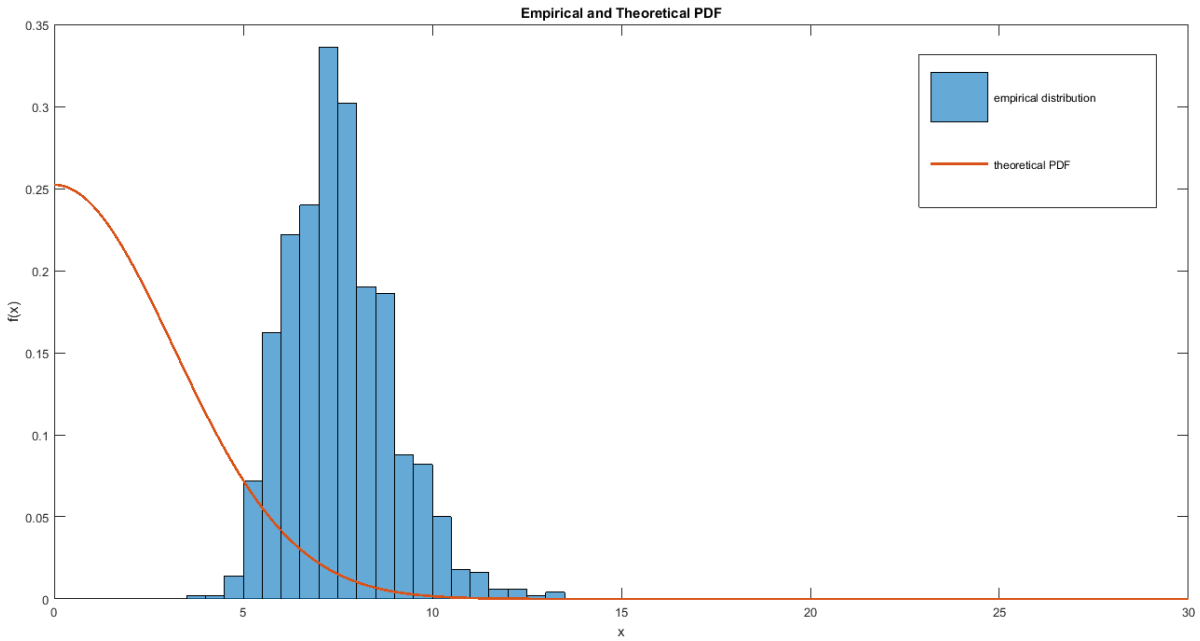


Figure 6.6: Comparison of the empirical distribution and the theoretical PDF under $\mathcal{H}_0$ for M.1a, $ONR = 0.1$

The empirical distribution in figure 6.6 is calculated by taking the histogram for the calculated $T(\boldsymbol{z})$ under $\mathcal{H}_0$ and normalizing the area to 1, the theoretical distribution under $\mathcal{H}_0$ is given in (4.31). It can be observed how the empirical distribution differs from the theoretical probability density function (PDF), under $\mathcal{H}_0$. Combining this with the fact that the decrease in ONR gives an increasing resemblance between the empirical distributions under $\mathcal{H}_0$ and $\mathcal{H}_1$, resulting in a larger overlap of the empirical distributions in comparison to the theoreticals. In figure 6.7, we observe how the empirical distributions are overlapping for the lowest given ONR value, while the area of the theoretical PDF's are less overlapping.

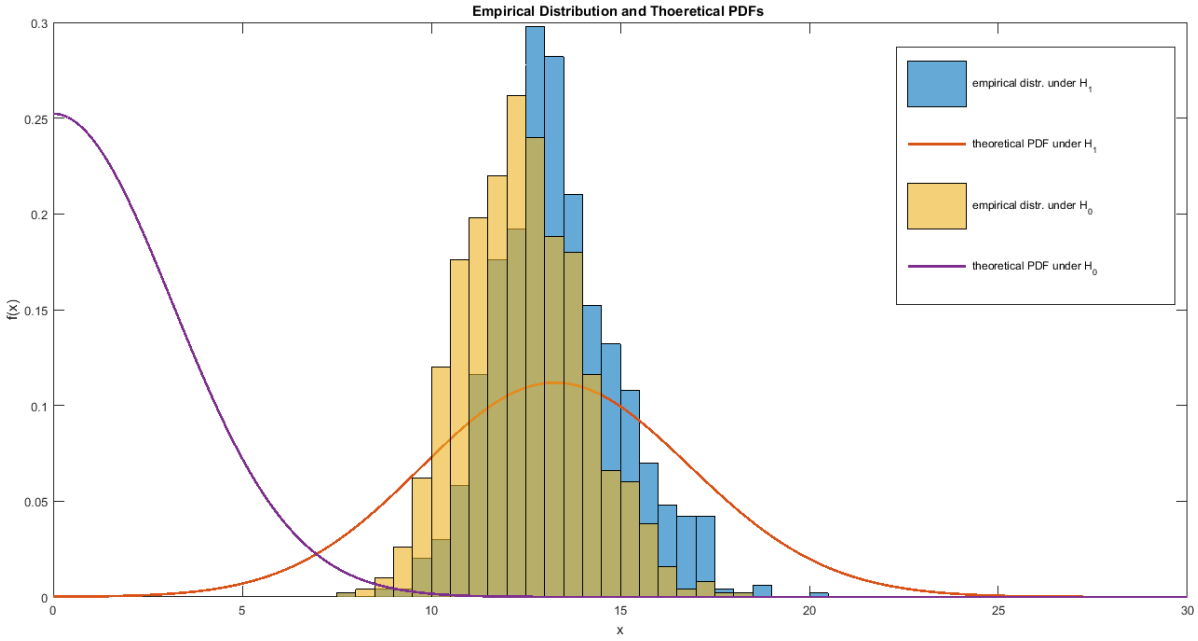


Figure 6.7: Empirical distribution and theoretical PDF's for M1.a, $ONR = 0.01$

Here, the average theoretical PDF ($\overline{PDF}$) is calculated as the average of the theoretical PDF's under $\mathcal{H}_1$. Specifically, the PDF of a folded Gaussian distribution $f_{|\mathcal{N}(\mu_i,\sigma^2)|}(x)$ with variance σ^2 and mean μ_i is given as [26]:

$$f_{|\mathcal{N}(\mu_i,\sigma^2)|}(x) = \frac{1}{\sqrt{\sigma^2 2\pi}} e^{-\frac{(x-\mu_i)^2}{2\sigma^2}} + \frac{1}{\sqrt{\sigma^2 2\pi}} e^{-\frac{(x+\mu_i)^2}{2\sigma^2}}, \tag{6.12}$$

where μ_i is in our case equal to:

$$\mu_i = \frac{\boldsymbol{z}^T \mathbf{1}_{\tilde{\mathcal{A}}}^{i*}}{\sqrt{(T)|\mathcal{A}_{i*}|}}. \tag{6.13}$$

The $\overline{PDF}$ is calculated as:

$$\overline{f}(x) = \frac{1}{1000} \sum_{1000}^{i=1} f_{|\mathcal{N}(\mu_i,\sigma^2)|}(x) \quad : \forall x \in [0, \infty). \tag{6.14}$$

The terms $\overline{PDF}$ and $\overline{f}(x)$ will be used interchangeably.
Since that the distribution under H_0 is not dependent on $\mathcal{A}_{i*}$, the domain of integration for a given P_{fa} is identical for all noise realisations $i \in \{1, 2, ..1000\}$. The domain $[a, \infty]$ can then be calculated as follows

$$P_{fa} = \int_a^\infty f_{|\mathcal{N}(0,\sigma^2)|}(x)(x)dx \tag{6.15}$$

Conequently, we can denote $\overline{P_d}$ also in terms of $\overline{f}(x)$ as follows

$$\overline{P_d}(P_{fa}) = \frac{1}{1000} \sum_{1000}^{i=1} \int_a^\infty f_{|\mathcal{N}(\mu_i,\sigma^2)|}(x)dx = \int_a^\infty \overline{f}(x)dx. \tag{6.16}$$

Hence, the $\overline{ROC}$-curve can also be analyzed with $\overline{f}(x)$ and $f_{|\mathcal{N}(0,\sigma^2)|}(x)$.

Considering these, we can observe that both the empirical and theoretical performance for M.1b is lower compared to M.1a. This is shown in figure 6.8.

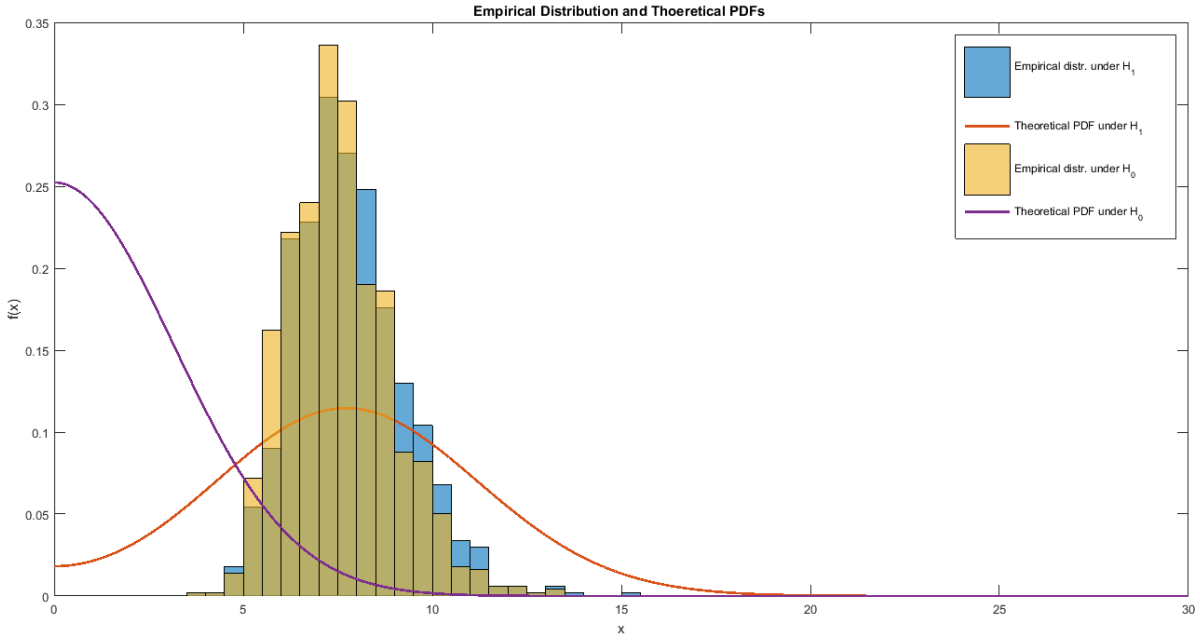


Figure 6.8: Empirical distribution and theoretical PDF's for M1.b, $ONR = 0.01$

We observe a more 'shifted' distributions under $\mathcal{H}_1$ to the left, which means that our gathered values for $T(\boldsymbol{z})$ are lower compared to those from M.1a. The main reason for lower values of $T(\boldsymbol{z})$ can be attributed to the cardinality of the estimated $\mathbf{1}_{\mathcal{A}}$:

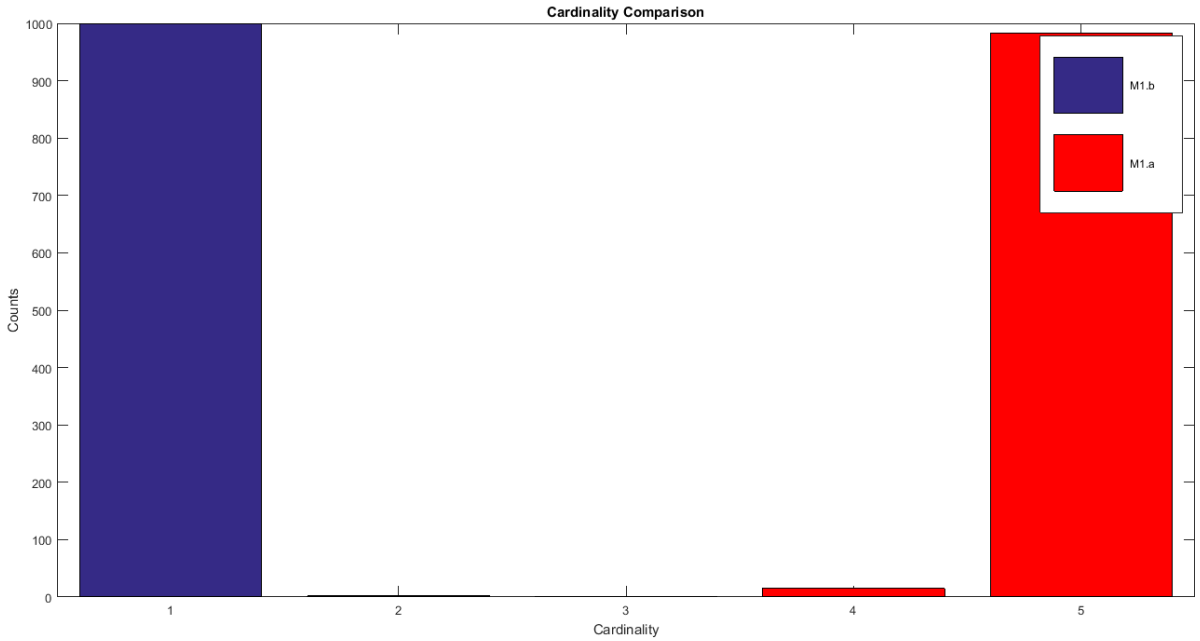


Figure 6.9: Cardinality histogram for M1.a and M1.b under $\mathcal{H}_1$, $ONR = 0.01$

Lower cardinality does not necessarily imply lower values for $T(\boldsymbol{z})$. However, for M1.b, we can observe that 'swinging' cardinality values,i.e., change of cardinality count between noise realisations, will give a non-unimodal empirical distribution and $\overline{PDF}$. For example, when we observe a bimodal distribution, see e.g. figure 6.10,

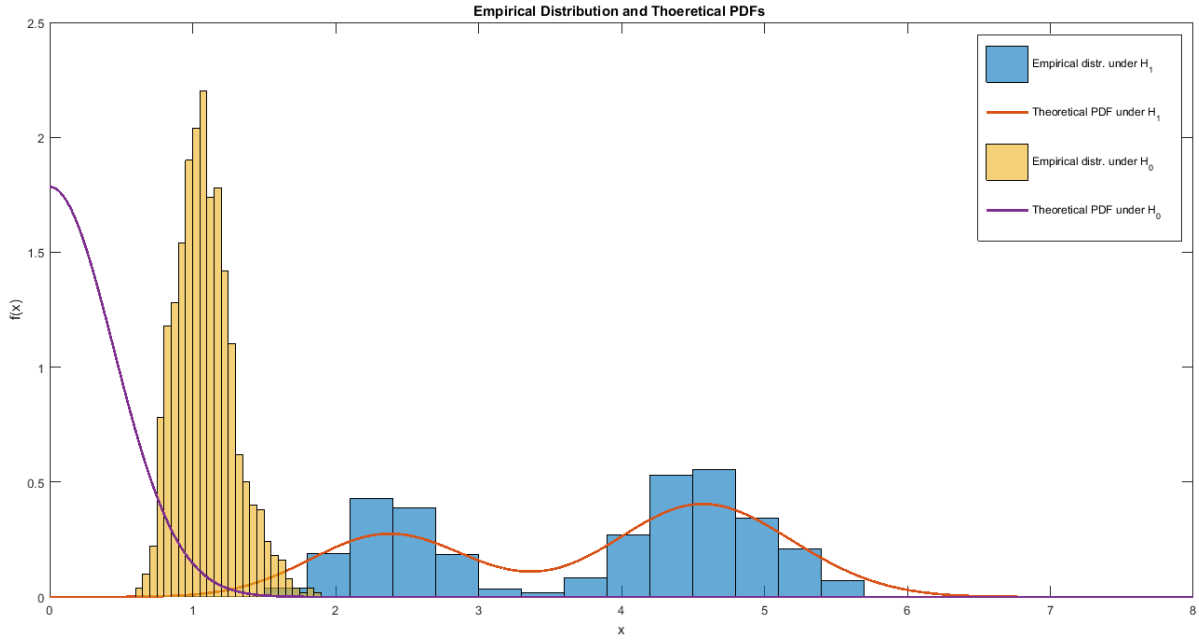


Figure 6.10: Empirical distribution and theoretical PDF's for M1.b, $ONR = 0.5$

we also observe how the distribution of the cardinality count is divided between 1 and 5 in figure 6.11.

