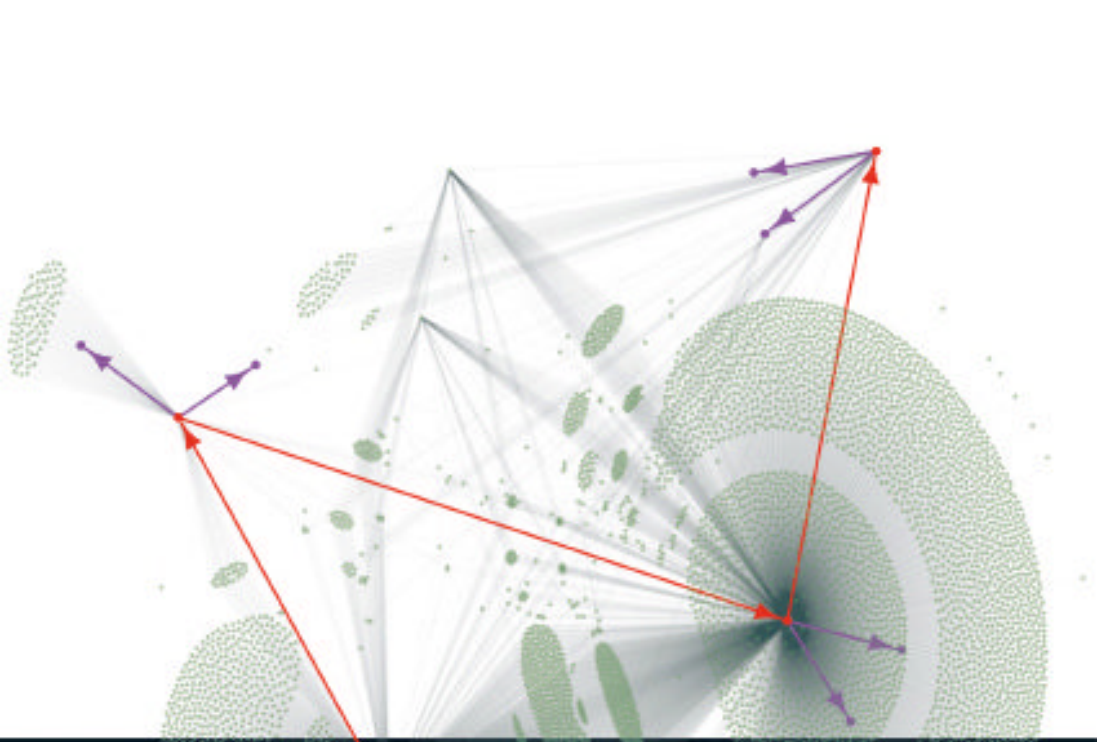


DATA ANALYSIS FOR NETWORK CYBER-SECURITY

Niall Adams • Nicholas Heard *editors*

Imperial College Press

DATA ANALYSIS FOR NETWORK CYBER-SECURITY

This page intentionally left blank

DATA ANALYSIS FOR NETWORK CYBER-SECURITY

editors

Niall Adams • **Nicholas Heard**

Imperial College London
Heilbronn Institute for Mathematical Research, University of Bristol

ICP

Imperial College Press

Published by

Imperial College Press
57 Shelton Street
Covent Garden
London WC2H 9HE

Distributed by

World Scientific Publishing Co. Pte. Ltd.

5 Toh Tuck Link, Singapore 596224

USA office: 27 Warren Street, Suite 401-402, Hackensack, NJ 07601

UK office: 57 Shelton Street, Covent Garden, London WC2H 9HE

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

DATA ANALYSIS FOR NETWORK CYBER-SECURITY

ISBN 978-1-78326-374-5

Typeset by Stallion Press
Email: enquiries@stallionpress.com

Printed in Singapore

Preface

The contents of this volume are contributions from invited speakers at a workshop entitled "Data Analysis for Cyber-Security", hosted by the University of Bristol in March 2013. We are grateful for the generous support of the Heilbronn Institute for Mathematical Research, an academic research unit of the Universisty of Bristol with interests related to cyber-security.

Cyber-security – the task of defending computers and people from electronic attack – is a pressing concern. For example, a report sponsored by the UK government in 2012 estimated the cost of cyber-attack to the UK economy as £29 billion. This cost is attributed to various types of attack, including extortion, fiscal fraud and identity theft. Notably, the largest category, intellectual property theft, accounted for around £9 billion. The scale of cyber-attack provoked the UK government to highlight cyber-security as a top priority for national security in 2013. From a UK point of view, based on recent figures, the problem is increasing: 78% of large organisations were subject to external attack in 2012, up from 73% in the previous year, while 63% of small business were subject to such attack over the same period, an increase of 41% on the previous year.

Cyber-security is a broad discipline, covering a range of academic disciplines including computer science, computer and network architecture, and statistics. This volume is concerned with *network* cyber-security, and particularly, analysis of data that are observed in relation to a network of either computers or people. As an exemplar, consider an institutional computer network, in which communicating devices (computers, printers, etc.) are *nodes* and communications between devices are *events* that occur on *edges* between nodes. Numerous types of cyber-attack have been observed in this context. A variety of such attacks are well described in Davidoff and Ham (2012) from the point of view of network forensics.

There are a number of problems that can be addressed by analysing network data. A primary example is constructing anomaly detection methods to identify when unusual traffic occurs. These complement signature-based methods, such as those embodied for example in the Snort intrusion

detection system (Caswell *et al.*, 2007). We believe the next generation of detection tools will have to utilise more information about past and present traffic behaviour, particularly with respect to temporal aspects. A particular advantage of anomaly detection methods over rule-based approaches is their potential to detect new, so-called *zero-day*, attacks.

There are significant challenges when addressing network data analysis problems. In the context of cyber-security, data sets are typically big, consisting of a large number of nodes and a great volume of traffic on existing edges. This alone raises significant computational challenges. If anomaly detection is the objective, then timeliness becomes an issue, and the *velocity* of the traffic on edges becomes an important factor. The combination of volume and velocity makes much network cyber-security a "big-data" problem. From a statistical or machine learning point of view, many cyber-security problems are *unsupervised*, which raises generic problems of model selection and control of hyperparameters and decision boundaries. These data analysis problems are generally exacerbated by increases in the volume, velocity and heterogeneity of network data. The precise timing of events on edges is a more subtle aspect that is only recently being seriously explored. This latter aspect is extensively addressed in this volume.

The above discussion is intended to make the case that cyber-security is both an important and interesting research problem. Some of the research opportunities are discussed in more detail in Meza *et al.* (2009). The chapters in this volume provide a view of this exciting and diverse area. To begin, Patrick Wolfe and Benjamin Olding give an introduction to the problem of statistical inference on graphs, and make first steps toward formal inferential procedures.

Alex Tartakovsky considers the problem of quickest change detection in the context of statistical anomaly detection. A key concern is to minimise the detection delay, an aspect of great importance in many practical network cyber problems. This chapter features a hybrid anomaly-spectral-signature-based system useful for efficient traffic filtering.

Joshua Neil and co-authors are concerned with aspects of network traffic that are localised in both time and graph space. In particular, they develop a scan statistic-based methodology for finding connected subgraphs that are locally connected in time and which have deviated from historic behaviour. The methodology is illustrated on large-scale network traffic data.

Summet Dua and Pradeep Chowriappa address situational awareness by considering user sentiment in social media. Such sentiments can provide

indicators and precursors for cyber-threat. A novel data-mining approach is proposed to identify aspects that influence the dynamics of the network over time.

Céline Lévy-Leduc considers both centralised and decentralised network versions of change detection for network traffic data. This includes both dimension reduction to simplify network traffic data to a manageable representation and nonparametric change detection adapted for censoring.