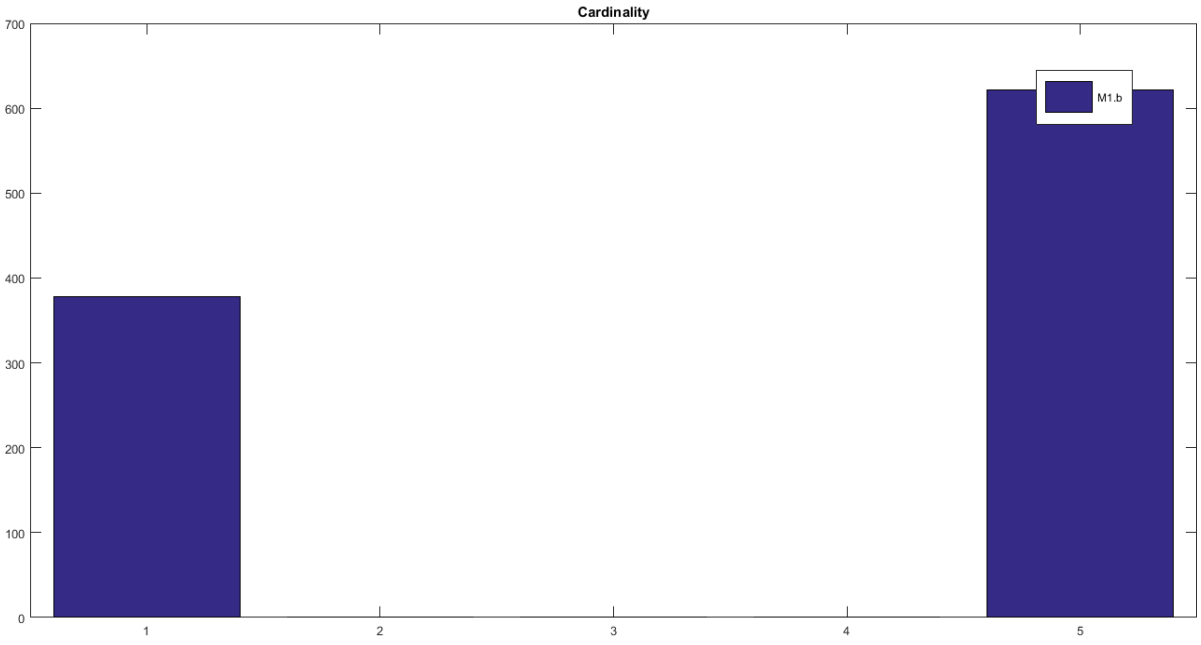


Figure 6.11: Cardinality histogram for M1.b under $\mathcal{H}_1$, $ONR = 0.5$

The feasible constraint set for attack b is smaller then for attack a. For attack b, the set consists of 10 communities, and a couple of single node sets. During the rounding process of the 'spread out values' $\boldsymbol{x}_c$, the chances of having a community set acknowledged within the first c Bernouille realisations is small, for the given ONR values. A community set of 10 is relatively small in comparison to the set of all possible combinations. Hence, rounding the higly 'spread out values' $\boldsymbol{x}_c$ to one of the 10 communities is not big. For lower ONR values, we can see that the chances are as good as none. Hence, this will result in 'selecting' a single node as the set of compromised nodes since that the chances are higher for a single element of 1 rather than five elements of 1 that form a community. This in turn yields a lower $T(\boldsymbol{z})$.

When we compare the results with the results of the Lasso Estimator:

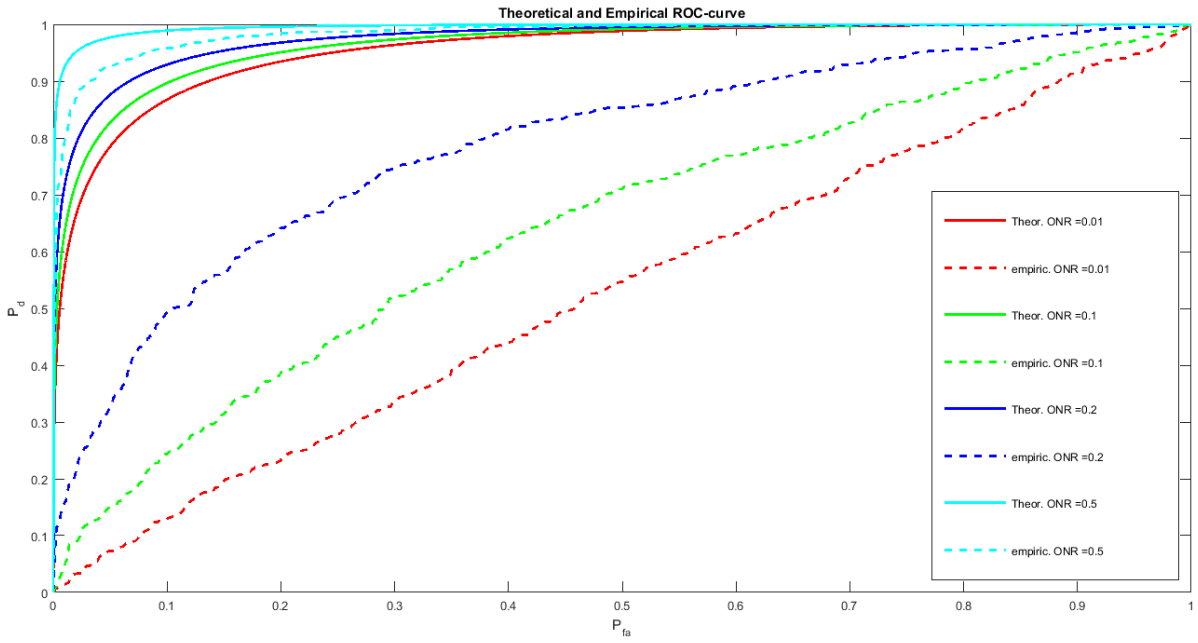


Figure 6.12: ROC-curves based on the Lasso-method for M1.a

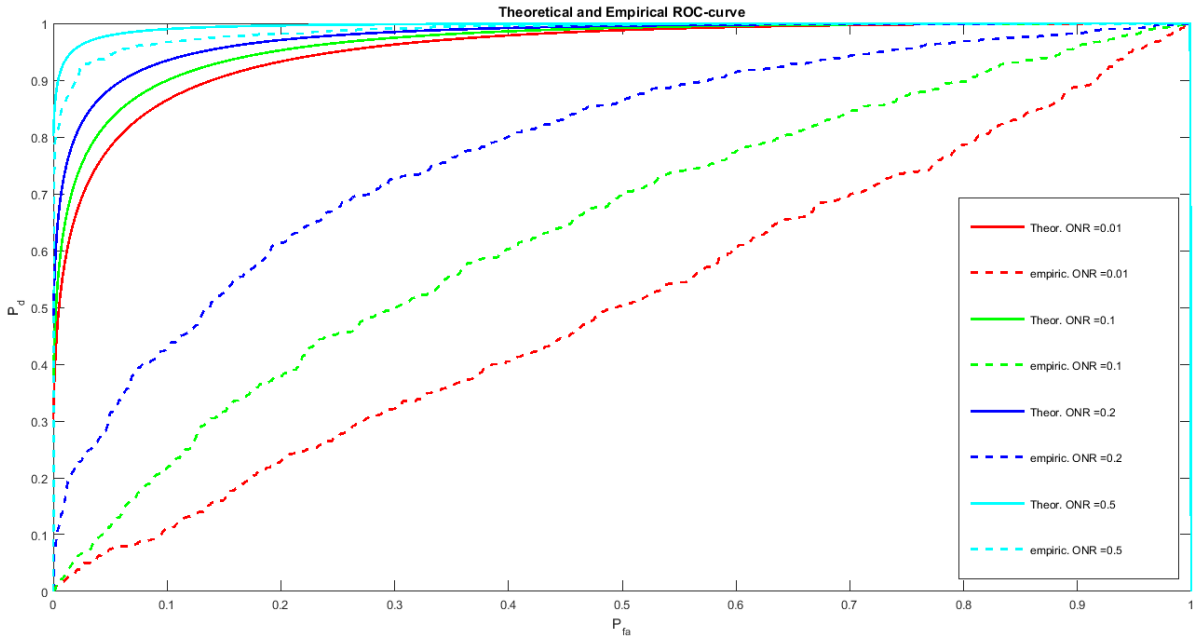


Figure 6.13: ROC-curves based on the Lasso-method for M1.b

Attack \ ONR	0.01	0.1	0.2	0.5
a	8.82	7.62	6.53	4.02
b	8.82	7.58	6.48	4.03

Table 6.3: $\overline{Error}$ for M.1 for the Lasso-method

For the Lasso-based method, we observe also a gap between the empirical ROC- and $\overline{ROC}$-curves. The reasoning is similar as above. A higher ONR results in values for the estimated energy $\boldsymbol{o}^T\boldsymbol{o}$ that are more stable and closer to the true value. In turn the empirical distributions will resemble more that of a normal distribution denoted as in (4.53).
For lower ONR values, since that the algorithm will select the nodes with the β highest magnitude values, this will not give an empirical distribution that resembles a zero-mean Gaussian distribution for $T(\boldsymbol{z})$. Instead, the calculated values for $T(z)$ will then be optimized due to how $\mathbf{1}_{\hat{\mathcal{A}}}$ is calculated. Selecting $\mathbf{1}_{\hat{\mathcal{A}}}$ that optimizes $f_{P_{dlasso}}(\boldsymbol{\lambda})$ will also optimize $T(\boldsymbol{z}) = \boldsymbol{z}^T\boldsymbol{o}$. Hence we will get a distribution for $T(z)$ that will select nodes from the 'right-tail' of a zero-mean Gaussian distribution, thereby overlapping with the empirical distribution under $\mathcal{H}_1$ due to the higher variance.
Also, this time we know that only the nodes with the highest magnitudes are selected for calculating $T(\boldsymbol{z})$. Combining this with the fact that the Lasso-based method does only take the cardinality constraint into account will result in selecting nodes with β highest magnitudes, this will yield a higher $\overline{Error}$.

6.4.2. Time-variant attacks

Fixed anomalous nodes

For M.2, the $\overline{ROC}$-curves are given below

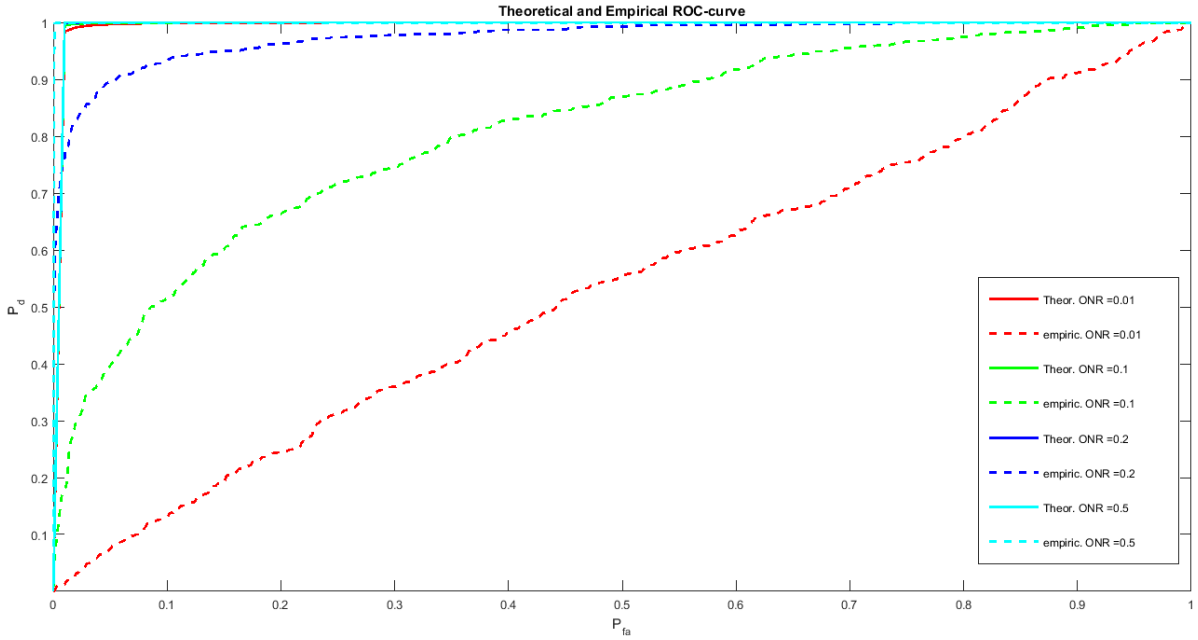


Figure 6.14: Theoretical and empirical $\overline{ROC}$-curve for M2.a

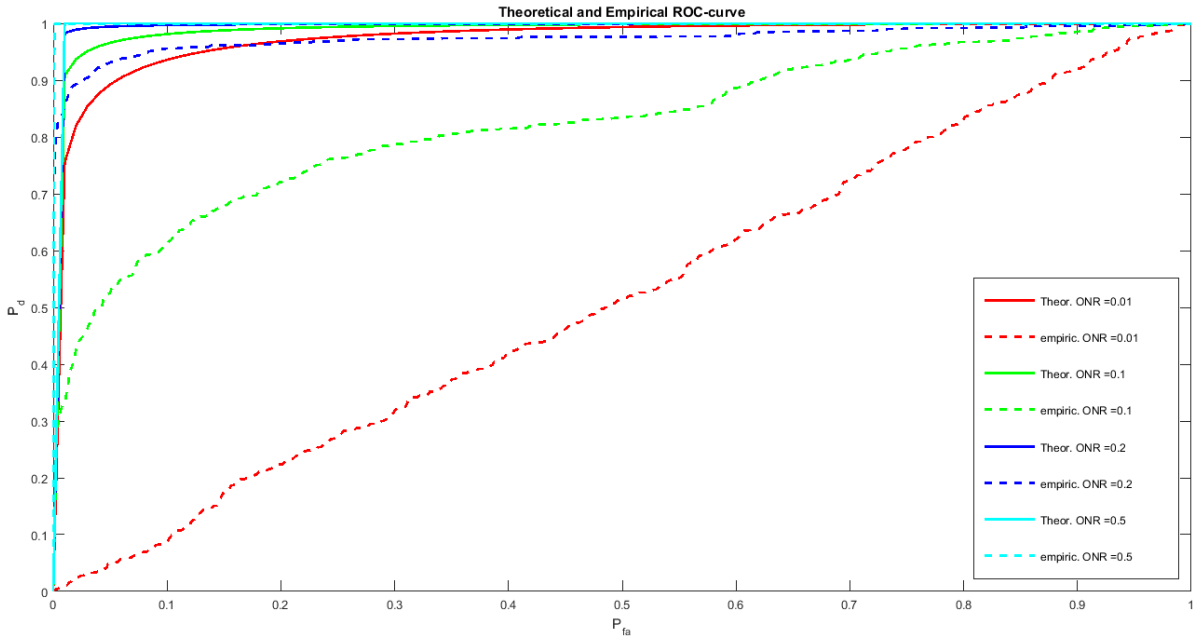


Figure 6.15: Theoretical and empirical $\overline{ROC}$-curve for M2.b

The $\overline{ROC}$-curve for M.2 and M.3 is calculated as follows:

$$\overline{P_d}(P_{fa}) = \frac{1}{1000}\sum_{i=1}^{1000} Q(Q^{-1}(P_{fa})_{\chi^2_{|\mathcal{A}_{i*}|}})_{\chi^2_{|\mathcal{A}_{i*}|}}(\mu_i), \tag{6.17}$$

where μ_i is defined as:

$$\mu_i = \frac{\boldsymbol{z}^T diag(\mathbf{1}_{\tilde{\mathcal{A}}}^{i*})\boldsymbol{z}}{\sigma^2} \tag{6.18}$$

The equation above is calculated for the gathered values $\mathbf{1}_{\tilde{\mathcal{A}}}^{i*}$ under $\mathcal{H}_1$ for all $i \in \{1, 2, ..1000\}$.