Finally, Nick Heard and Melissa Turcotte develop Bayesian anomaly detection methodology suited to the computational demands of large networks. A screening methodology, based on simple node- and edge-based statistical models is developed. While these models are relatively simple, there are numerous technical details addressed to enable their successful deployment.

Niall Adams, Nick Heard

References

Caswell, B., Beale, J. and Baker, A. (2007). *Snort Intrusion Detection and Prevention Toolkit* (Syngress Media).

Davidoff, S. and Ham, J. (2012). *Network Forensics: Tracking Hackers Through Cyberspace* (Prentice Hall, Upper Saddle River, NJ).

Meza, J., Campbell, S. and Bailey, D. (2009). Mathematical and statistical opportunities in cyber security, CoRR. Available at: http://arxiv.org/abs/0904.1616.

This page intentionally left blank

Contents

This page intentionally left blank

Chapter 1

Inference for Graphs and Networks: Adapting Classical Tools to Modern Data

Benjamin P. Olding* and Patrick J. Wolfe†

*Jana Mobile Inc.
883 Boylston Street
Boston, MA 02116, USA
olding@post.harvard.edu

†University College London
London, WC1E 6BT, United Kingdom
p.wolfe@ucl.ac.uk

Graphs and networks provide a canonical representation of relational data, with massive network data sets becoming increasingly prevalent across a variety of scientific fields. Although tools from mathematics and computer science have been eagerly adopted by practitioners in the service of network inference, they do not yet comprise a unified and coherent framework for the statistical analysis of large-scale network data. This chapter serves as both an introduction to the topic and a first step toward formal inference procedures. We develop and illustrate our arguments using the example of hypothesis testing for network structure. We invoke a generalized likelihood ratio framework and use it to highlight the growing number of topics in this area that require strong contributions from statistical science. We frame our discussion in the context of previous work from across a variety of disciplines, and conclude by outlining fundamental statistical challenges whose solutions will in turn serve to advance the science of network inference.

1.1. Introduction

Graphs and networks have long been a subject of significant mathematical and scientific interest, deemed worthy of study for their own sake and often associated with scientific data. However, a diverse and rapidly growing set of contemporary applications is fast giving rise to massive networks that *themselves* comprise the data set of interest – and to analyze these network data, practitioners in turn require analogs to classical inferential procedures.

While past decades have witnessed a variety of advances in the treatment of graphs and networks as combinatoric or algebraic objects, corresponding advances in formal data analysis have largely failed to keep pace. Indeed, the development of a successful framework for the statistical analysis of network data requires the repurposing of existing models and algorithms for the specific purpose of inference. In this chapter, we pose the question of how modern statistical science can best rise to this challenge as well as benefit from the many opportunities it presents. We provide first steps toward formal network inference procedures through the introduction of new tests for network structure, and employ concrete examples throughout that serve to highlight the need for additional research contributions in this burgeoning area.

1.1.1. *Modern network data sets*

Though once primarily the domain of social scientists, a view of networks as data objects is now of interest to researchers in areas spanning biology, finance, engineering, and library science, among others. Newman (2003) provides an extensive review of modern network data sets; other examples of note include mobile phone records, which link customers according to their phone calls (Eagle *et al.*, 2008); the internet, including both web pages connected by hyperlinks (Adamic and Huberman, 2000) and peer-to-peer networks (Stutzbach *et al.*, 2006); electrical power networks, in which local grids are physically connected by long-distance transmission lines (Watts and Strogatz, 1998); and publication networks, where citations provide explicit links between authors (de Solla Price, 1965).

At the same time, other scientific fields are beginning to reinterpret traditional data sets as networks, in order to better understand, summarize, and visualize relationships amongst very large numbers of observations. Examples include protein–protein interaction networks, with isolated pairs of proteins deemed connected if an experiment suggests that they interact (Batada *et al.*, 2006); online financial transactions, whereupon items are considered to be linked if they are typically purchased together (Jin *et al.*, 2007); food webs, with species linked by predator–prey relationships (Dunne *et al.*, 2002); and spatial data sets (Thompson, 2006; Ceyhan *et al.*, 2007).

1.1.2. *Organization and aims of the chapter*

The above examples attest both to the wide variety of networks encountered in contemporary applications, as well as the multiple expanding literatures

on their analysis. In this chapter, we focus on introducing the subject from first principles and framing key inferential questions. We begin with a discussion of relational data in Section 1.2, and introduce notation to make the connection to networks precise. We discuss model specification and inference in Section 1.3, by way of concrete definitions and examples. We introduce new ideas for detecting network structure in Section 1.4, and apply them to data analysis by way of formal testing procedures. In Section 1.5 we discuss open problems and future challenges for large-scale network inference in the key areas of model elicitation, approximate fitting procedures, and issues of data sampling. In a concluding appendix we provide a more thorough introduction to the current literature, highlighting contributions to the field from statistics as well as a variety of other disciplines.

1.2. Networks as Relational Data

We begin our analysis by making explicit the connection between networks and relational data. In contrast to data sets that may that arise from pairwise distances or affinities of points in space or time, many modern network data sets are massive, high-dimensional, and non-Euclidean in their structure. We therefore require a way to describe these data other than through purely pictorial or tabular representations – and the notion of cataloging the pairwise relationships that comprise them, with which we begin our analysis, is natural.

1.2.1. *Relational data matrices and covariates*

Graphs provide a canonical representation of relational data as follows: Given n entities or objects of interest with pairs indexed by (i, j), we write $i \sim j$ if the ith and jth entities are related, and $i \nsim j$ otherwise. These assignments may be expressed by way of an $n \times n$ adjacency matrix $\boldsymbol{A}$, whose entries $\{A_{ij}\}$ are nonzero if and only if $i \sim j$. While both the structure of $\boldsymbol{A}$ and the field over which its entries are defined depend on the application or specific data set, a natural connection to graph theory emerges in which entities are represented by vertices, and relations by edges; we adopt the informal but more suggestive descriptors "node" and "link," respectively. The *degree* of the ith node is in turn defined as $\sum_{j=1}^{n} A_{ij}$.

In addition, the data matrix $\boldsymbol{A}$ is often accompanied by covariates $c(i)$ associated with each node, $i \in \{1, 2, \ldots, n\}$. Example 1.1 below illustrates a case in which these covariates take the form of binary categorical variables. We shall refer back to these illustrative data throughout Sections 1.2

and 1.3, and later in Section 1.4 will consider a related real-world example: the social network recorded by Zachary (1977), in which nodes represent members of a collegiate karate club and links represent friendships, with covariates indicating a subsequent split of the club into two disjoint groups.

Example 1.1 (Network Data Set). *As an example data set, consider the ten-node network defined by data matrix* $\boldsymbol{A}$ *and covariate vector* c *as*

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}; \quad c = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}.$$

A visualization of the corresponding network is shown in Figure 1.1; however, note that as no geometric structure is implied by the data set itself, a pictorial rendering such as this is arbitrary and non-unique.

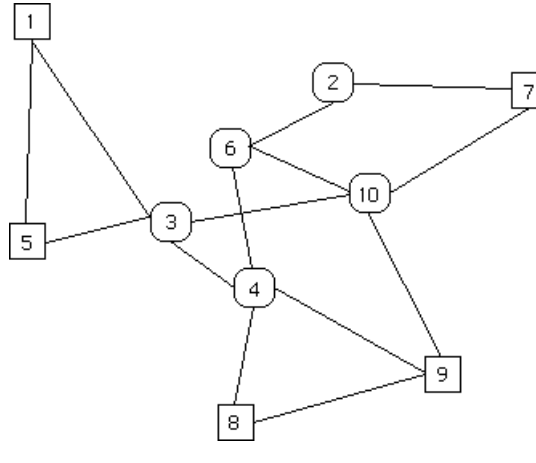


Fig. 1.1. The network data of Example 1.1, with nodes indexed by number and binary categorical covariate values by shape. Note that no Euclidean embedding accompanies the data, making visualization a challenging task for large-scale networks.

In Example 1.1, categorical covariates $c(i)$, $i \in \{1, 2, \ldots, n\}$ are given; however, in network data sets of practical interest, these covariates may well be latent. This in turn gives rise to many of the principal questions of network inference – in contrast to the traditional setting of relational data. Therefore, the issues of network modeling which arise tend to be distinct; as such, classical approaches (e.g., contingency table tests) are directly applicable to network data only in very restricted circumstances.

1.2.2. *Networks as distinct from relational data*

The main distinction between modern-day network data and classical relational data lies in the requisite computational complexity for inference. Indeed, the computational requirements of large-scale network data sets are substantial. With n nodes we associate $\binom{n}{2} = n(n-1)/2$ symmetric relations; beyond this quadratic scaling, latent covariates give rise to a variety of combinatorial expressions in n. Viewed in this light, methods to determine relationships amongst *subsets* of nodes can serve as an important tool to "coarsen" network data. In addition to providing a lower-dimensional summary of the data, such methods can serve to increase the computational efficiency of subsequent inference procedures by enabling data reduction and smoothing. The general approach is thus similar to modern techniques for high-dimensional Euclidean data, and indeed may be viewed as a clustering of nodes into groups.

From a statistical viewpoint, this notion of subset relations can be conveniently described by a k-ary categorical covariate, with k specifying the (potentially latent) model order. By incorporating such a covariate into the probability model for the data adjacency matrix $\boldsymbol{A}$, the "structure" of the network can be directly tested if this covariate is observed, or instead inferred if latent. It is easily seen that the cardinality of the resultant model space is exponential in the number of nodes n; even if the category sizes themselves are given, we may still face a combinatorial inference problem. Thus, even a straightforwardly posed hypothesis test for a relatively simple model can easily lead to cases where exact inference procedures are intractable.

1.3. Model Specification and Inference

Fields such as probability, graph theory, and computer science have each posited specific models which can be applied to network data; however,

when appealing to the existing literature, it is often the case that neither the models nor the analysis tools put forward in these contexts have been developed specifically for inference. In this section, we introduce two basic network models and relate them to classical statistics. The first such model consists of nodes whose degrees are identically distributed, whereas the second implies group structure via latent categorical covariates. Inferring relationships amongst groups of nodes from data in turn requires the standard tools of statistics, including parameter estimation and hypothesis testing. We provide examples of such procedures below, illustrating their computational complexity, and introduce corresponding notions of approximate inference.

1.3.1. *Erdös–Rényi: A first illustrative example*

We begin by considering one of the simplest possible models from random graph theory, attributed to Erdös and Rényi (1959) and Gilbert (1959), and consisting of pairwise links that are generated independently with probability p. Under this model, all nodes have identically distributed degrees; it is hence appropriate to describe instances in which no group structure (by way of categorical covariates) is present. In turn, we shall contrast this with an explicit model for structure below.

Adapted to the task of modeling undirected network data, the Erdös–Rényi model may be expressed as a sequence of $\binom{n}{2}$ Bernoulli trials corresponding to off-diagonal elements of the adjacency matrix $\boldsymbol{A}$.

Definition 1.1 (Erdös–Rényi Model). *Let $n > 1$ be integral and fix some $p \in [0,1]$. The Erdös–Rényi random graph model corresponds to matrices $\boldsymbol{A} \in \{0,1\}^{n \times n}$ defined element-wise as*

$$\forall\, i, j \in \{1, 2, \ldots, n\} : i < j,\ A_{ij} \overset{iid}{\sim} \text{Bernoulli}(p); \quad A_{ji} = A_{ij},\ A_{ii} = 0.$$

Erdös–Rényi thus provides a one-parameter model yielding independent and identically distributed binary random variables representing the absence or presence of pairwise links between nodes; as this binary relation is symmetric, we take $A_{ji} = A_{ij}$. The additional stipulation $A_{ii} = 0$ for all i implies that our relation is also irreflexive; in the language of graph theory, the corresponding (undirected, unweighted) graph is said to be *simple*, as it exhibits neither multiple edges nor self-loops. The event $i \sim j$ is thus a Bernoulli(p) random variable for all $i \neq j$, and it follows that the degree $\sum_{j=1}^{n} A_{ij}$ of each network node is a Binomial($n-1, p$) random variable.

Fitting the parameter p is straightforward; the maximum likelihood estimator (MLE) corresponds to the sample proportion of observed links:

$$\widehat{p} := \frac{1}{\binom{n}{2}} \sum_{i<j} A_{ij} = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij}.$$

Example 1.1, for instance, yields $\widehat{p} = 14/45$.

Given a relational data set of interest, we can test the agreement of data in $\boldsymbol{A}$ with this model by employing an appropriately selected test statistic. If we wish to test this uniformly generic model with respect to the notion of network structure, we may explicitly define an alternate model and appeal to the classical Neyman–Pearson testing framework.

In this vein, the Erdös–Rényi model can be generalized in a natural way to capture the notion of local rather than global exchangeability: we simply allow Bernoulli parameters to depend on k-ary categorical covariates $c(i)$ associated with each node $i \in \{1, 2, \ldots, n\}$, where the $k \leq n$ categories represent groupings of nodes. Formally we define

$$c \in \mathbb{Z}_k^n; \quad c(i) : \{1, 2, \ldots, n\} \mapsto \mathbb{Z}_k,$$

and a set of $\binom{k+1}{2}$ distinct Bernoulli parameters governing link probabilities within and between these categories, arranged into a $k \times k$ symmetric matrix and indexed as $p_{c(i)c(j)}$ for $i, j \in \{1, 2, \ldots, n\}$.

In the case of binary categorical covariates, we immediately obtain a formulation of Holland and Leinhardt (1981), the simplest example of a so-called *stochastic block model*. In this network model, pairwise links between nodes correspond again to Bernoulli trials, but with a parameter chosen from the set $\{p_{00}, p_{01}, p_{11}\}$ according to binary categorical covariates associated with the nodes in question.

Definition 1.2 (Simple Stochastic Block Model). *Let $c \in \{0,1\}^n$ be a binary n-vector for some integer $n > 1$, and fix parameters $p_{00}, p_{01}, p_{11} \in [0,1]$. Set $p_{10} = p_{01}$; the model then corresponds to matrices $\boldsymbol{A} \in \{0,1\}^{n \times n}$ defined element-wise as*

$$\forall\, i, j \in \{1, 2, \ldots, n\} : i < j,\ A_{ij} \sim \text{Bernoulli}(p_{c(i)c(j)});$$
$$A_{ji} = A_{ij},\ A_{ii} = 0.$$

If the vector of covariates c is given, then finding the maximum-likelihood parameter estimates $\{\widehat{p}_{00}, \widehat{p}_{01}, \widehat{p}_{11}\}$ is trivial after a re-ordering of

nodes via permutation similarity: For any $n \times n$ permutation matrix $\boldsymbol{\Pi}$, the adjacency matrices $\boldsymbol{A}$ and $\boldsymbol{\Pi A \Pi'}$ represent isomorphic graphs, the latter featuring permuted rows and columns of the former. If $\boldsymbol{\Pi}$ re-indexes nodes according to their categorical groupings, then we may define a conformal partition

$$\boldsymbol{\Pi A \Pi'} = \begin{pmatrix} \boldsymbol{A_{00}} & \boldsymbol{A_{01}} \\ \boldsymbol{A'_{01}} & \boldsymbol{A_{11}} \end{pmatrix}$$

that respects this ordering, such that exchangeability is preserved within – but not across – submatrices $\boldsymbol{A_{00}}$ and $\boldsymbol{A_{11}}$. We may then simply compute sample proportions corresponding to each submatrix $\{\boldsymbol{A_{00}}, \boldsymbol{A_{01}}, \boldsymbol{A_{11}}\}$ to yield $\{\widehat{p}_{00}, \widehat{p}_{01}, \widehat{p}_{11}\}$.