Attack \ ONR	0.01	0.1	0.2	0.5
a	8.83	6.68	5.55	3.02
b	7.2460	2.3640	0.2800	0

Table 6.4: $\overline{Error}$ for M.2

For M.2, once again a gap can be observed between the empirical ROC and theoretical $\overline{ROC}$-curves. Also, the steepness of the theoretical $\overline{ROC}$-curves remain very high even for low ONR values. Particularly for attack a, we observe an almost 'perfect saturation' for ONR=0.01. This is mainly due to the high non-centrality parameter μ. In figure 6.16 it can be observed that for a *dof*=20 a small overlapping area is observed between a distribution with $\mu = 50$ and a centralized X^2 distribution. The average of μ for the lowest ONR for M2.a is also around 50.

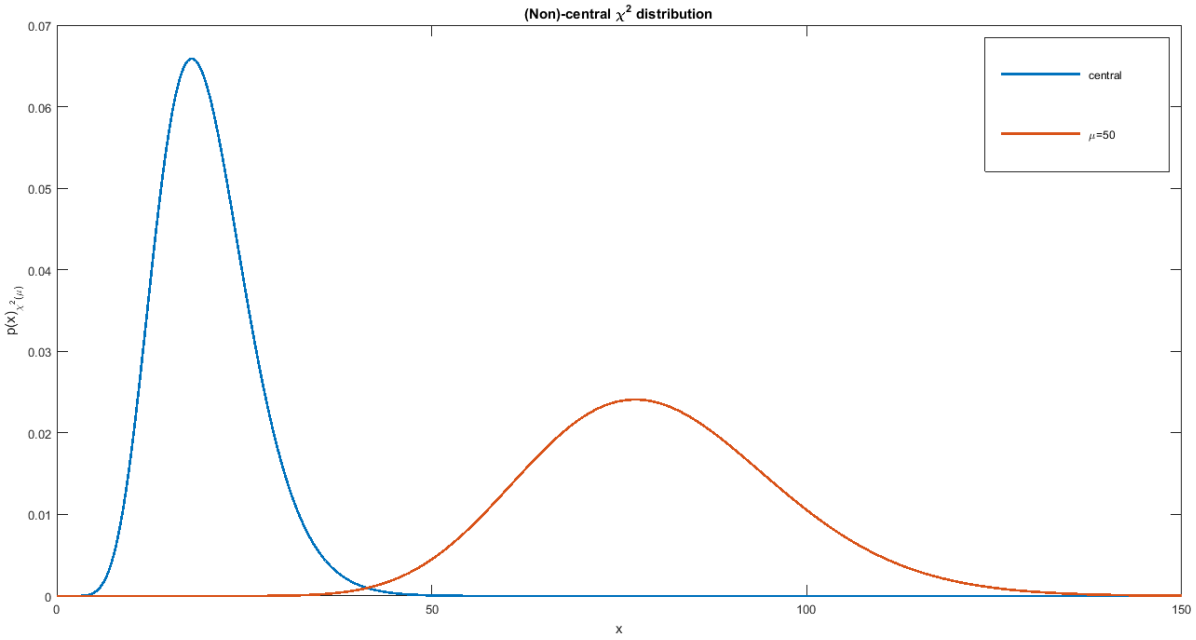


Figure 6.16: PDF of the central and non-central X^2 distributions with degrees of freedom equal to 20

If we want to analyze the relationship between the theoretical distributions and the $\overline{ROC}$-curve, it is better to show the average distributions for all possible cardinality cases apart, i.e., $|\mathcal{A}| = 1, 2, 3 \ldots \beta$. In other words, for every possible cardinality a different average distribution $\overline{f}(x)_{|\mathcal{A}|}$ is calculated.

$$\overline{f}(x)_{|\mathcal{A}|} = \frac{1}{|\mathcal{O}_{|\mathcal{A}|}|} \sum_{i \in \mathcal{O}_{|\mathcal{A}|}} f_{\mathrm{X}^2_{|\mathcal{A}_{i*}|}(\mu_i)}(x) : \forall \quad |\mathcal{A}| \in \{1, 2, \ldots \beta\} \tag{6.19}$$

Where the set $\mathcal{O}_{|\mathcal{A}|}$ is defined as follows

$$i \in \mathcal{O}_{|\mathcal{A}|} \quad if \quad |\mathcal{A}_{i*}| = |\mathcal{A}| : \forall \quad i \in \{1, 2, ..1000\} \wedge |\mathcal{A}| \in \{1, 2, \ldots \beta\} \tag{6.20}$$

This is necessary since that $\overline{P_d}(P_{fa})$ can not be calculated directly with the total average distribution under a single integration, as is the case for $\mathrm{M}.1$, i.e.,

$$\overline{P_d}(P_{fa}) = \int_a^\infty \overline{f}(x) dx \tag{6.21}$$

Every average distribution must now be integrated in a different domain $[a, \infty]$ for a given P_{fa}.
For every average distribution we can now calculate $\overline{P_{d_{|(A)|}}}(P_{fa})$ for a given P_{fa}.

$$\overline{P_{d_{|(A)|}}}(P_{fa}) = \int_a^\infty \overline{f}(x)_{|\mathcal{A}|} dx \quad : \forall \quad |\mathcal{A}| \in \{1, 2, \ldots \beta\} \tag{6.22}$$

Where a is calculated from the integer of the central X^2-distribution as

$$P_{fa} = \int_a^\infty f_{\mathrm{X}^2_{|(T-1)_{\mathcal{A}}|}}(x) dx \tag{6.23}$$

Depending on the occurrence of the cardinality, the average $\overline{P_d}$ is then equal to the weighted average of the $\overline{P_{d_{|\mathcal{A}|}}}$ of every average distribution.

$$\overline{P_d}(P_{fa}) = \sum_{|(A)|=1}^{\beta} \frac{|\mathcal{O}_{|\mathcal{A}|}|}{1000} P_{d_{|(A)|}}(P_{fa}) \tag{6.24}$$

For example, if there are 800 occurrences of $|\mathcal{A}| = 1$ and 200 occurrences of $|\mathcal{A}| = 5$, then $\overline{P_d}$ is calculated as

$$\overline{P_d}(P_{fa}) = 0.8 * \overline{P_{d_1}}(P_{fa}) + 0.2 * \overline{P_{d_5}}(P_{fa}) \tag{6.25}$$

Contrary to $\mathrm{M}.2\mathrm{a}$, the 'perfect saturation' is not observed for $\mathrm{M}.2\mathrm{b}$. For example, If we observe the empirical distribution and the average distributions for $\mathrm{M}.2\mathrm{b}, \mathrm{ONR} = 0.2$, we observe in figure 6.17, similar as in $\mathrm{M}.1\mathrm{b}$, a bi-modal distribution due to a 'swinging' cardinality count:

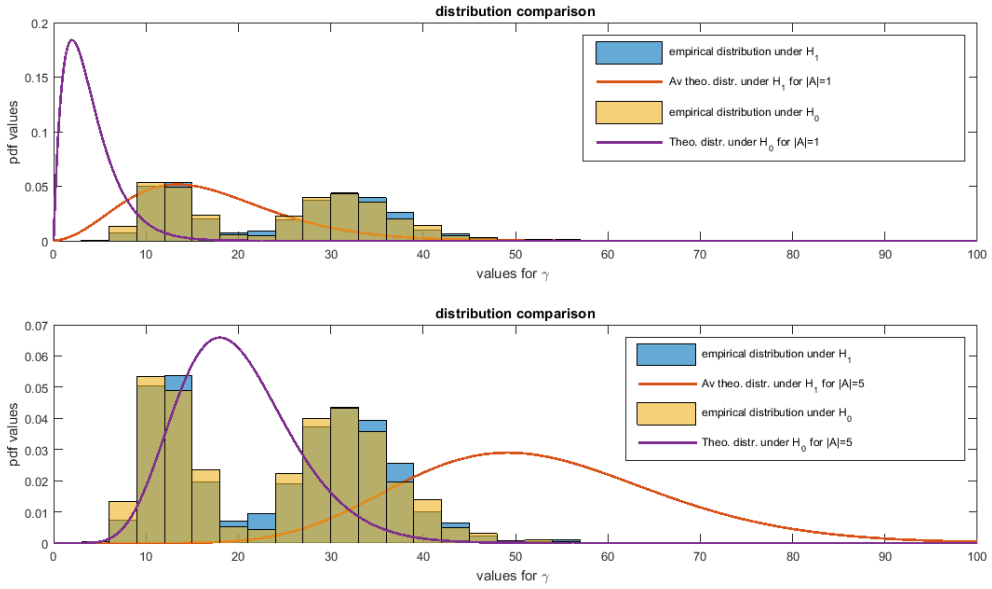


Figure 6.17: Empirical distribution and theoretical PDF's for M2.b, $ONR = 0.01$

It is explained in chapter 4 that lower degrees of freedom can actually increase the performance. A decrease in degrees of freedom is also associated with a lower non-centrality parameter which in turn will result in a higher overlap between the distributions for $\mathcal{H}_1$ and $\mathcal{H}_0$. Hence the peaks around the lower values for x, can be subscribed to the values of $T(\boldsymbol{z})$ where the cardinality of $\mathbf{1}_{\tilde{\mathcal{A}}}$ is equal to 1. The occurrence of a low cardinality estimation has a similar explanation like for the case of M.1b. An increasing noise variance also gives an increase in the scattering of the estimated values for $\boldsymbol{x}$. Hence it is not guaranteed that the algorithm will always acknowledge a community set within the first c Bernouille realisations. With scattering it is meant that the values of $\boldsymbol{x}$ will resemble less of a binary value, the $'1'$ elements are 'spread out' over several nodes due to the binary relaxation.
For higher ONR, we observe that the bimodality disappears as in figure 6.18

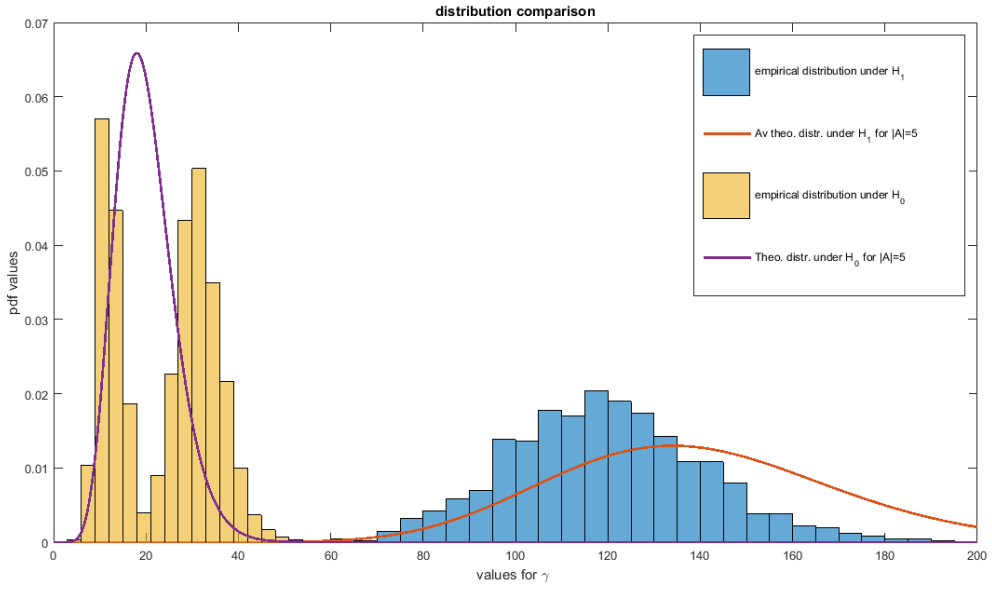


Figure 6.18: Empirical distribution and theoretical PDF's for M2.b, $ONR = 0.5$

Apart from the rounding related issues, a non-optimal cardinality can also be associated with a higher detection peformance. It is explained in 4 that the optimal P_d is not always derived with only maximizing μ. It possible that lower μ values can give a higher detection probability if the lower μ values are combined with lower cardinality. For M.2a, there is also a variation in cardinality observed for the highest ONR. It can be shown for M.2a that the variation stems from the optimal P_d for lower cardinalities. The structure of the sensor graph that is used for attack a contains a larger set of node combinations that is feasible and with a cardinality equal to 5. Hence the chances are higher that the first c Bernouille realisations acknowledge a feasible maximum cardinality subset of nodes. Interestingly, we can still observe for $ONR = 0.5$ a varying cardinality value. We will now argue that this is due to the case that the lower cardinality will yield a higher detection probability.

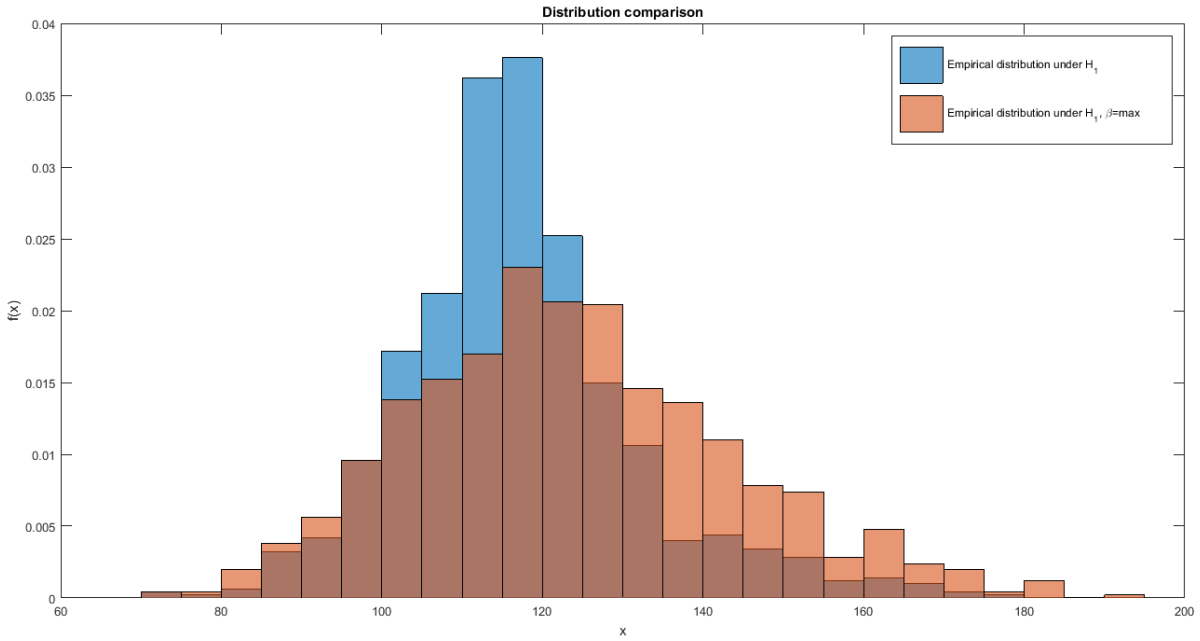


Figure 6.19: The empirical distribution for when the complete algorithm is run and when the algorithm is run only for β_{max}: M.2a, $ONR = 0.5$, $\mathcal{H}_1$

In figure 6.19 we see now 2 histograms. The blue histogram represents the true empirical distribution, the red histogram shows the values that we will get if we would run the algorithm in 2 for only β_{max}. The latter giving us cardinality values of almost only 5. Although we have a varying cardinality for the true empirical distribution, it can still be observed that there is not a bi-modal distribution similar as in M.2b:

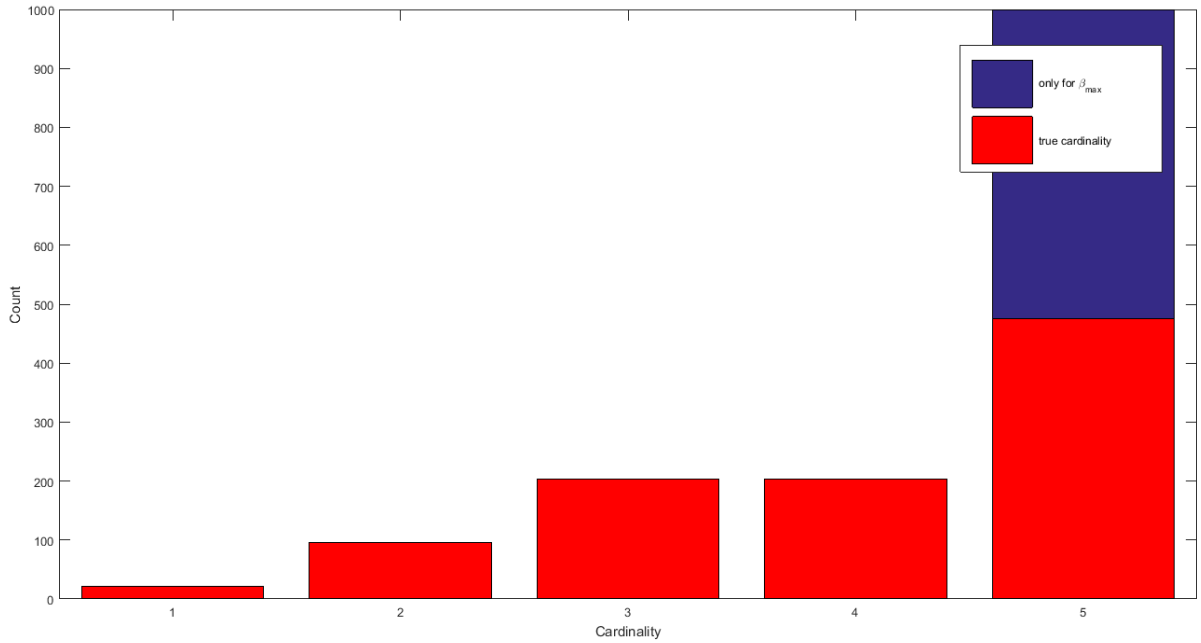


Figure 6.20: Cardinality counts for M.2a, $ONR = 0.5$ under $\mathcal{H}_0$

Looking at figure 6.19, we can conclude that there are estimated node sets with cardinality lower than 5 that returns a non-optimal value for μ. But these non-optimal μ values do not differ much from the optimal μ values that we would get for a maximal cardinality. The difference is small enough to yield a higher P_d for node sets with a smaller cardinality. With the help of figures 6.19 and 6.20, it can be clearly seen that the variation in the cardinality stems from choosing the optimal detection probability, contrary to M.2b.

A second note must be made on the distributions under $\mathcal{H}_0$. Looking at the equation in (4.42), theoretically, they should be independent of σ^2, since a centralized χ^2 distribution is the summation of zero mean and unit variance squared Gaussian distributions. Thus, indifferent of the noise variance value σ^2, the algorithm in 2 always normalizes the variances for the selection of the nodes under $\mathcal{H}_0$ that have mean values of 0. Consequently, the empirical distributions under H_0 should also be independent of the ONR values

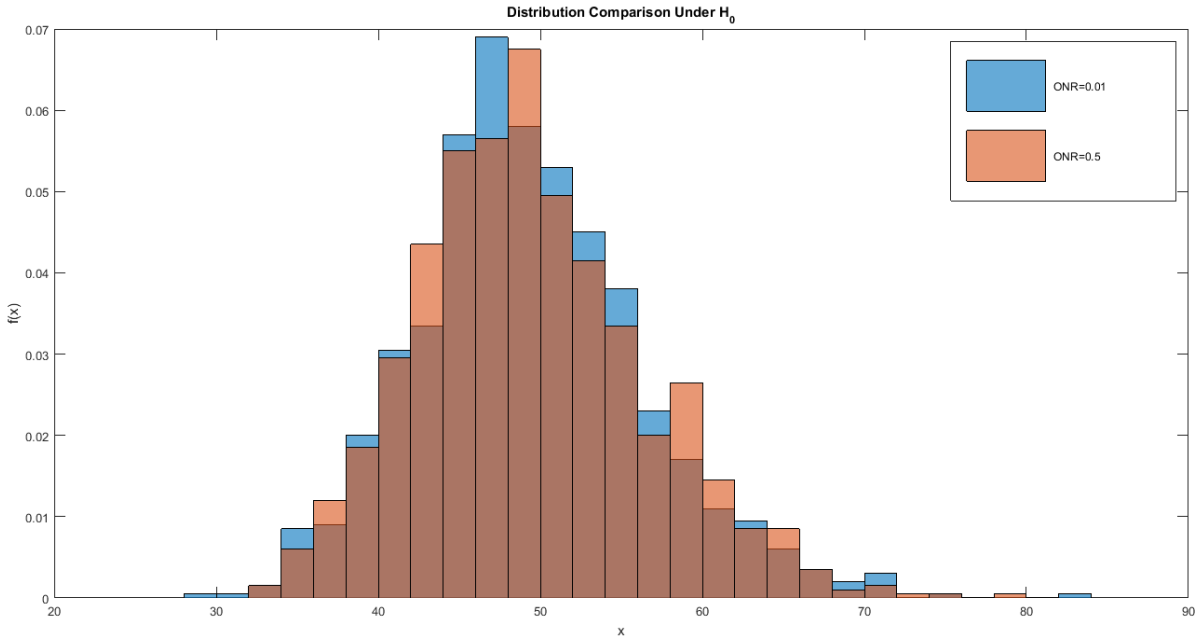


Figure 6.21: Empirical distribution under $\mathcal{H}_0$ for M.2a

Comparing the empirical distributions under $\mathcal{H}_0$ for the highest and lowest ONR values, we can observe that they are similar in shape.

Now we will show the empirical and theoretical $\overline{ROC}$-curves that are derived for the Lasso-method.

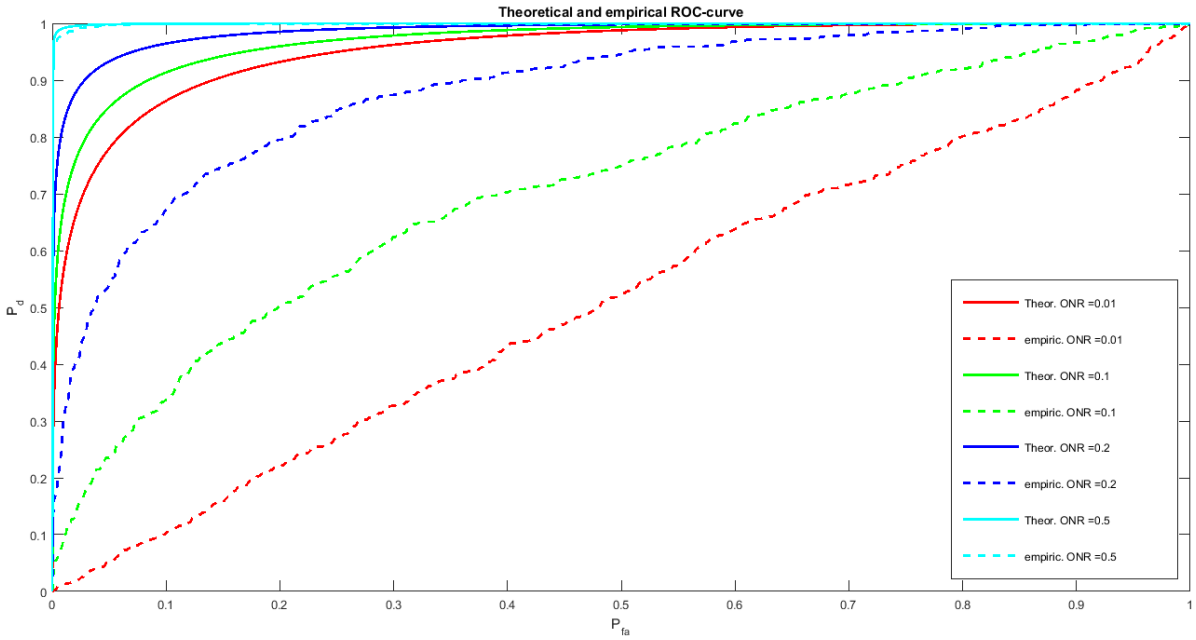


Figure 6.22: ROC-curves based on the Lasso-method for M2.a

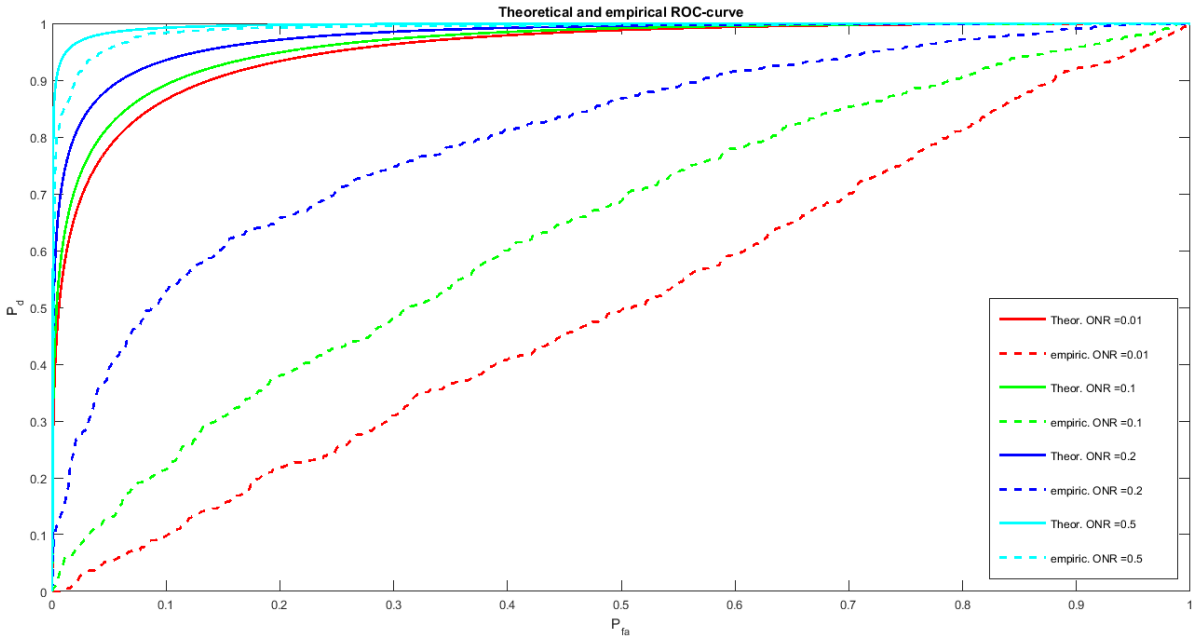


Figure 6.23: ROC-curves based on the Lasso-method for M2.b

Attack \ ONR	0.01	0.1	0.2	0.5
a	9.05	9.11	9.24	9.36
b	8.92	8.31	8.02	7.77

Table 6.5: $\overline{Error}$ for M.2 for the Lasso-method

The $\overline{ROC}$-curves are similar to the $\overline{ROC}$-curves in M.1, for the Lasso-methods.

Time-varying anomalous nodes

Now, we provide the empirical and theoretical $\overline{ROC}$-curves for M.3

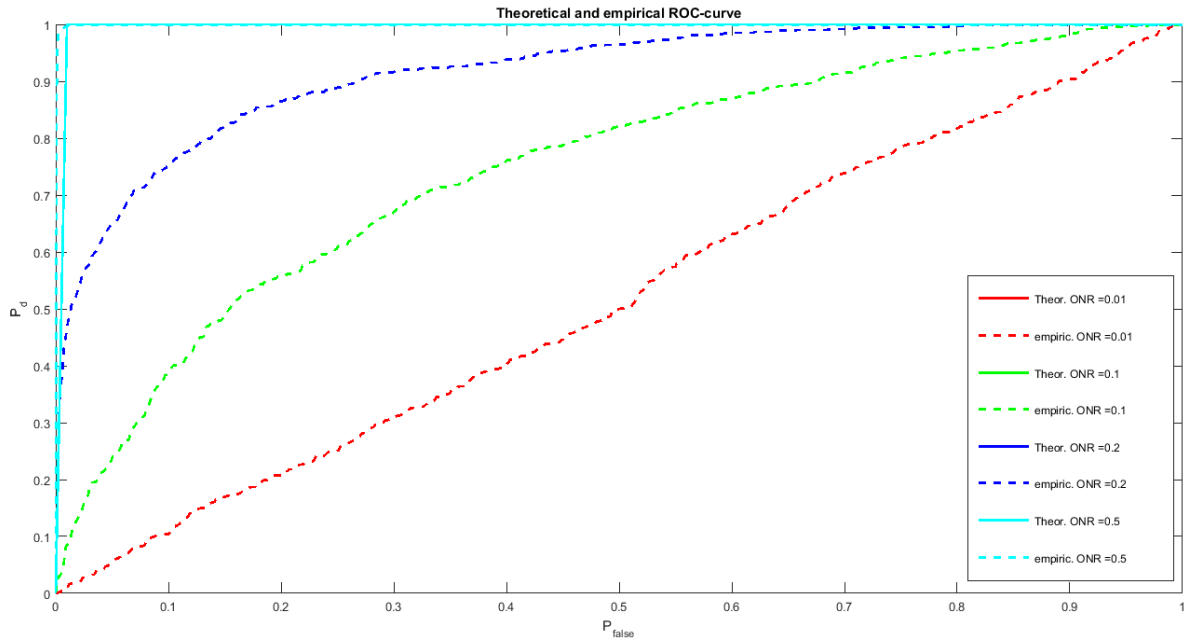


Figure 6.24: Theoretical and empirical $\overline{ROC}$-curve for M3.a

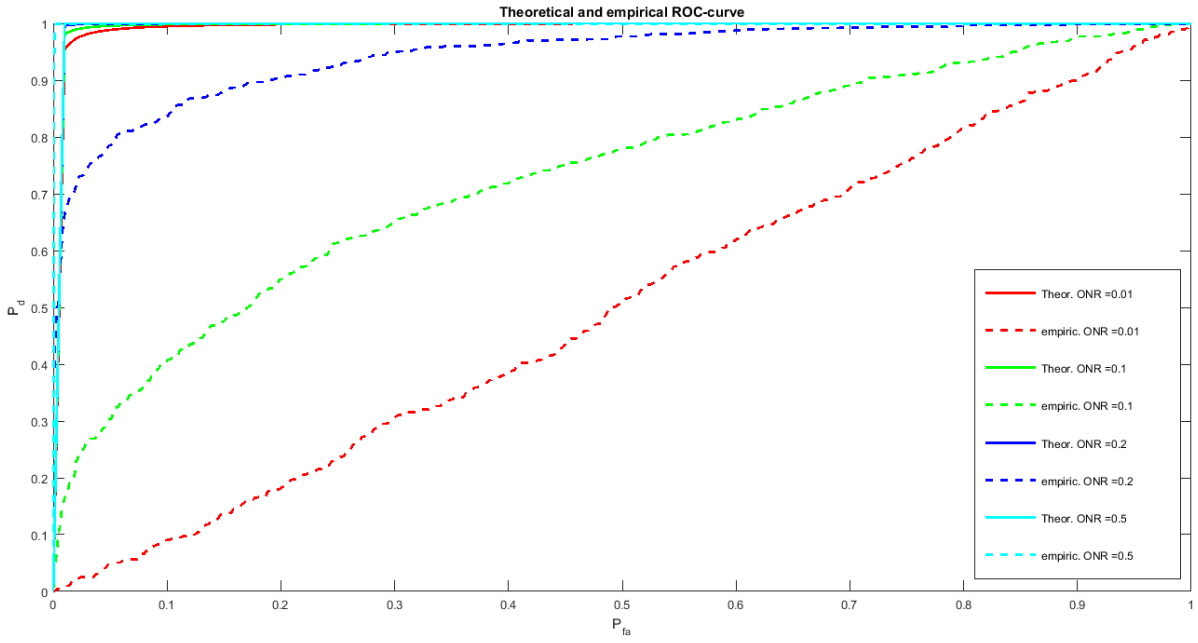


Figure 6.25: Theoretical and empirical $\overline{ROC}$-curve for M3.b

Attack \ ONR	0.01	0.1	0.2	0.5
a	8.83	7.74	6.88	5.10
b	7.09	5.15	3.30	1.19

Table 6.6: $\overline{Error}$ for M.3

For problem 3 we observe a similar pattern as for problem 2, hence there is not much to add for problem 3. The only difference now is that every timeframe i contains a different estimated support set $\mathbf{1}_{\mathcal{A}_i}$. The cardinality is now defined as the total cardinality for $i \in \{1, \ldots T\}$. Contrary to M.1b and M.2b, the bi-modal distribution is not observed for M.3b. For example, for the lowest ONR we observe a smooth single peak distributions:

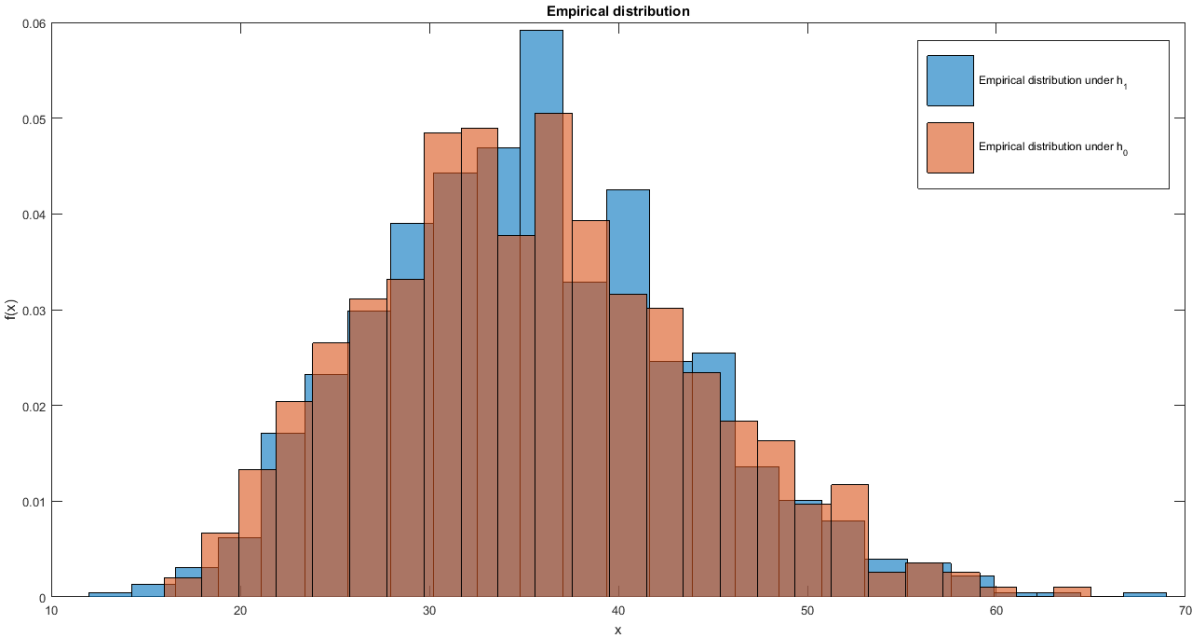


Figure 6.26: Empirical distribution and theoretical PDF's for M.3b, $ONR = 0.01$

However the 'appearance' can be deceiving in the sense that the varying cardinality suddenly looks like it has a different reasoning compared to M.1b and M.2b. In order to understand why we see a single peak, we need to look at the shape of the cardinality histogram:

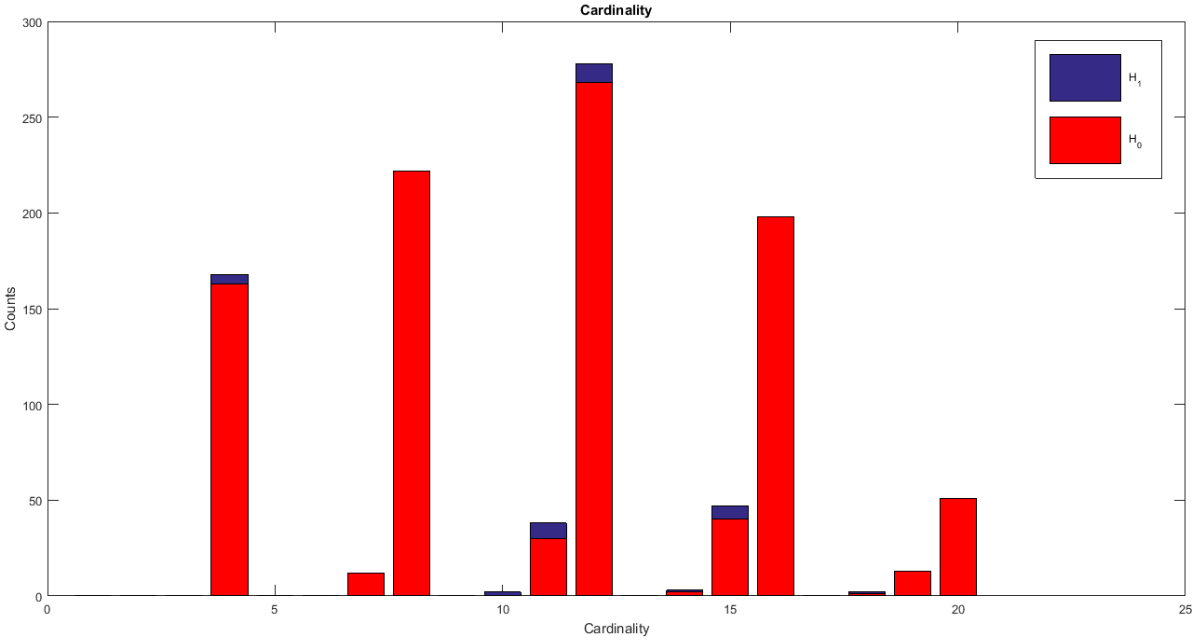


Figure 6.27: Cardinality counts for M.3b, $ONR = 0.01$ under $\mathcal{H}_0$

When we observe the cardinality histograms, we can see that the shape of the cardinality histogram

is similar to the distributions. With similarity is meant that there is a peak around a cardinality value of 12, i.e., the 'centre'. And a decreasing count can be seen for increasing or decreasing cardinality values, i.e., the 'tail'. The peaks in the distributions caused by the varying cardinality are now faded out to a single peak, because the cardinality values associated with these peaks are now lower in transition compared to $\mathrm{M.2b}$. For $\mathrm{M.2b}$, we observed a 'swinging' cardinality count between 1 and 5, although these cardinalities are not huge in difference, the total degrees of freedom is then swinging between 4 and 20. For $\mathrm{M.3}$ the cardinality values is equal to the total degrees of freedom. Hence, the main cause for a varying cardinality for $\mathrm{M.3b}$ is still similar as for $\mathrm{M.1b}$ and $\mathrm{M.2b}$, but it is less obvious when observing the empirical distributions.

Finally, the empirical and theoretical $\overline{ROC}$-curves for the Lasso-based method for M.3 will be given.

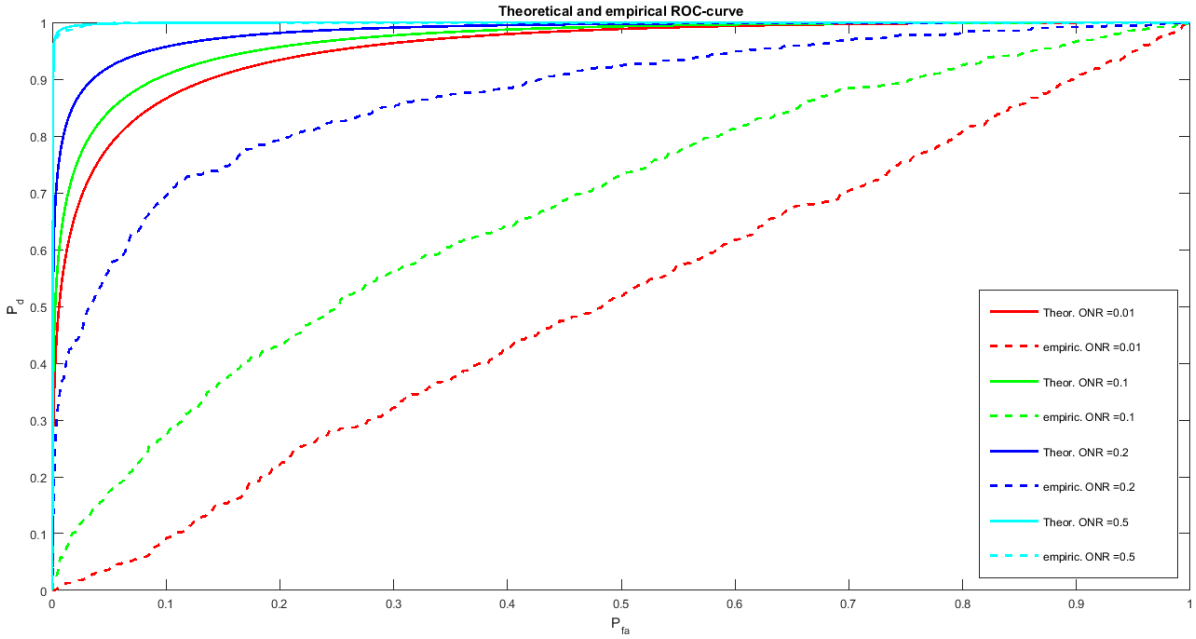


Figure 6.28: ROC-curves based on the Lasso-method for M3.a

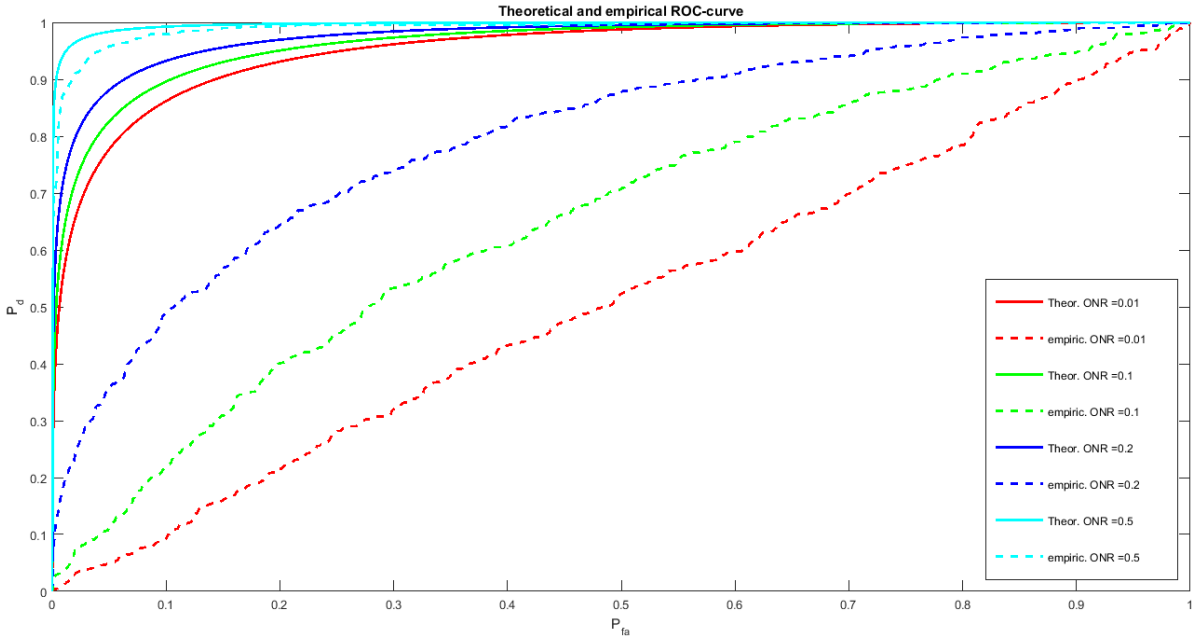


Figure 6.29: ROC-curves based on the Lasso-method for M3.b

Attack \ ONR	0.01	0.1	0.2	0.5
a	9.0190	9.1755	9.2240	9.3970
b	8.97	9.00	9.000	8.96

Table 6.7: $\overline{Error}$ for M.3 for the Lasso-method

6.4.3. $\overline{Error}$ Elucidation

For attack a, if we look at the provided framework, generally speaking, we can observe that M.1 gives the lowest $\overline{Error}$. This can be attributed to an equal distribution of the energy over the compromised nodes in M.1. A node with a higher energy input increases the chance of being detected, but this will be at the expense of nodes with a lower energy input. Resulting in a higher $\overline{Error}$.
For attack b, we observe that M.2 yields the lowest $\overline{Error}$ for the provided algorithm, generally speaking. The inequality of the energy over the nodes is then an advantage for the auditor. The nodes with higher energy will determine mostly the community to be selected during the rounding process.
Also, we observe now that the $\overline{Error}$ is higher for M.3 compared with M.2 for both attacks. For M.2, we only need to estimate a single set of compromised nodes. Hence, more information is available for the single set that must be estimated. For M.3, we need to estimate a different set for every timeframe, this can lead to a higher average error due to the single timeframe information for every estimated set.

For the Lasso method, a significant difference can not be observed between attack a and b. A difference that can be observed is between the time invariant and time-varying models. This can be once again attributed to distribution of the outlier energy over the nodes. The $\overline{Error}$ of M.2 and M.3 are higher since the energy of the outliers are not equally distributed over the compromised nodes. Nodes that have a smaller outlier energy input have a higher chance of being erroneously indexed as 'clean'.

6.5. Discussion

Despite that the empirical results are not as good as the theoretical results, we can observe that for an increasing ONR, starting from 0.5, the derived methods provide promising results. Furthermore, for an ONR as small as 0.1 we can still observe a significant steepness of the empirical ROC-curves. For the Lasso-based methods, we can observe similar results. However, the Lasso-based methods yield higher values for $\overline{Error}$ due to the non-utilization of all the constraints.

As explained in chapter 6.4, the set of feasible node combinations, i.e., the set of all the possible node combinations that are subjected to the constraint set, is of great influence on the selection of the malicious nodes. A smaller feasible nodeset can lead to a higher 'spreading out' of the binary relaxed values.

Combined with the feasible nodeset, we also explained how the distribution of the outlier energy over the nodes affects the $\overline{Error}$ in chapter 6.4.3. Equally distributed outlier energy provides a lower $\overline{Error}$ for a larger feasible nodeset. While, for a more restricted nodeset, as in attack b, this is not the case. Energy that is accumulated into a single node for example, can lead to a better decision making for the estimation of $\mathbf{1}_{\tilde{\mathcal{A}}}$, for smaller feasible nodesets.

For M.3, since that we need to estimate $\mathbf{1}_{\mathcal{A}i}$ for all i, generally speaking, this will lead to higher $\overline{Error}$ since that we have less 'information' for every $\mathbf{1}_{\mathcal{A}i}$ that we need to estimate.

In the next chapter, we will discuss three different ideas for future research.

7

Future work

In this chapter, we will propose two different ideas for future research. For the first idea, we provide a different reasoning for the use of the detection theory metrics. The second idea is based on solving the classical outlier detection theory with the help of state of the art regression methods and our current contribution.

7.1. Framework for Colored Noise

Similar as what we did in chapter 4, it is also possible to derive a framework for in case that we are dealing with colored noise, i.e.,

$$\boldsymbol{n} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{C}) \tag{7.1}$$

The maximum likelihood estimation for colored Gaussian noise looks as follows [21]

$$\boldsymbol{o}^* = \underset{\boldsymbol{o}}{\operatorname{argmin}} (\boldsymbol{y} - \boldsymbol{o})^T \boldsymbol{C}^{-1} (\boldsymbol{y} - \boldsymbol{o}) \tag{7.2}$$

From which we must derive an expression for $\boldsymbol{o}$. It is then possible as in chapter 4 to derive different optimization problems for the different models.
The standard notation of the test statistic for colored noise is given as [25]

$$T(\boldsymbol{z}) = \boldsymbol{o}^T \boldsymbol{C}^{(-1)} \boldsymbol{z} - \frac{1}{2} \boldsymbol{o}^T \boldsymbol{C}^{(-1)} \boldsymbol{o} \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \gamma \tag{7.3}$$

From which we derive expressions for the distributions and the false alarm and detection probabilities similar as in chapter 4.