Note that by construction, submatrices $\boldsymbol{A_{00}}$ and $\boldsymbol{A_{11}}$ yield subgraphs that are themselves Erdös–Rényi, and are said to be *induced* by the two respective groups of categorical covariates. Nonzero entries of $\boldsymbol{A_{01}}$ are said to comprise the *edge boundary* between these two induced subgraphs; indeed, the matrix obtained by setting all entries of $\boldsymbol{A_{00}}$ and $\boldsymbol{A_{11}}$ to zero yields in turn a *bipartite* graph whose vertices can be partitioned according to their binary covariate values.

The following example illustrates these concepts using the simulated data of Example 1.1.

Example 1.2 (Similarity and Subgraphs). *Let the ten-node network of Example 1.1 be subject to an isomorphism that re-orders nodes according to the two groups defined by their binary covariate values, and define the permutation-similar data matrix* $\widetilde{\boldsymbol{A}}$ *and permuted covariate vector* $\tilde{c}$ *as follows*:

$$\widetilde{\boldsymbol{A}} = \left(\begin{array}{ccccc|ccccc} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right) = \begin{pmatrix} \widetilde{\boldsymbol{A}}_{\boldsymbol{00}} & \widetilde{\boldsymbol{A}}_{\boldsymbol{01}} \\ \widetilde{\boldsymbol{A}}'_{\boldsymbol{01}} & \widetilde{\boldsymbol{A}}_{\boldsymbol{11}} \end{pmatrix}; \quad \tilde{c} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

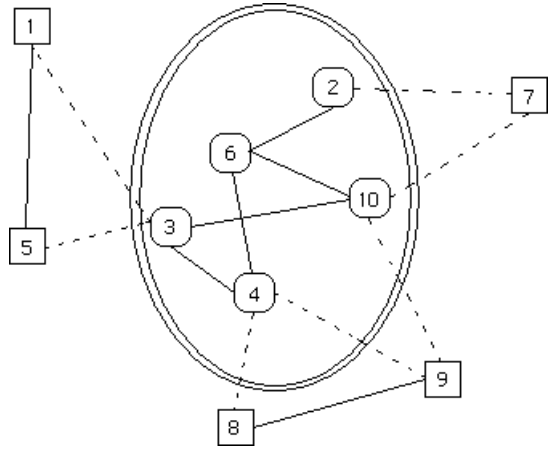


Fig. 1.2. Subgraphs based on the binary covariates of Example 1.1, again represented graphically by node shape. The conformal partition of Example 1.2 implies two induced subgraphs: solid lines inside the ellipse are links represented in submatrix $\widetilde{\boldsymbol{A}}_{\mathbf{00}}$, while those outside it appear as links in $\widetilde{\boldsymbol{A}}_{\mathbf{11}}$. The remaining links, shown as dashed lines, correspond to values of 1 in submatrix $\widetilde{\boldsymbol{A}}_{\mathbf{01}}$ and comprise the associated edge boundary.

Figure 1.2 illustrates the corresponding subgraphs using the visualization of Figure 1.1; *assuming a simple stochastic block model in turn leads to the following maximum-likelihood parameter estimates*:

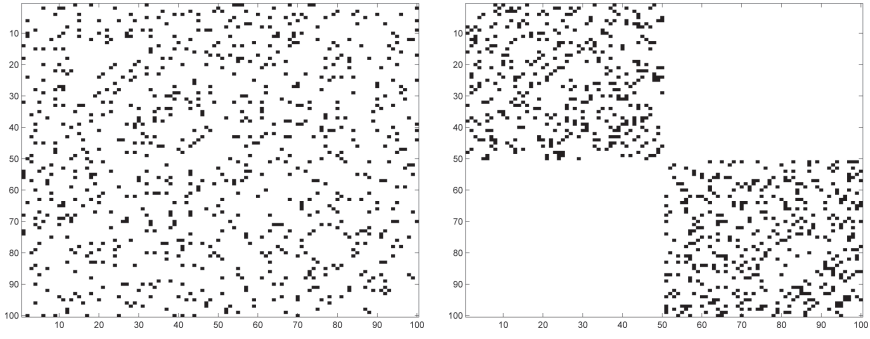
$$\widehat{p}_{00} = \frac{5}{10}; \quad \widehat{p}_{01} = \frac{7}{25}; \quad \widehat{p}_{11} = \frac{2}{10}.$$

Example 1.2 illustrates the ease of model fitting when binary-valued covariates are known; the notion of permutation similarity plays a similar role in the case of k-ary covariates.

1.3.2. *Approximate inference*

The careful reader will have noted that in the case of known categorical covariates, examples such as those above can be expressed as contingency tables – a notion we revisit in Section 1.4 – and hence may admit exact inference procedures. However, if covariates are latent, then an appeal to maximum-likelihood estimation induces a combinatorial optimization problem; in general, no fast algorithm is known for likelihood maximization over the set of covariates and Bernoulli parameters under the general k-group stochastic block model.

The principal difficulty arises in maximizing the n-dimensional k-ary covariate vector c over an exponentially large model space; estimating the



(a) Simulated data matrix $\boldsymbol{A}$ (b) Transformed data $\boldsymbol{\Pi A \Pi'}$

Fig. 1.3. Representations $\boldsymbol{A}$ and $\boldsymbol{\Pi A \Pi'}$ of data drawn from the stochastic block model of Example 1.3, corresponding to isomorphic graphs (black boxes denote links). Though $p_{01} = 0$, only a small subset of permutation similarity transformations $\boldsymbol{\Pi}(\cdot)\boldsymbol{\Pi'}$ will reveal the disconnected nature of this network.

$\binom{k+1}{2}$ associated Bernoulli parameters then proceeds in exact analogy to Example 1.2 above. The following example illustrates the complexity of this inference task.

Example 1.3 (Permutation and Maximization). *Consider a 100-node network generated according to the stochastic block model of Definition 1.2, with each group of size 50 and* $p_{00} = p_{11} = 1/2$, $p_{01} = 0$. *Figure 1.3 shows two permutation-similar adjacency matrices,* $\boldsymbol{A}$ *and* $\boldsymbol{\Pi A \Pi'}$, *that correspond to isomorphic graphs representing this network; inferring the vector* c *of binary categorical covariates from data* $\boldsymbol{A}$ *in Figure 1.3a is equivalent to finding a permutation similarity transformation* $\boldsymbol{\Pi A \Pi'}$ *that reveals the distinct division apparent in Figure 1.3b.*

Given the combinatorial nature of this problem in general, it is clear that fitting models to real-world network data can quickly necessitate approximate inference. To this end, Example 1.3 motivates an important means of exploiting algebraic properties of network adjacency structure: the notion of a *graph spectrum*. Eigenvalues associated with graphs reveal several of their key properties (Chung, 1997) at a computational cost that scales as the cube of the number of nodes, offering an appealing alternative in cases where exact solutions are of exponential complexity.

As the adjacency matrix $\boldsymbol{A}$ itself fails to be positive semidefinite, the spectrum of a labeled graph is typically defined via a Laplacian matrix $\boldsymbol{L}$ as follows.

Definition 1.3 (Graph Laplacian). *Let $i \sim j$ denote a symmetric adjacency relation defined on an n-node network. An associated $n \times n$ symmetric, positive-semidefinite matrix $\boldsymbol{L}$ is called a graph Laplacian if, for all $i, j \in \{1, 2 \ldots, n\} : i \neq j$, we have*

$$\boldsymbol{L}: \begin{cases} L_{ij} < 0 & \text{if } i \sim j, \\ L_{ij} = 0 & \text{if } i \nsim j. \end{cases}; \quad L_{ji} = L_{ij}.$$

Note that the diagonal of $\boldsymbol{L}$ is defined only implicitly, via the requirement of positive-semidefiniteness; a typical diagonally dominant completion termed the *combinatorial Laplacian* takes $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, where $\boldsymbol{D}$ is a diagonal matrix of node degrees such that $D_{ii} = \sum_{j=1}^{n} A_{ij}$. An important result is that the dimension of the kernel of $\boldsymbol{L}$ is equal to the number of connected components of the corresponding graph; hence $p_{01} = 0$ implies in turn that at least two eigenvalues of $\boldsymbol{L}$ will be zero in Example 1.3 above.

Correspondingly, Fiedler (1973) termed the second-smallest eigenvalue of the combinatorial Laplacian the *algebraic connectivity* of a graph, and recognized that positive and negative entries of the corresponding eigenvector (the "Fiedler vector") define a partition of nodes that nearly minimizes the number of edge removals needed to disconnect a network. In fact, in the extreme case of two equally sized, disconnected subgraphs – as given by Example 1.3 – this procedure exactly maximizes the likelihood of the data under a two-group stochastic block model; more generally, it provides a means of approximate inference that we shall return to in Section 1.4.

As reviewed by von Luxburg (2007), the observation of Fiedler was later formalized as an algorithm termed spectral bisection (Pothen *et al.*, 1990), and indeed leads to the more general notion of *spectral clustering* (von Luxburg *et al.*, 2008). This remains an active area of research in combinatorics and theoretical computer science, where a simple stochastic block model with $p_{00}, p_{11} > p_{01}$ is termed a "planted partition" model (Bollobás and Scott, 2004).

1.4. Testing for Network Structure

Identifying some degree of structure within a network data set is an important prerequisite to formal statistical analysis. Indeed, if all nodes of a network are truly unique and do not admit any notion of grouping, then the corresponding data set – no matter how large – is really only a single observation. On the other hand, if every node can be considered independent

under an assumed model, then depicting the data set as a network is unhelpful: the data are best summarized as n independent observations of nodes whose connectivity structure is uninformative.

In this section we invoke a formal hypothesis testing framework to explore the notion of detecting network structure in greater detail, and propose new approaches that are natural from a statistical point of view but have thus far failed to appear in the literature. To illustrate these ideas we apply three categories of tests to a single data set – that of Section 1.4.1 below – and in turn highlight a number of important topics for further development.

1.4.1. *The Zachary karate data*

Zachary (1977) recorded friendships between 34 members of a collegiate karate club that subsequently split into two groups of size 16 and 18. These data are shown in Figure 1.4, with inter- and intra-group links given in Table 1.1. The network consists of 78 links, with degree sequence (ordered in accordance with the node numbering of Figure 1.4) given by

$$(16, 9, 10, 6, 3, 4, 4, 4, 5, 2, 3, 1, 2, 5, 2, 2, 2,$$
$$2, 2, 3, 2, 2, 2, 5, 3, 3, 2, 4, 3, 4, 3, 6, 13, 17),$$

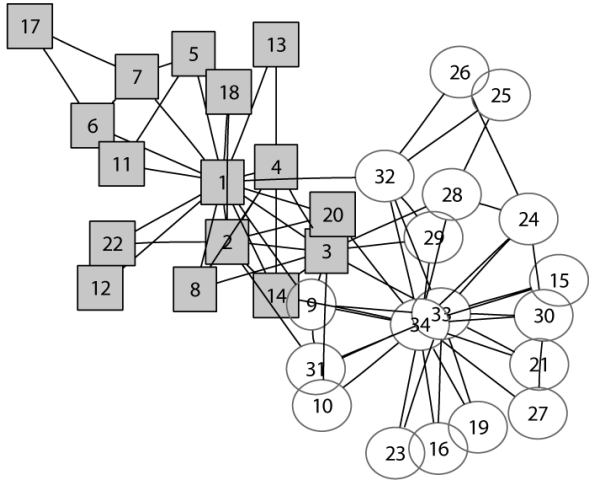


Fig. 1.4. Visualization of the Zachary karate data of Section 1.4.1. Nodes are numbered and binary categorical covariate values, reflecting the subsequent group split, are indicated by shape.

Table 1.1. Zachary (1977) karate data.

Counts	# Links	# No Links	Total
Intra-subgroup: 0–0	33	87	120
Inter-subgroup: 0–1	10	278	288
Intra-subgroup: 1–1	35	118	153
Total	78	483	561

and corresponding sample proportion of observed links given by $\hat{p} = 78/\binom{34}{2} = 78/561$.

Sociologists have interpreted the data of Zachary not only as evidence of network structure in this karate club, but also as providing binary categorical covariate values through an indication of the subsequent split into two groups, as per Figure 1.4. This in turn provides us with an opportunity to test various models of network structure – including those introduced in Section 1.3 – with respect to ground truth.

1.4.2. *Tests with known categorial covariates*

We begin by posing the question of whether or not the most basic Erdös–Rényi network model of Definition 1.1 – with each node being equally likely to connect to any other node – serves as a good description of the data, given the categorical variable of observed group membership. The classical evaluation of this hypothesis comes via a contingency table test.

Example 1.4 (Contingency Table Test). *Consider the data of Section 1.4.1. When categorical covariates are known, a contingency table test for independence between rows and columns may be performed according to the data shown in Table 1.1. The Pearson test statistic T_{χ^2} in this case evaluates to over* 47, *and with only* 2 *degrees of freedom, the corresponding p-value for these data is less than* 10^{-3}.

In this case, the null hypothesis – that the Erdös–Rényi model's sole Bernoulli parameter can be used to describe both inter- and intra-subgroup connection probabilities – can clearly be rejected.

As in the case of Zheng *et al.* (2006) and others, this χ^2 approach has been generally used to reject an Erdös–Rényi null when given network data include a categorical covariate for each node. (A cautionary reminder is in order: employing this method when covariates are inferred from data corresponds to a misuse of maximally selected statistics (Altman *et al.*, 1994).) Of course, in cases where it is computationally feasible, we may

instead use simulation to determine the exact distribution of any chosen test statistic T under whichever null model is assumed.

1.4.3. *The case of latent categorial covariates*

The Erdös–Rényi model of Definition 1.1 clearly implies a lack of network structure through its nodal properties, thus supporting its use as a null model in cases such as Example 1.4 and those described above. In contrast, the partial exchangeability exhibited by the stochastic block model of Definition 1.2 suggests its use as an alternate model that explicitly exhibits network structure. To this end, the usual Neyman–Pearson logic implies the adoption of a generalized likelihood ratio test statistic:

$$T_{LR} = \frac{\sup_{p} \prod_{i>j} \mathbb{P}(A_{ij}\,;p)}{\max_{c} \sup_{p_{00},p_{01},p_{11}} \prod_{i>j} \mathbb{P}(A_{ij}\,;p_{00},p_{01},p_{11},c(i),c(j))}$$

$$= \frac{\prod_{i>j} \widehat{p}^{\,A_{ij}} \,(1-\widehat{p})^{1-A_{ij}}}{\max_{c} \sup_{p_{00},p_{01},p_{11}} \prod_{i>j} (p_{c(i)c(j)})^{A_{ij}} (1-p_{c(i)c(j)})^{1-A_{ij}}}.$$

As we have seen in Section 1.3.2, however, maximizing the likelihood of the covariate vector $c \in \{0,1\}^n$ in general requires an exhaustive search. Faced with the necessity of approximate inference, we recall that the spectral partitioning algorithms outlined earlier in Section 1.3.2 provide an alternative to exact likelihood maximization in c. The resultant test statistic $T_{\widehat{LR}}$ is computationally feasible, though with reduced power, and to this end we may test the data of Section 1.4.1 as follows.