7.1.1. Model 1

For M.1, i.e.,

$$\boldsymbol{o} = c \mathbf{1}_{\mathcal{A}} \tag{7.4}$$

the optimal value for c is derived as

$$c = \frac{\mathbf{1}_{\mathcal{A}}{}^T \boldsymbol{C}^{-1} \boldsymbol{z}}{\mathbf{1}_{\mathcal{A}}{}^T \boldsymbol{C}^{-1} \mathbf{1}_{\mathcal{A}}} \tag{7.5}$$

By substituting this formulation for $\boldsymbol{o}$ in the standard formation given in (7.3), we get the following formulation for the test statistic

$$T(\boldsymbol{z}) = \frac{|\mathbf{1}_{\mathcal{A}}{}^T \boldsymbol{C}^{-1} \boldsymbol{z}|}{\sqrt{\mathbf{1}_{\mathcal{A}}{}^T \boldsymbol{C}^{-1} \mathbf{1}_{\mathcal{A}}}} \underset{H_0}{\overset{H_1}{\gtrless}} \gamma' \tag{7.6}$$

The details are given in appendix E.
Once again, the test statistic is a folded Gaussian function with the following distributions under both

hypotheses

$$\begin{aligned}\mathcal{H}_0: \quad T([\boldsymbol{z}]) &\sim \begin{cases} 2\mathcal{N}(0,\sigma^2), & \text{if } T([\boldsymbol{z}]) \geq 0, \\ 0, & \text{if } T([\boldsymbol{z}]) < 0, \end{cases} \\ \mathcal{H}_1: \quad T([\boldsymbol{z}]) &\sim \begin{cases} \mathcal{N}(A,\sigma^2) + \mathcal{N}(-A,\sigma^2), & \text{if } T([\boldsymbol{z}]) \geq 0, \\ 0, & \text{if } T([\boldsymbol{z}]) < 0, \end{cases} \end{aligned} \tag{7.7}$$

where A is now defined as

$$A = \frac{\mathbf{1}_{\tilde{\mathcal{A}}}{}^T \boldsymbol{C}^{-1} \boldsymbol{z}}{\sqrt{\mathbf{1}_{\tilde{\mathcal{A}}}{}^T \boldsymbol{C}^{-1} \mathbf{1}_{\tilde{\mathcal{A}}}}} \tag{7.8}$$

From which we can derive formulations for P_d and P_{fa} respectively as

$$P_d = Q\left(\frac{\gamma' - A}{\sqrt{\sigma^2}}\right) + Q\left(\frac{\gamma' + A}{\sqrt{\sigma^2}}\right) \tag{7.9}$$

$$P_{fa} = 2Q\left(\frac{\gamma'}{\sqrt{\sigma^2}}\right) \tag{7.10}$$

and upon substitution we get

$$P_d = Q\left(Q^{-1}(P_{fa}/2) - \frac{A}{\sqrt{\sigma^2}}\right) + Q\left(Q^{-1}(P_{fa}/2) + \frac{A}{\sqrt{\sigma^2}}\right) \tag{7.11}$$

Hence our cost function for our optimization problem is defined as

$$f_{P_{d1}}(\mathbf{1}_{\tilde{\mathcal{A}}}) = \frac{|\mathbf{1}_{\tilde{\mathcal{A}}}{}^T \boldsymbol{C}^{-1} \boldsymbol{z}|}{\sqrt{\mathbf{1}_{\tilde{\mathcal{A}}}{}^T \boldsymbol{C}^{-1} \mathbf{1}_{\tilde{\mathcal{A}}}}} \tag{7.12}$$

with the same reasoning given as in chapter 4.
Similarly as in chapter 5, it is possible to use the Dinkelbach method for the relaxed formulation for our cost function. To do so, we first square and rewrite the cost function, similar as in chapter 5, as

$$f_{P_{d1}}(\mathbf{1}_{\tilde{\mathcal{A}}}) = \frac{Tr(\mathbf{1}_{\tilde{\mathcal{A}}} \mathbf{1}_{\tilde{\mathcal{A}}}{}^T \boldsymbol{C}^{-1} \boldsymbol{z}\boldsymbol{z}^T \boldsymbol{C}^{-1})}{Tr(\mathbf{1}_{\tilde{\mathcal{A}}} \mathbf{1}_{\tilde{\mathcal{A}}}^T \boldsymbol{C}^{-1})} \tag{7.13}$$

and relax the denominator and nominator as follows

$$\tilde{f}_{P_{d1}} = \frac{Tr(\boldsymbol{X}_B \boldsymbol{C}^{-1} \boldsymbol{z}\boldsymbol{z}^T \boldsymbol{C}^{-1})}{Tr(\boldsymbol{X}_B \boldsymbol{C}^{-1})} \tag{7.14}$$

where the variable $\boldsymbol{X}_B$ is the same variable as in chapter 5.2. The optimization problem is similarly solved as in the algorithm in 1. The constraint set in the algorithm does not change for the colored noise framework and thus will be the same as in (5.19).

7.1.2. Model 2 and 3

For M.2 and M.3 we get a similar optimization problem as in (4.49). In order to derive an expression for $\boldsymbol{o}$, we first rewrite the MLE as

$$\boldsymbol{o}^* = \underset{\boldsymbol{o}}{\operatorname{argmin}} \|\boldsymbol{D}\boldsymbol{z} - \boldsymbol{D}\boldsymbol{o}\|_2^2 \tag{7.15}$$

where $\boldsymbol{D}$ is defined as

$$\boldsymbol{C}^{-1} = \boldsymbol{D}^T \boldsymbol{D} \tag{7.16}$$

From this formulation, it is clear that our formulation for $\boldsymbol{o}$ is derived as

$$\boldsymbol{D}\boldsymbol{o} = diag(\mathbf{1}_{\tilde{\mathcal{A}}}) D\boldsymbol{z} \tag{7.17}$$

thus

$$\boldsymbol{o} = \boldsymbol{D}^{-1} diag(\mathbf{1}_{\tilde{\mathcal{A}}}) D\boldsymbol{z} \tag{7.18}$$

Now, substituting this formulation in the equation given in (7.3) we get the following test statistic function

$$T(\mathbf{1}_{\tilde{\mathcal{A}}}) = \boldsymbol{z}^T \boldsymbol{D}^T diag(\mathbf{1}_{\tilde{\mathcal{A}}}) \boldsymbol{D}\boldsymbol{z} \tag{7.19}$$

Which is a squared summation of prewhitened Gaussian data, i.e., χ^2 distribution, for both hypotheses.

$$\begin{aligned} \mathcal{H}_0 : \quad & T(\boldsymbol{z}) \sim \chi^2_{|\{\mathcal{A}_1\}|} \\ \mathcal{H}_1 : \quad & T(\boldsymbol{z}) \sim \chi^2_{|\{\mathcal{A}_1\}|}(\mu) \end{aligned} \tag{7.20}$$

For colored noise, μ is defined as

$$\mu = \frac{\boldsymbol{z}^T \boldsymbol{D}^T diag(\mathbf{1}_{\tilde{\mathcal{A}}}) \boldsymbol{D}\boldsymbol{z}}{\sigma^2} \tag{7.21}$$

From this, we can denote the detection and false alarm probabilities as

$$P_d = Q(\gamma')_{\chi^2_{|\{\mathcal{A}_1\}|}(\mu)} \tag{7.22}$$

$$P_{fa} = Q(\gamma')_{\chi^2_{|\{\mathcal{A}_1\}|}} \tag{7.23}$$

and substitution gives

$$P_d = Q(Q^{-1}(P_{fa})_{\chi^2_{|\mathcal{A}|}})_{\chi^2_{|\mathcal{A}|}(\mu)} \tag{7.24}$$

Now, we get a similar cost function as in (4.49) with a similar reasoning given for that cost function.

$$f_{P_{d2}}(\mathbf{1}_{\tilde{\mathcal{A}}}) = f_{P_{d3}}(\mathbf{1}_{\tilde{\mathcal{A}}}) = \boldsymbol{z}^T \boldsymbol{D}^T diag(\mathbf{1}_{\tilde{\mathcal{A}}}) \boldsymbol{D}\boldsymbol{z} \tag{7.25}$$

Where the optimization problem is solved for different values of the cardinality constraint. This cost function can now be rewritten as

$$f_{P_{d2}}(\mathbf{1}_{\tilde{\mathcal{A}}}) = f_{P_{d3}}(\mathbf{1}_{\tilde{\mathcal{A}}}) = Tr(\boldsymbol{D}\boldsymbol{z}\boldsymbol{z}^T \boldsymbol{D}^T diag(\mathbf{1}_{\tilde{\mathcal{A}}})) \tag{7.26}$$

From which we get the following relaxed cost function

$$\tilde{f}_{P_{d2}} = \tilde{f}_{P_{d3}}(\boldsymbol{x}) = Tr(\boldsymbol{D}\boldsymbol{z}\boldsymbol{z}^T \boldsymbol{D}^T diag(\boldsymbol{x})) \tag{7.27}$$

The variable $\boldsymbol{x}$ is the same variable given in chapter 5.2.

7.2. Undermining the Current Framework

The focus in this thesis was put on the identification of the compromised nodes. This identification is achieved by using the metrics that are provided in detection theory. We can re-motivate the usage of these metrics such that a new optimization problem can be derived. The framework that is derived in this thesis identifies the compromised nodes with the help of the maximization of the detection probability, P_d. A motivation for the optimization of the detection probability was given in chapter 3.1. The awareness of this framework by the malicious agents can lead to a behavior that is focused on undermining this framework.

In chapter 3.3, we explained how it is possible to steer a consensus based distributed network with the help of an attacking schedule. Likewise, the agents can come up with a new adjusted attacking schedule that will minimize the detection probability **while** there is still a non negligible or lowerbound of influence on the network. From the formulations that is derived for the detection probability in chapter 4.3, we can see that they are dependent on the estimated energy of the outliers. A higher energy leads to higher chances of being detected. Consequently, for the time varying models, the agents can lower the risk of detection by adjusting the entries of $\boldsymbol{z} = \boldsymbol{o} + \boldsymbol{n}$ such that the absolute values of the entries corresponding to the compromised nodes are low compared with the clean nodes. The choice for the outlying values, for the agents, boils then down to the consideration between the influence on the network and risk of detection. A higher influence means a looser bound on the outlying values, which can

result in higher chances of detection due to the possible higher energy input. For the identification of the compromised nodes, an alternative optimization problem for M.2 must be formulated then:

$$\begin{aligned}
\min_{\boldsymbol{x},\boldsymbol{x}_c} \quad & \boldsymbol{z}^T diag(\boldsymbol{x})\boldsymbol{z} \\
\text{subject to} \quad & B_1(\boldsymbol{x}_c) \leq \rho \quad or \quad C_1(\boldsymbol{x}_c) \leq \rho, \\
& K_1(\boldsymbol{x}_c) \leq \beta, \\
& K_1(\boldsymbol{x}_c) \geq \delta, \\
& D_1(\boldsymbol{x}_c) \geq \alpha, \\
& 0 \leq [\boldsymbol{x}_c]_i \leq 1, \qquad : \forall \quad i \in \{1, \ldots N\}, \\
& \boldsymbol{x}_c = \boldsymbol{x}_i, \qquad : \forall \quad i \in \{1, \ldots . T\}
\end{aligned} \tag{7.28}$$

Apart from the minimization of the cost function, we introduced a new constraint for the lowerbound of the cardinality. Only an upperbound for the cardinality will give a trivial solution for $\boldsymbol{x}_c$ in the form of an all zero vector since that we are minimizing the total energy. Instead, we will solve the optimization problem for different values of a **lowerbound** δ. The complete algorithm will then be similar as the algorithm derived in the algorithmic scheme of 3, the only difference is the above mentioned changes in the optimization problem part of the algorithm. The different values obtained for the estimation of $\mathbf{1}_{\tilde{\mathcal{A}}}$ can then be used in order to find the minimum of P_d.
For M.3, the problem is almost similar. However, the malicious agents can be more specific now in choosing the compromised nodes since that a higher freedom is acquired by the malicious agents. Whereas in M.2 the agents can only adjust the values of the outliers, in M.3 the agents can also determine which nodes to compromise on the basis of the observed signal values. Hence, it can be said that the model of M.3 will be more effective in the undermining of the current framework:

$$\begin{aligned}
\min_{\boldsymbol{x}} \quad & \boldsymbol{z}^T diag(\boldsymbol{x})\boldsymbol{z} \\
\text{subject to} \quad & B_1(\boldsymbol{x}_i) \leq \rho \quad or \quad C_1(\boldsymbol{x}_i) \leq \rho, \quad : \forall \quad i \in \{1, \ldots . T\}, \\
& K_1(\boldsymbol{x}_i) \leq \beta, \qquad : \forall \quad i \in \{1, \ldots . T\}, \\
& K_1(\boldsymbol{x}_i) \geq \delta, \qquad : \forall \quad i \in \{1, \ldots . T\}, \\
& D_1(\boldsymbol{x}_i) \leq \alpha, \qquad : \forall \quad i \in \{1, \ldots . T\}, \\
& 0 \leq [\boldsymbol{x}_i]_j \leq 1 \qquad : \forall \quad j \in \{1, \ldots N\} \wedge i \in \{1, \ldots . T\}
\end{aligned} \tag{7.29}$$

7.3. Combining Regression methods and Current Framework

Often, it is not possible to reformulate the problem of outlier detection as we did in chapter 4.1. This means that the hypothesis testing problem will look as a classical outlier detection problem:

$$\begin{aligned}
\mathcal{H}_0 : \quad & \boldsymbol{y} = \boldsymbol{s} + \boldsymbol{n} \\
\mathcal{H}_1 : \quad & \boldsymbol{y} = \boldsymbol{s} + \boldsymbol{o} + \boldsymbol{n}
\end{aligned} \tag{7.30}$$

By making use of the GLRT, a threshold function is derived that looks as follows:

$$T(\boldsymbol{y}) = \boldsymbol{y}^T(\boldsymbol{s}_1 + \boldsymbol{o} - \boldsymbol{s}_0) \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \gamma \tag{7.31}$$

Where $\boldsymbol{s}_0$ stands for the estimated signal value for $\mathcal{H}_0$, $\boldsymbol{s}_1$ is the estimated signal value for $\mathcal{H}_1$ and $\boldsymbol{o}$ is the vector with estimated outlier values. Steps are provided in Appendix D.
The distribution of the threshold function is denoted as follows for both of the hypotheses:

$$\begin{aligned}
\mathcal{H}_0 : \quad & T(\boldsymbol{z}) \sim \mathcal{N}(\boldsymbol{s}_0^T(\boldsymbol{s}_1 + \boldsymbol{o}) - \boldsymbol{s}_0^T\boldsymbol{s}_0, (\boldsymbol{s}_1 + \boldsymbol{o} - \boldsymbol{s}_0)^T(\boldsymbol{s}_1 + \boldsymbol{o} - \boldsymbol{s}_0)\sigma^2) \\
\mathcal{H}_1 : \quad & T(\boldsymbol{z}) \sim \mathcal{N}((\boldsymbol{s}_1 + \boldsymbol{o})^T(\boldsymbol{s}_1 + \boldsymbol{o}) - \boldsymbol{s}_0^T(\boldsymbol{s}_1 + \boldsymbol{o}), (\boldsymbol{s}_1 + \boldsymbol{o} - \boldsymbol{s}_0)^T(\boldsymbol{s}_1 + \boldsymbol{o} - \boldsymbol{s}_0)\sigma^2)
\end{aligned} \tag{7.32}$$

And also the P_d and P_{fa} are formulated respectively as

$$P_d = Q\left(\frac{\lambda' - ((\boldsymbol{s}_1 + \boldsymbol{o})^T(\boldsymbol{s}_1 + \boldsymbol{o}) - \boldsymbol{s}_0^T(\boldsymbol{s}_1 + \boldsymbol{o}))}{\sqrt{(\boldsymbol{s}_0 - (\boldsymbol{s}_1 + \boldsymbol{o}))^T(\boldsymbol{s}_0 - (\boldsymbol{s}_1 + \boldsymbol{o}))\sigma^2}}\right) \tag{7.33}$$

$$P_{fa} = Q\left(\frac{\lambda' - (\boldsymbol{s}_0^T(\boldsymbol{s}_1 + \boldsymbol{o}) - \boldsymbol{s}_0^T\boldsymbol{s}_0)}{\sqrt{(\boldsymbol{s}_0 - (\boldsymbol{s}_1 + \boldsymbol{o}))^T(\boldsymbol{s}_0 - (\boldsymbol{s}_1 + \boldsymbol{o}))\sigma^2}}\right) \tag{7.34}$$

Substitution gives:

$$P_d = Q(Q^{-1}(P_{fa}) - \sqrt{\frac{(\boldsymbol{s}_0 - (\boldsymbol{s}_1 + \boldsymbol{o}))^T(\boldsymbol{s}_0 - (\boldsymbol{s}_1 + \boldsymbol{o}))}{\sigma^2}}) \tag{7.35}$$

Since that the GLRT requires a maximum likelihood estimation of the unknown parameters, we must first estimate these parameters.

$$\boldsymbol{s}_0^* = \underset{\boldsymbol{s}_0}{\operatorname{argmax}}\, p(\boldsymbol{y} = \boldsymbol{s}_0 | \boldsymbol{s}_0) \tag{7.36}$$

$$(\boldsymbol{s}_1^*, \boldsymbol{o}^*) = \underset{\boldsymbol{s}_1, \boldsymbol{o}}{\operatorname{argmax}}\, p(\boldsymbol{y} = \boldsymbol{s}_1 + \boldsymbol{o} | \boldsymbol{s}_1, \boldsymbol{o}) \tag{7.37}$$

As it is mentioned before in this thesis, it is not possible to leverage the information regarding the support of the outliers that needs to be estimated directly into an optimization problem for the estimation of the maximum likelihood as:

$$\begin{aligned} \min_{\boldsymbol{s}_1, \boldsymbol{o}} \quad & \|\boldsymbol{y} - \boldsymbol{s}_1 - \boldsymbol{o}\|_2^2 \\ \text{subject to} \quad & \boldsymbol{s}_1 \in \mathcal{C}_s, \\ & \mathbf{1}_{\mathcal{A}} \in \mathcal{C}, \end{aligned} \tag{7.38}$$

Where $\mathcal{C}_s$ is the constraint set for the signal that is related to the underlying structure of the network, see chapter 1.
Instead, we propose an idea that will combine the framework provided in this thesis together with a regression method. Hence, we will use our post and previous work as building blocks for the proposed idea.

Firstly, we estimate the signal and the outliers without the information of the support set. Only a regression parameter is used for the tuning of the estimation.

$$\begin{aligned} (\boldsymbol{o}_R^*, \boldsymbol{s}_1^*) = \quad & \underset{\mathbf{1}_{\mathcal{A}}}{\operatorname{argmin}} \quad \|\boldsymbol{y} - \boldsymbol{s}_1 - \boldsymbol{o}_R\|_2^2 + \lambda \|\boldsymbol{o}_R\|_1 \\ & \text{subject to} \quad \boldsymbol{s}_1 \in \mathcal{C}_s, \end{aligned} \tag{7.39}$$

In equation (7.39), the Lasso based regression method is denoted. For a more refined regression method we can also use the elastic net method, see chapter 4.2.
From the estimated outlier signal $\boldsymbol{o}_R$, we carry out a second maximum likelihood estimation where we can now use the relaxations derived in chapter 5.2 in order to estimate the outliers together with the information provided about the network structure.

$$\begin{aligned} (\boldsymbol{o}^*) = \quad & \underset{\mathbf{1}_{\mathcal{A}}}{\operatorname{argmin}} \quad \left\|\boldsymbol{o}_R^* - \boldsymbol{o}\right\|_2^2 \\ & \text{subject to} \quad \mathbf{1}_{\mathcal{A}} \in \mathcal{C}, \end{aligned} \tag{7.40}$$

Despite that we substitute now $\boldsymbol{o}$ directly with an expression dependent on $\mathbf{1}_{\mathcal{A}}$ and calculate the estimation directly in equation (7.40), it is shown in section 4.4 that we will get the identical optimization problem as for the matched filter that is derived in chapter 4.3 for all three models. Hence, the relaxations in chapter 5.2 are still applicable.
Alternatively, it is also possible to carry out the second optimization problem for $\boldsymbol{y} - \boldsymbol{s}_1^*$ instead of $\boldsymbol{o}_R^*$, i.e.:

$$\begin{aligned} (\boldsymbol{o}^*) = \quad & \underset{\mathbf{1}_{\mathcal{A}}}{\operatorname{argmin}} \quad \left\|\boldsymbol{y} - \boldsymbol{s}_1^* - \boldsymbol{o}\right\|_2^2 \\ & \text{subject to} \quad \mathbf{1}_{\mathcal{A}} \in \mathcal{C}, \end{aligned}, \tag{7.41}$$

which should give a lower mean square error (MSE) then when we apply the optimization problem in equation (7.40).
The total MSE is now equal to:

$$\epsilon = \frac{1}{N} \left\|\boldsymbol{y} - \boldsymbol{s}_1^* - \boldsymbol{o}^*\right\|_2^2 \tag{7.42}$$

The main idea behind this method is that we can simulate values for the signal and outliers for varying values of the regression parameter λ. Thereby, we obtain a set of values for the signal and outliers where we can choose the values that gives us the lowest ϵ. The main disadvantage of this idea is the

fact that we can not identify when the optimal ϵ is achieved. For example, starting with $\lambda = 0$, if we can observe a trend where ϵ is decreasing for an increasing λ until a certain value for λ, it is not guaranteed that a minimum is reached for ϵ. Instead, a further increase in λ can result in an unpredictable behavior in terms of the trend of the ϵ. However, we can know that after a certain value for λ, the values that we get for $\boldsymbol{o}_R$ or $\boldsymbol{y} - \boldsymbol{s}_1^*$ will result in a trivial vector very close to zero for the Lasso method (presumably, also for the elastic net method). Hence, a further increase of λ will not give non-trivial values for $\boldsymbol{o}_R$. Thus, by choosing a step value δ, after a certain amount of steps for λ, we can investigate which of these values (from the set of estimated values for $\boldsymbol{s}_1$ and $\boldsymbol{o}$) will give us an optimal value for ϵ.

In the next chapter, we will conclude our thesis by giving a brief summary over the thesis.

8

Conclusion

In this chapter we will briefly summarize the work that we have conducted throughout the thesis. Furthermore, we will give a conclusion for our numerical findings and give a short explanation what the future work incloses.

The scope of this thesis was to find a way for the detection of outliers while structure based information about the network is utilized. In chapter 3, we showed how we can make use of detection theory for the derivation and motivation of an optimization problem that will identify these outliers. Furthermore, for the modeling of the outlying values, we proposed three different anomalous action models. All three models provide a different optimization problem. Also in chapter 3, we provided a motivation for the structuring to be used of these outliers over the network.

In chapter 4, we showed how we can exploit the data model such that the problem can be re-defined as a structured signal detection problem. An explanation is also given about the shortcomings of the classical regression methods that enforce sparsity. Finally, we derived the optimization problems for all three outlier models.

In chapter 5, we showed how it is possible to convexify these optimization problems that are derived in the previous chapter. Firstly, a relaxation is provided for the constraints. Secondly, a combination of fractional programming and semidefinite relaxation is used for the cost function of $\mathrm{M}.1$. Furthermore, a simple linear relaxation is provided for $\mathrm{M}.2$ and $\mathrm{M}.3$.

Chapter 6 is dedicated to numerical experiments for the evaluation of the relaxation algorithms provided in the previous chapter. We explained why the empirical results for the ROC-curves do not match the theoretical results. Furthermore, an explanation is also given for the varying cardinality values of the estimated nodeset and how these are related to the attack modes a and b. An observation is also made for the average error of the Lasso method and the provided framework. We observe that the Lasso method gives a higher average error due to the lack of utilization of the network properties. We also explained how the distribution of the outlier energy over the nodes affects the average error together with the constraint set.

Finally, in chapter 7, two different ideas are proposed that can be considered for future research. These ideas are related to the conducted work in this thesis. The first idea is based on reconsidering the usage of the metrics of detection theory. The second idea explains how a regression method can be combined with the provided framework in order to solve the classical outlier detection problem while the structure based information about the network is utilized.

A

Model 1 Log-Likelihood Ratio Test Derivation

First, we calculate c by taking the derivative of the mean square error and forcing it to zero.

$$\frac{\partial \sum_{i=1}^{NT}[(\boldsymbol{z}-c\mathbf{1}_{\tilde{\mathcal{A}}})]_i^2}{\partial c} = -2\sum_{i\in\mathcal{A},j\in 1,..T}[(\boldsymbol{z}-c\mathbf{1}_{\tilde{\mathcal{A}}})]_{ij} = 0 \tag{A.1}$$

From which we get the following equation

$$\sum_{i\in\mathcal{A},i\in 1,..T}[(\boldsymbol{z}-c\mathbf{1}_{\tilde{\mathcal{A}}})]_i = \mathbf{1}_{\tilde{\mathcal{A}}}{}^T\boldsymbol{z} - cT|\mathcal{A}| = 0 \tag{A.2}$$

Thus c can be formulated as:

$$c = \frac{\mathbf{1}_{\tilde{\mathcal{A}}}{}^T\boldsymbol{z}}{|\mathcal{A}|(T)} \tag{A.3}$$

Now, by using the standard derived notation for the Log-likelihood ratio for the classical white Gaussian noise, signal detection problem(4.17). We get the following expression by substitution:

$$T(\boldsymbol{z}) = \frac{c}{\sigma^2}\sum_{i\in\mathcal{A},j\in 1,..T}[\boldsymbol{z}]_{ij}[\mathbf{1}_{\tilde{\mathcal{A}}}]_{ij} - \frac{c^2}{2\sigma^2}\sum_{i\in\mathcal{A},i\in 1,..T}[\mathbf{1}_{\tilde{\mathcal{A}}}]_{ij} \tag{A.4}$$

Substituting c with the derived equation:

$$T(\boldsymbol{z}) = \frac{1}{\sigma^2}\frac{\mathbf{1}_{\tilde{\mathcal{A}}}{}^T\boldsymbol{z}}{|\mathcal{A}|(T)}\sum_{i\in\mathcal{A},j\in 1,..T}[\boldsymbol{z}]_{ij} - \frac{1}{2\sigma^2}\left(\frac{\mathbf{1}_{\tilde{\mathcal{A}}}{}^T\boldsymbol{z}}{|\mathcal{A}|(T)}\right)^2\sum_{i\in\mathcal{A},i\in 1,..T}[\mathbf{1}_{\tilde{\mathcal{A}}}]_{ij} \tag{A.5}$$

This can be further simplified to:

$$T(\boldsymbol{z}) = \frac{1}{\sigma^2}\frac{(\mathbf{1}_{\tilde{\mathcal{A}}}{}^T\boldsymbol{z})^2}{|\mathcal{A}|(T)} - \frac{1}{2\sigma^2}\frac{(\mathbf{1}_{\tilde{\mathcal{A}}}{}^T\boldsymbol{z})^2}{|\mathcal{A}|(T)} = \frac{1}{2\sigma^2}\frac{(\mathbf{1}_{\tilde{\mathcal{A}}}{}^T\boldsymbol{z})^2}{|\mathcal{A}|(T)} \tag{A.6}$$

Which gives the final simplification for $T(\boldsymbol{z})$:

$$T(\boldsymbol{z}) = \frac{|\boldsymbol{z}^T\mathbf{1}_{\tilde{\mathcal{A}}}|}{\sqrt{(T)|\mathcal{A}|}} \underset{H_0}{\overset{H_1}{\gtrless}} \gamma' \tag{A.7}$$

B

Rounding Methods

Rounding algorithms, for all three rounding algorithms $\mathcal{C}_1$ is defined as the 'hard' constraint set. Furthermore, the $Bern(x)$ operator returns the outcome of a Bernouille trial with a probability of x.

B.1. For Time-invariant attacks

Algorithm 4 Rounding Algorithm for M.1

Input: $\boldsymbol{x}_c \in [0,1]^N$
output: $\mathbf{1}_{\mathcal{A}}^*$
for l ←1 **to** c **do**
    for j ←1 **to** N **do**
        $[\boldsymbol{s}_l]_j = Bern([\boldsymbol{x}_c]_j)$
    end for
    $\boldsymbol{s}_l^{(1)}$ is the stacking of T amounts of the vector $\boldsymbol{s}_l$
    $[\boldsymbol{c}]_l = \frac{(\boldsymbol{s}_l{}^T \boldsymbol{z})^2}{\|\boldsymbol{s}_l\|_1}$
end for
$\boldsymbol{S}$ is the matrix with columns $\boldsymbol{s}_l^{(1)}$ for all $l \in \{0,1,2,\ldots c\}$
Sort the elements of $\boldsymbol{c}$ in descending order.
Sort the columns of $\boldsymbol{S}$ with the same order that is used to sort $\boldsymbol{c}$.
hit=0
i=1
while hit=0 **do**
    if $[\boldsymbol{S}]_{i*} \in \mathcal{C}_1$ **then**
        hit=1
    else
        i=i+1
    end if
end while
$\mathbf{1}_{\mathcal{A}}^*$ is equal to the 'unstacked' form of vector $[\boldsymbol{S}]_{i*}$
return $\mathbf{1}_{\mathcal{A}}^*$

B.2. For Time-variant attacks

Algorithm 5 Rounding Algorithm for M.2

Input: $(\boldsymbol{x}_c, \beta) : \boldsymbol{x}_c \in [0,1]^N, \beta > 0$
output: $\mathbf{1}^*_{\mathcal{A}}$
β is defined as the cardinality constraint for $\mathcal{C}_1$.
for l ←1 **to** c **do**
    for j ←1 **to** N **do**
        $[\boldsymbol{s}_{cl}]_j := Bern([\boldsymbol{x}_c]_j)$
    end for
    $\boldsymbol{s}_l$ is the stacking of T amounts of the vector $\boldsymbol{s}_{cl}$
    $[\boldsymbol{c}]_l := \boldsymbol{z}^T diag(\boldsymbol{s}_l)\boldsymbol{z}^{(1)}$
end for
$\boldsymbol{S}$ is the matrix with columns $\boldsymbol{s}_l$ for all $l \in \{0, 1, 2, \ldots c\}$
Sort the elements of $\boldsymbol{c}$ in descending order.
Sort the columns of $\boldsymbol{S}$ with the same order that is used to sort $\boldsymbol{c}$.
hit:=0
i:=1
while hit=0 **do**
    if $[\boldsymbol{S}]_{i*} \in \mathcal{C}_1$ **then**
        hit:=1
    else
        i:=i+1
    end if
end while
$\mathbf{1}^*_{\mathcal{A}}$ is equal to the 'unstacked' form of vector $[\boldsymbol{S}]_{i*}$
return $\mathbf{1}^*_{\mathcal{A}}$

Algorithm 6 Rounding Algorithm for M.3

Input: $(\boldsymbol{x}_c, \beta, i) : \boldsymbol{x}_c \in [0,1]^N, \beta > 0, i \in \{1, 2 \ldots T\}$
output: $\mathbf{1}^*_{\mathcal{A}}$
β is defined as the cardinality constraint for $\mathcal{C}_1$ and i is the time.
for l ←1 **to** c **do**
    for j ←1 **to** N **do**
        $[\boldsymbol{s}_{cl}]_j := Bern([\boldsymbol{x}]_j)$
    end for
    $[\boldsymbol{c}]_l := \boldsymbol{z}_l^T diag(\boldsymbol{s}_{cl}) \boldsymbol{z}_l$
end for
$\boldsymbol{S}$ is the matrix with columns $\boldsymbol{s}_{cl}$ for all $l \in \{0, 1, 2, \ldots c\}$
Sort the elements of $\boldsymbol{c}$ in descending order.
Sort the columns of $\boldsymbol{S}$ with the same order that is used to sort $\boldsymbol{c}$.
hit=0
i=1
while hit=0 **do**
    if $[\boldsymbol{S}]_{i*} \in \mathcal{C}_1$ **then**
        hit:=1
    else
        i:=i+1
    end if
end while
$\mathbf{1}^*_{\mathcal{A}} := [\boldsymbol{S}]_{i*}$
return $\mathbf{1}^*_{\mathcal{A}}$

C

Explanation Of the Distributions For the Lasso-based Method

For an unshifted χ^2 distribution with degrees of freedom equal to 1, the pdf looks as follows.