Example 1.5 (Generalized Likelihood Ratio Test). *Let T_{LR} be the test statistic associated with a generalized likelihood ratio test of Erdös–Rényi versus a two-group stochastic block model, and $T_{\widehat{LR}}$ correspond to an approximation obtained by spectral partitioning in place of the maximization over group membership. For the data of Section 1.4.1, simulation yields a corresponding p-value of less than 10^{-3} with respect to $T_{\widehat{LR}}$, with Figure 1.5 confirming the power of this test.*

Our case study has so far yielded reassuring results. However, a closer look reveals that selecting appropriate network models and test statistics may require more careful consideration.

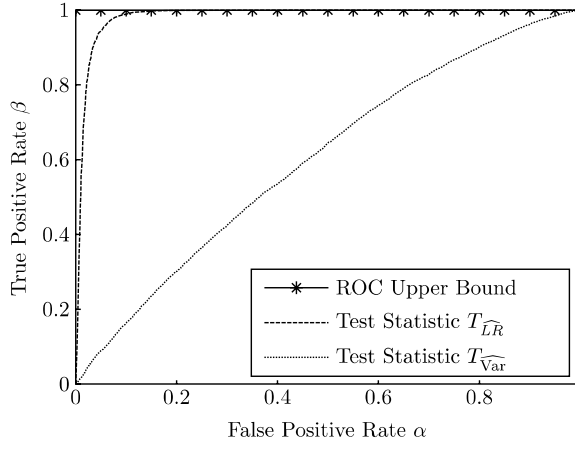


Fig. 1.5. Receiver operating characteristic (ROC) curves corresponding to tests of the data of Section 1.4.1, with Erdös–Rényi null and two-group stochastic block model alternate. Test statistics $T_{\widehat{LR}}$ and $T_{\widehat{\text{Var}}}$ were calculated via simulation, with the ROC upper bound obtained using knowledge of the true group membership for each node.

Example 1.6 (Degree Variance Test). *Suppose we adopt instead the test statistic of Snijders (1981)*:

$$T_{\widehat{\text{Var}}} = \frac{1}{n-1} \sum_{i=1}^{n} \left(\sum_{j=1}^{n} A_{ij} - \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{ij} \right)^2,$$

the sample variance of the observed degree sequence $\sum_{j=1}^{n} A_{ij}$. *A glance at the data of Section 1.4.1 indicates the poor fit of an Erdös–Rényi null, and indeed simulation yields a p-value of less than* 10^{-3}. *Figure 1.5, however, reveals that* $T_{\widehat{\text{Var}}}$ *possesses very little power.*

This dichotomy between a low p-value, and yet low test power, highlights a limitation of the models exhibited thus far: in each case, both the expected degree sequence and the corresponding node connectivity properties are determined by exactly the same set of model parameters. In this regard, test statistics depending on the data set only through its degree sequence can prove quite limiting, as the difference between the two models under consideration lies entirely in their node connectivity properties, rather than the heterogeneity of their degree sequences.

Indeed, significant degree variation is a hallmark of many observed network data sets, the data of Section 1.4.1 included; sometimes certain nodes are simply more connected than others. In order to conclude that

rejection of a null model necessarily implies the presence of network structure expressed through categorical covariates, a means of allowing for heterogenous degree sequences must be incorporated into the null as well as the alternate.

1.4.4. *Decoupling degree sequence and connectivity*

An obvious way to decouple properties of the degree sequence from those of connectivity is to restrict the model space to *only* those networks exhibiting the observed degree sequence. However, simulation of such graphs becomes nontrivial when they are restricted to be simple (i.e., without multiple edges or self-loops), thus rendering the test calculations of Section 1.4.2 more difficult to achieve in practice. Correspondingly, such fixed-degree models have remained largely absent from the literature to date.

Recent advances in graph simulation methods, however, help to overcome this barrier (Viger and Latapy, 2005; Blitzstein and Diaconis, 2011). The importance sampling approach of Blitzstein and Diaconis (2011) enables us here to test the data set of Section 1.4.1 using fixed-degree models that match its observed degree sequence. Although the corresponding normalizing constants cannot be computed in closed form, we may specify a proposal distribution, draw samples, and calculate unnormalized importance weights.

Example 1.7 (Fixed-Degree Test). *Consider the set of all simple graphs featuring an observed degree sequence, and define a null model under which each of these graphs is equally likely. As an alternate model, let each graph be weighted in proportion to its likelihood under the two-group stochastic block model of Definition 1.2; in this case the normalizing constant will depend on parameters* p_{00}, p_{01}, *and* p_{11}. *The corresponding fixed-degree generalized likelihood ratio test statistic* T_{LR-FD} *is given in analogy to Example 1.5 by*

$$\frac{1}{\max\limits_{c} \sup\limits_{p_{00},p_{01},p_{11}} \prod\limits_{i>j} \mathbb{P}(A_{ij}\,; p_{00}, p_{01}, p_{11}, c(i), c(j))}.$$

Just as before, calculation of T_{LR-FD} *requires a combinatorial search over group assignments* c; *moreover, the fixed-degree constraint precludes an analytical* sup *operation over parameters* p_{00}, p_{01}, *and* p_{11}. *We therefore define an approximation* $T_{\widehat{LR-FD}}$ *employing spectral partitioning in place of the maximization over group membership, and substituting the analytical*

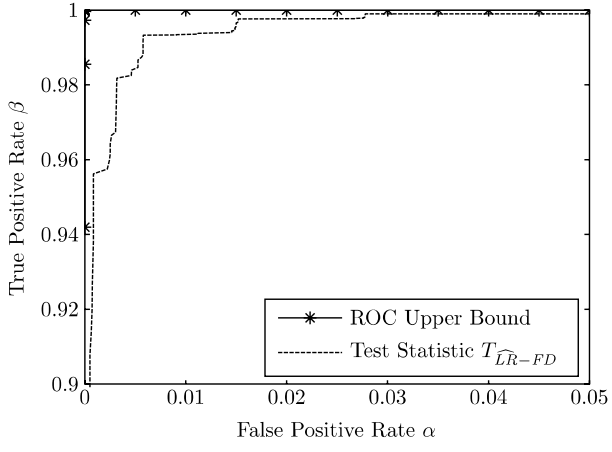


Fig. 1.6. ROC curve of $T_{\widehat{LR}-FD}$ using fixed-degree models for both the null and alternate hypotheses. (The stepped appearance of the curve is an artifact of the importance sampling weights.) Also shown is an ROC upper bound, obtained using knowledge of the true group membership for each node.

sup *under two-group stochastic block likelihood for the exact* sup *operation. The substantial power of this test for the data of Section 1.4.1 is visible in Figure 1.6*; *the estimated p-value of this data set remains below* 10^{-3}.

Note that specification of parameters p_{00}, p_{01}, and p_{11} was required to generate Figure 1.6 via simulation; here, we manually fit these three parameters to the data, starting with their estimates under the two-group stochastic block model, until the likelihood of the observed data approached the median likelihood under our parameterization. A more formal fitting procedure could, of course, be adopted in practice.

1.5. Open Problems in Network Inference

The examples of Sections 1.3 and 1.4 were designed to be illustrative, and yet they also serve to illuminate broader questions that arise as we seek to extend classical notions of statistics to network data. As we have seen in Section 1.3, for instance, the inclusion of latent k-ary categorical covariates immediately necessitates a variety of combinatorial calculations. The increasing prevalence of large, complex network data sets presents an even more significant computational challenge for statistical inference. Indeed, longstanding inferential frameworks – as exemplified by the hypothesis tests of Section 1.4, for instance – are crucial to the analysis of networks

and relational data, and yet their implementations can prove remarkably difficult even for small data sets.

To address these broader questions and impact the future of network inference, we believe that statisticians should focus on the following three main categories of open problems, whose descriptions comprise the remainder of this section:

(1) We must work to specify models that can more realistically describe observed network data. For instance, the fixed-degree models introduced earlier account explicitly for heterogeneous degree sequences; in the case of large-scale network data sets, even more flexible models are needed.
(2) We must build approximations to these models for which likelihood maximization can be readily achieved, along with tools to evaluate the quality of these approximations. The spectral partitioning approach featured in our examples of Section 1.4 serves as a prime example; however, validation of approximate inference procedures remains an important open question.
(3) We must seek to understand precisely how network sampling influences our statistical analyses. In addition to better accounting for data gathering mechanisms, sampling can serve as a method of data reduction. This in turn will enable the application of a variety of methods to data sets much larger than those exhibited here.

1.5.1. *Model elicitation and selection*

More realistic network models can only serve to benefit statistical inference, regardless of their computational or mathematical convenience (Banks and Constantine, 1998). Models tailored to different fields, and based on theory fundamental to specific application areas, are of course the long-term goal – with the exponential random graph models reviewed by Anderson *et al.* (1999) and Snijders *et al.* (2006) among the most successful and widely known to date. However, additional work to determine more general models for network structure will also serve to benefit researchers and practitioners alike. As detailed in the Appendix, there are presently several competing models of this type, each with its own merits: stochastic block models (Wang and Wong, 1987), block models with mixed membership (Airoldi *et al.*, 2007), and structural models that explicitly incorporate information regarding the degree sequence in addition to group membership (Chung *et al.*, 2003).

At present, researchers lack a clearly articulated strategy for selecting between these different modeling approaches – the goodness-of-fit procedures of Hunter *et al.* (2008), based on graphical comparisons of various network statistics, provide a starting point, but comparing the complexity of these different modeling strategies poses a challenge. Indeed, it is not even entirely clear how best to select the number of groups used in a single modeling strategy alone. For the data of Section 1.4.1, for example, we restricted our definition of network structure to be a binary division of the data into two groups, whereas many observed data sets may cluster into an *a priori* unknown number of groups.

It is also worth noting that many different fields of mathematics may provide a source for network data models. While graph theory forms a natural starting point, other approaches based on a combination of random matrices, algebra, and geometry may also prove useful. For example, the many graph partitioning algorithms based on spectral methods suggest the use of corresponding generative models based on the eigenvectors and eigenvalues of the graph Laplacian. The primary challenge in this case appears to be connecting such models to the observed data matrix $\boldsymbol{A}$, which typically consists of binary entries.

1.5.2. *Approximate inference and validation*

Computationally or mathematically convenient models will also continue to play a key role in network analysis. Even simple generic models of structure are very high-dimensional, and with network data sets commonly consisting of thousands to millions of nodes, model dimensionality spirals out of control at an impossible rate. Somehow this fundamental challenge of network data – how to grapple with the sheer number of relational observations – must be turned into a strength so that further analysis may proceed. Reducing the dimensionality through an approximate clustering is an excellent first step to build upon, but computationally realizable inference schemes must also follow in turn.

The usefulness of such approximations will ultimately be determined by the extent to which evaluation tools can be developed for massive data sets. Whenever models are sufficiently complex to necessitate approximate inference procedures, such models must be paired with mechanisms to relate the quality of the resulting analysis back to the original problem and model specification. Indeed, assurances are needed to convince thoughtful practitioners that analyzing a different model, or maximizing a quantity other than desired likelihood, is a useful exercise.

Other approaches to validation may focus on the outcome of the analysis in some way, rather than its theoretical underpinnings. With ground truth by its very definition available only for small-scale illustrative problems, or for those which are generally felt to have already been solved, prediction may provide a valuable substitute. By monitoring the results of approximation over time relative to revealed truth, confidence in the adopted inference procedure may be grown.

1.5.3. *Sampling, missingness, and data reduction*

A final concern is to better understand how sampling mechanisms influence network inference. Consider that two critical assumptions almost always underpin the vast majority of contemporary network analyses: First, that all links within the collection of observed nodes have been accounted for; and second, that observed nodes within the network comprise the only nodes of interest. In general, neither of these assumptions may hold in practice.

To better understand the pitfalls of the first assumption, consider that while observing the presence of a link between nodes is typically a feasible and well-defined task, observing the *absence* of a link can in many cases pose a substantial challenge. Indeed, careful reflection often reveals that zero entries in relational data sets are often better thought of as unobserved (Clauset *et al.*, 2008; Marchette and Priebe, 2008). The implications of this fact for subsequent analysis procedures – as well as on approximate likelihood maximization procedures and spectral methods in particular – remain unclear.

The second assumption, that all nodes of interest have in fact been recorded, also appears rarely justified in practice. Indeed, it seems an artifact of this assumption that most commonly studied data sets consist of nodes which form a connected network. While in some cases the actual network may in fact consist of a single connected component, researchers may have unwittingly selected their data conditioned upon its appearance in the largest connected component of a much larger network. How this selection in turn may bias the subsequent fitting of models has only recently begun to be investigated (Handcock and Gile, 2010).

A better understanding of missingness may also lend insight into optimal sampling procedures. Although researchers themselves may lack influence over data gathering mechanisms, the potential of such methods for data reduction is clear. One particularly appealing approach is to first

sample very large network data sets in a controlled manner, and then apply exact analysis techniques. In some cases the resultant approximation error can be bounded (Belabbas and Wolfe, 2009), implying that the effects on inferential procedures in question can be quantified.

Other data reduction techniques may also help to meet the computational challenges of network analysis; for example, Krishnamurthy *et al.* (2007) examined contractions of nodes into groups as a means of lessening data volume. Such strategies of reducing network size while preserving relevant information provide an alternative to approximate likelihood maximization that is deserving of further study.

1.6. Conclusion

In many respects, the questions being asked of network data sets are not at all new to statisticians. However, the increasing prevalence of large networks in contemporary application areas gives rise to both challenges and opportunities for statistical science. Tests for detecting network structure in turn form a key first step toward more sophisticated inferential procedures, and moreover provide practitioners with much-needed means of formal data analysis.

Classical inferential frameworks are precisely what is most needed in practice, and yet as we have seen, their exact implementation can prove remarkably difficult in the setting of modern high-dimensional, non-Euclidean network data. To this end, we hope that this chapter has succeeded in helping to chart a path toward the ultimate goal of a unified and coherent framework for the statistical analysis of large-scale network data sets.

Acknowledgments

Research supported in part by the National Science Foundation under Grants DMS-0631636 and CBET-0730389; by the Defense Advanced Research Projects Agency under Grant No. HR0011-07-1-0007; by the US Army Research Office under PECASE Award W911NF-09-1-0555 and MURI Award 58153-MA-MUR; by the UK EPSRC under Mathematical Sciences Established Career Fellowship EP/K005413/1 and Institutional Sponsorship Award EP/K503459/1; by the UK Royal Society under a Wolfson Research Merit Award; and by Marie Curie FP7 Integration Grant PCIG12-GA-2012-334622 within the 7th European Union Framework Program.

Appendix: A Review of Approaches to Network Analysis

Three canonical problems in network data analysis have consistently drawn attention across different contexts: network model elicitation, network model inference, and methods of approximate inference.

A.1. Model Elicitation

With new network data sets being generated or discovered at rapid rates in a wide variety of fields, model elicitation – independent even of model selection – remains an important topic of investigation. Although graph theory provides a natural starting point for identifying possible models for graph-valued data, practitioners have consistently found that models such as Erdös–Rényi lack sufficient explanatory power for complex data sets. Its inability to model all but the simplest of degree distributions has forced researchers to seek out more complicated models.