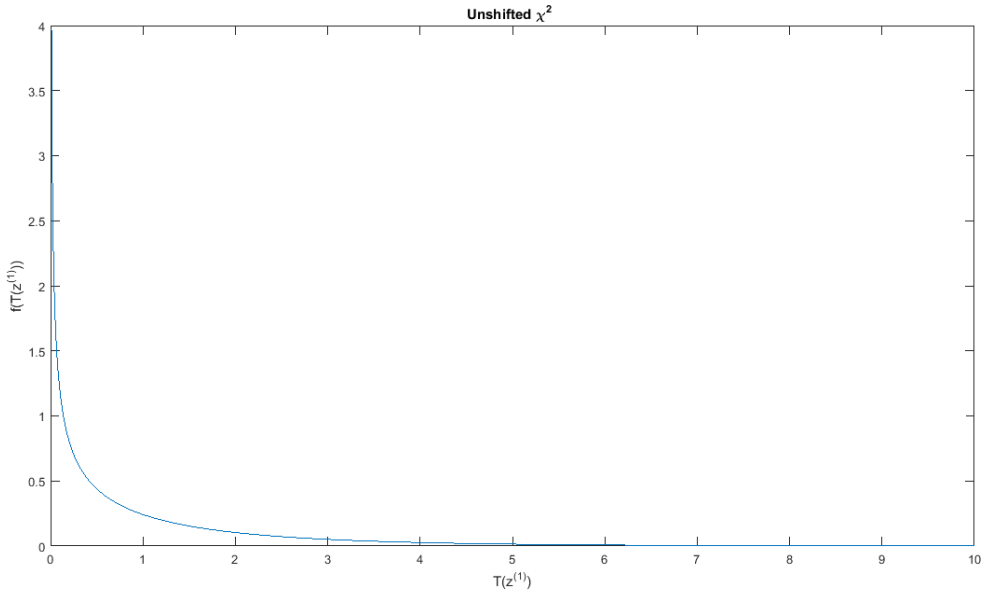


Figure C.1: Unshifted χ^2 distribution with $dof = 1$

In case that we subtract a positive value $\frac{\lambda^2}{\sigma^2}$ from the distribution, the PDF will look as follows

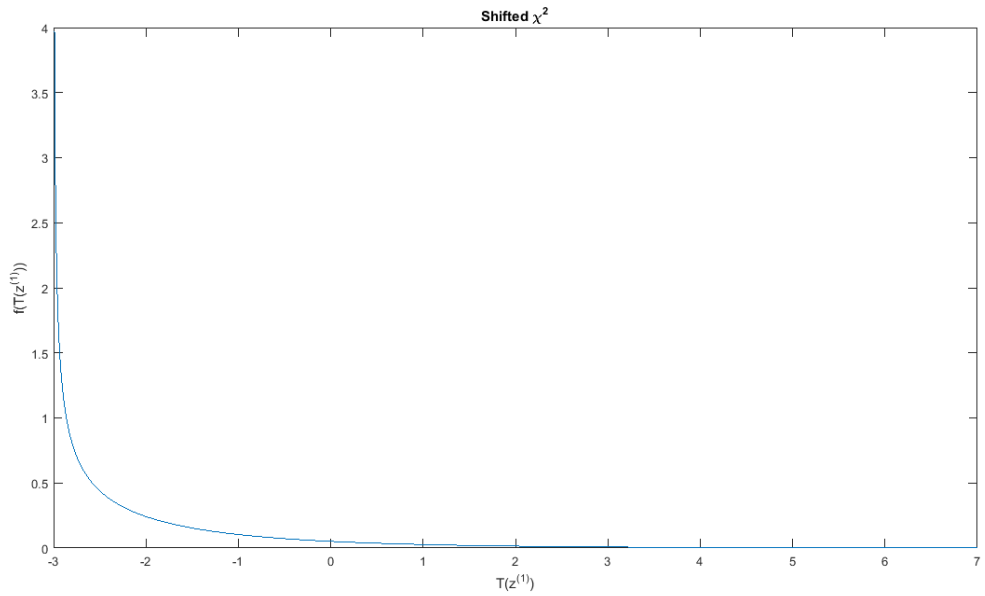


Figure C.2: Shifted χ^2 distribution with $dof = 1$

A shift to the left can be observed, i.e., $f_{\chi^2}(T([\boldsymbol{z}]_i) + \frac{\lambda^2}{\sigma^2})$.
Now, when we use the $(.)_+$ operator, the domain $T([\boldsymbol{z}]_i) < 0$ only returns values that are equal to 0. Thus, the probability for the occurrence of $\frac{[\boldsymbol{z}]_i^2 - [\boldsymbol{\lambda}]}{\sigma^2} < 0$, is 'added up' to the probability that $\frac{[\boldsymbol{z}]_i^2 - [\boldsymbol{\lambda}]}{\sigma^2} = 0$, for $\frac{([\boldsymbol{z}]_i^2 - [\boldsymbol{\lambda}])_+}{\sigma^2} = 0$. The probability value of $\frac{[\boldsymbol{z}]_i^2 - [\boldsymbol{\lambda}]}{\sigma^2} < 0$ is calculated with help of the cumulative distribution function (cdf), i.e., $F_{\chi^2}(\frac{\lambda^2}{\sigma^2})$.

D

Constraint Explanation For the Derivation of Threshold Function In the Future Work

By taking the log-likelihood of the ratio test, we get first

$$T(\boldsymbol{y}) = (\boldsymbol{y} - \boldsymbol{s}_1 - \boldsymbol{o})^T(\boldsymbol{y} - \boldsymbol{s}_1 - \boldsymbol{o}) - (\boldsymbol{y} - \boldsymbol{s}_0)^T(\boldsymbol{y} - \boldsymbol{s}_0) \tag{D.1}$$

With the help of simplification we get:

$$T(\boldsymbol{y}) = (\boldsymbol{s}_1 + \boldsymbol{o})^T(\boldsymbol{s}_1 + \boldsymbol{o}) - (\boldsymbol{s}_0)^T(\boldsymbol{s}_0) - 2\boldsymbol{y}^T(\boldsymbol{s}_1 + \boldsymbol{o} - \boldsymbol{s}_0) \tag{D.2}$$

We can now simplify this expression so that all the terms that are not dependent on $\boldsymbol{y}$ are dismissed

$$T(\boldsymbol{y}) = \boldsymbol{y}^T(\boldsymbol{s}_1 + \boldsymbol{o} - \boldsymbol{s}_0) \tag{D.3}$$

E

Model 1 Log-Likelihood Ratio Test Derivation for colored noise

First, we calculate c by taking the derivative of the mean square error and forcing it to zero.

$$\frac{\partial(\boldsymbol{y}-c\mathbf{1}_{\tilde{A}})^{T}\boldsymbol{C}^{-1}(\boldsymbol{y}-c\mathbf{1}_{\tilde{A}})}{\partial c}=-\boldsymbol{z}^{T}\boldsymbol{C}^{-1}\mathbf{1}_{\tilde{A}}-\mathbf{1}_{\tilde{A}}\boldsymbol{C}^{-1}\boldsymbol{z}+2c\mathbf{1}_{\tilde{A}}^{T}\boldsymbol{C}^{-1}\mathbf{1}_{\tilde{A}}=0 \tag{E.1}$$

Thus c can be formulated as:

$$c=\frac{\mathbf{1}_{\tilde{A}}{}^{T}\boldsymbol{C}^{-1}\boldsymbol{z}}{\mathbf{1}_{\tilde{A}}{}^{T}\boldsymbol{C}^{-1}\mathbf{1}_{\tilde{A}}} \tag{E.2}$$

Now, by using the standard derived notation for the Log-likelihood ratio for the classical white Gaussian noise, signal detection problem(7.3). We get the following expression by substitution:

$$T(\boldsymbol{z}^{(t+1)})=c\mathbf{1}_{\tilde{A}}^{T}\boldsymbol{C}^{(-1)}\boldsymbol{z}-\frac{1}{2}c\mathbf{1}_{\tilde{A}}{}^{T}\boldsymbol{C}^{(-1)}c\mathbf{1}_{\tilde{A}} \tag{E.3}$$

Substituting c with the derived equation:

$$T(\boldsymbol{z}^{(t+1)})=\frac{\mathbf{1}_{\tilde{A}}{}^{T}\boldsymbol{C}^{-1}\boldsymbol{z}}{\mathbf{1}_{\tilde{A}}{}^{T}\boldsymbol{C}^{-1}\mathbf{1}_{\tilde{A}}}\mathbf{1}_{\tilde{A}}^{T}\boldsymbol{C}^{(-1)}\boldsymbol{z}-\frac{1}{2}\frac{\mathbf{1}_{\tilde{A}}{}^{T}\boldsymbol{C}^{-1}\boldsymbol{z}}{\mathbf{1}_{\tilde{A}}{}^{T}\boldsymbol{C}^{-1}\mathbf{1}_{\tilde{A}}}\mathbf{1}_{\tilde{A}}{}^{T}\boldsymbol{C}^{(-1)}\frac{\mathbf{1}_{\tilde{A}}{}^{T}\boldsymbol{C}^{-1}\boldsymbol{z}}{\mathbf{1}_{\tilde{A}}{}^{T}\boldsymbol{C}^{-1}\mathbf{1}_{\tilde{A}}}\mathbf{1}_{\tilde{A}} \tag{E.4}$$

This can be further simplified to:

$$T(\boldsymbol{z}^{(t+1)})=\frac{(\mathbf{1}_{\tilde{A}}{}^{T}\boldsymbol{C}^{-1}\boldsymbol{z})^{2}}{\mathbf{1}_{\tilde{A}}{}^{T}\boldsymbol{C}^{-1}\mathbf{1}_{\tilde{A}}}\underset{H_0}{\overset{H_1}{\gtrless}}\gamma \tag{E.5}$$

And the final form is:

$$T(\boldsymbol{z}^{(t+1)})=\frac{|\mathbf{1}_{\tilde{A}}{}^{T}\boldsymbol{C}^{-1}\boldsymbol{z}|}{\sqrt{\mathbf{1}_{\tilde{A}}{}^{T}\boldsymbol{C}^{-1}\mathbf{1}_{\tilde{A}}}}\underset{H_0}{\overset{H_1}{\gtrless}}\gamma' \tag{E.6}$$

Bibliography

[1] Ghada El-Kabbany and Mohamed Rasslan. Security issues in distributed computing system models. *Security Solutions for Hyperconnectivity and the Internet of Things, Advances in Information Security, Privacy, and Ethics (AISPE)*, 36:211–259, 08 2016.

[2] R. Gentz, S. X. Wu, H. Wai, A. Scaglione, and A. Leshem. Data injection attacks in randomized gossiping. *IEEE Transactions on Signal and Information Processing over Networks*, 2(4):523–538, 2016.

[3] Olle Abrahamsson, Danyo Danev, and Erik G. Larsson. Opinion dynamics with random actions and a stubborn agent. 2019.

[4] Nikhil Ravi and Anna Scaglione. Detection and isolation of adversaries in decentralized optimization for non-strongly convex objectives. 10 2019.

[5] Or Shalom, Amir Leshem, and Anna Scaglione. Localization of data injection attacks on distributed m-estimation. *2019 IEEE Data Science Workshop (DSW)*, pages 22–26, 2019.

[6] Yangyue Feng, Chiara Foglietta, Alessio Baiocco, Stefano Panzieri, and Stephen Wolthusen. Malicious false data injection in hierarchical electric power grid state estimation systems. pages 183–192, 01 2013.

[7] Arpan Chattopadhyay and Urbashi Mitra. Security against false data injection attack in cyber-physical systems. *IEEE Transactions on Control of Network Systems*, PP:1–1, 07 2019.

[8] Stefano Vissicchio, Olivier Tilmans, Laurent Vanbever, and Jennifer Rexford. Central control over distributed routing. *ACM SIGCOMM Computer Communication Review*, 45:43–56, 08 2015.

[9] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovačević. Signal recovery on graphs: Variation minimization. *IEEE Transactions on Signal Processing*, 63(17):4609–4624, 2015.

[10] A. Gavili and X. Zhang. On the shift operator, graph frequency, and optimal filtering in graph signal processing. *IEEE Transactions on Signal Processing*, 65(23):6303–6318, 2017.

[11] P. A. Gagniuc. Markov chains: From theory to implementation and experimentation. 2017.

[12] Alexandr Andoni, Jiecao Chen, Robert Krauthgamer, Bo Qin, David Woodruff, and Qin Zhang. On sketching quadratic forms. pages 311–319, 01 2016.

[13] Jerzy Neyman and Egon Sharpe Pearson. On the Problem of the Most Efficient Tests of Statistical Hypotheses. *Phil. Trans. Roy. Soc. Lond. A*, 231(694-706):289–337, 1933.

[14] Arpan Chattopadhyay and Urbashi Mitra. Security against false data injection attack in cyber-physical systems. *IEEE Transactions on Control of Network Systems*, PP:1–1, 07 2019.

[15] S. Li, Y. Yılmaz, and X. Wang. Quickest detection of false data injection attack in wide-area smart grids. *IEEE Transactions on Smart Grid*, 6(6):2725–2735, 2015.

[16] L. Lei, W. Yang, C. Yang, and H. Shi. False data injection attack on consensus□based distributed estimation. *International Journal of Robust and Nonlinear Control*, 27, 10 2016.

[17] A. Adler, M. Elad, Y. Hel-Or, and E. Rivlin. Sparse coding with anomaly detection. pages 1–6, 2013.

[18] Xiaohang Li, Chunlin Song, Shaokun Liu, and Jinlong Li. Adaptive compensation control under constant false data injection attack. *IOP Conference Series: Materials Science and Engineering*, 782:032072, 04 2020.

[19] B. Tang, Jun Yan, S. Kay, and H. He. Detection of false data injection attacks in smart grid under colored gaussian noise. pages 172–179, 2016.

[20] Gilbert Strang. *Linear algebra and its applications*. Thomson, Brooks/Cole, Belmont, CA, 2006.

[21] S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall, 1997.

[22] Trevor Park and George Casella. The bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686, 2008.

[23] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[24] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005.

[25] S.M. Kay. *Fundamentals of Statistical Signal Processing: Detection theory*. Fundamentals of Statistical Signal Processing. PTR Prentice-Hall, 1993.

[26] Michail Tsagris, Christina Beneki, and Hossein Hassani. On the folded normal distribution. *Mathematics*, 2, 02 2014.

[27] Francis Bach. Learning with submodular functions: A convex optimization perspective. 2011.

[28] Stephen Boyd and Lieven Vandenberghe. Convex optimization. 2004.

[29] Werner Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7):492–498, 1967.

[30] Franz Rendl. Semidefinite relaxations for integer programming. pages 687–726, 2010.

[31] Stephen Boyd and Lieven Vandenberghe. Semidefinite programming relaxations of non-convex problems in control and combinatorial optimization. 1997.

[32] Jean Gallier. Notes on the schur complement. 12 2010.

[33] Lin Xiao and S. Boyd. Fast linear iterations for distributed averaging. 5:4997–5002 Vol.5, 2003.

[34] Nathanaël Perraudin, Johan Paratte, David Shuman, Lionel Martin, Vassilis Kalofolias, Pierre Vandergheynst, and David K. Hammond. GSPBOX: A toolbox for signal processing on graphs. *ArXiv e-prints*, August 2014.