Barabási (2002) and Palla *et al.* (2005) survey a wide variety of network data sets and conclude that commonly encountered degree sequences follow a power law or similarly heavy-tailed distribution; the Erdös–Rényi model, with its marginally binomial degree distribution, is obviously insufficient to describe such data sets. Barabási and Albert (1999) introduced an alternative by way of a generative network modeling scheme termed "preferential attachment" to explicitly describe power-law degree sequences. Under this scheme, nodes are added sequentially to the graph, being preferentially linked to existing nodes based on the current degree sequence. A moment's reflection will convince the reader that this model is in fact an example of a Dirichlet process (Pemantle, 2007).

Though the preferential attachment approach serves to describe the observed degree sequences of many networks, it can fall short of correctly modeling their patterns of connectivity (Li *et al.*, 2005); moreover, heterogenous degree sequences may not necessarily follow power laws. A natural solution to both problems is to condition on the observed degree sequence as in Section 1.4.4 and consider the connections between nodes to be random. As described earlier, the difficulties associated with simulating fixed-degree simple graphs have historically dissuaded researchers from this direction, and hence fixed-degree models have not yet seen wide use in practice.

As an alternative to fixed-degree models, researchers have instead focused on the so-called configuration model (Newman *et al.*, 2001) as well as models which yield graphs of given expected degree (Chung *et al.*, 2003).

The configuration model specifies the degree sequence exactly, as with the case of fixed-degree models, but allows both multiple links between nodes and "self-loops" in order to gain a simplified asymptotic analysis. Models featuring given expected degrees specify only the expected degree of each node – typically set equal to the observed degree – and allow the degree sequence to be random. Direct simulation becomes possible if self-loops and multiple links are allowed, thus enabling approximate inference methods of the type described in Section 1.3.2. However, observed network data sets do not always exhibit either of these phenomena, thus rendering the inferential utility of these models highly dependent on context. In the case of very large data sets, for example, the possible presence or absence of multiple connections or self-loops in the model may be irrelevant to describing the data on a coarse scale. When it becomes necessary to model network data at a fine scale, however, a model which allows for these may be insufficiently realistic.

Graph models may equally well be tailored to specific fields. For example, sociologists and statisticians working in concert have developed a class of well-known approaches collectively known as exponential random graph models (ERGMs) or alternatively as p^* models. Within this class of models, the probability of nodes being linked to each other depends explicitly on parameters that control well-defined sufficient statistics; practitioners draw on sociological theory to determine which connectivity statistics are critical to include within the model. A key advantage of these models is that they can readily incorporate covariates into their treatment of connectivity properties. For a detailed review, along with a discussion of some of the latest developments in the field of social networks, the reader is referred to Anderson *et al.* (1999) and Snijders *et al.* (2006).

Since their original introduction, ERGMs have been widely adopted as models for social networks. They have not yet, however, been embraced to the same extent by researchers outside of social network analysis. Sociologists can rely on existing theory to select models for how humans form relationships with each other; researchers in other fields, though, often cannot appeal to equivalent theories. For exploratory analysis, they may require more generic models to describe their data, appearing to prefer models with a latent vector of covariates to capture probabilistically exchangeable blocks. Indeed, as noted in Section 1.3.1, this approach falls under the general category of stochastic block modeling. Wang and Wong (1987) detail similarities and differences between this approach and the original specification of ERGMs.

Stochastic block modeling, though relatively generic, may still fail to adequately describe networks in which nodes roughly group together, yet in large part fail to separate into distinct clusters. In cases such as this, where stochastic exchangeability is too strong an assumption, standard block modeling breaks down. To this end, two possible modeling solutions have been explored to date in the literature. Hoff *et al.* (2002) introduced a latent space approach, describing the probability of connection as a function of distances between nodes in an unobserved space of known dimensionality. In this model, the observed grouping of nodes is a result of their proximity in this latent space. In contrast, Airoldi *et al.* (2007) retained the explicit grouping structure that stochastic block modeling provides, but introduced the idea of mixed group membership to describe nodes that fall between groups. Node membership here is a vector describing partial membership in all groups, rather than an indicator variable specifying a single group membership.

A.2. Model Fitting and Inference

Even when a model or class of models for network data can be specified, realizing inference can be challenging. One of the oldest uses of random graph models is as a null; predating the computer, Moreno and Jennings (1938) simulated a random graph model quite literally by hand in order to tabulate null model statistics. These authors drew cards out of a ballot shuffling apparatus to generate graphs of the same size as a social network of schoolgirls they had observed. Comparing the observed statistics to the distribution of tabulated statistics, they rejected the hypothesis that the friendships they were observing were formed strictly by chance.

Asymptotic tests may alleviate the need for simulation in cases of large network data sets, and are available for certain models and test statistics – the χ^2-test of Section 1.4.2 being one such example. As another example, Holland and Leinhardt (1981) developed asymptotic tests based on likelihood ratio statistics to select between different ERGMs. Sociologists and statisticians together have developed results for other test statistics as well, many of which are reviewed by Wasserman and Faust (1994).

A desire upon the rejection of a null model, of course, is the fitting of an alternate. However, as demonstrated in Section 1.3.2, direct fitting by maximum likelihood can prove computationally costly, even for basic network models. A common solution to maximizing the likelihood under an ERGM, for example, is to employ a Markov chain Monte Carlo strategy (Snijders

et al., 2006). Handcock *et al.* (2007) also used such methods to maximize the likelihood of a latent space network model; additionally, these authors suggested a faster, though approximate, two-stage maximization routine.

Other researchers have employed greedy algorithms to maximize the model likelihood. Newman and Leicht (2007) used expectation maximization (EM) to fit a network model related to stochastic block modeling. Relaxing the precise requirements of the EM algorithm, both Hofman and Wiggins (2008) and Airoldi *et al.* (2008) have applied a variational Bayes approach (see, e.g., Jordan *et al.* (1999)) to find maximum likelihood estimates of parameters under a stochastic block model. Reichardt and Bornholdt (2004) applied simulated annealing to maximize the likelihood of network data under a Potts model, a generalization of the Ising model. Rosvall and Bergstrom (2007, 2008) have also employed simulated annealing in network inference in order to maximize information-theoretic functionals of the data.

Following any kind of model fitting procedure, a goodness-of-fit test of some kind is clearly desirable. Yet, researchers have thus far struggled to find a clear solution to this problem. Hunter *et al.* (2008) have proposed a general method of accumulating a wide set of network statistics, and comparing them graphically to the distribution of these same statistics under a fitted model. Networks which fit well should in turn exhibit few statistics that deviate far from those simulated from the corresponding model.

A.3. Approximate Inference Procedures

In most cases of practical interest, and in particular for large network data sets, model likelihoods cannot be maximized in a computationally feasible manner, and researchers must appeal to a heuristic that yields some approximately maximized quantity. With this goal in mind, the idea of likelihood maximization has been subsumed by the idea of fast graph partitioning described in Section 1.3.2, as it is the process of determining group membership which typically poses the most computational challenges. The invention of new algorithms that can quickly partition large graphs is clearly of great utility here.

A.3.1. *Algorithmic approaches*

Computer scientists and physicists have long been active in the creation of new graph partitioning algorithms. In addition to techniques such as spectral bisection, many researchers have also noted that the inherently

sparse nature of most real-world adjacency structures enables faster implementations of spectral methods (see, e.g., White and Smyth (2005)).

Researchers have sought to also incorporate graph partitioning concepts that allow for multiple partitions of varying sizes. Some researchers, such as Eckmann and Moses (2002) and Radicchi *et al.* (2004), have attempted to use strictly local statistics to aid in the clustering of nodes into multiple partitions. Girvan and Newman (2002) focused in contrast on global statistics, by way of measures of the centrality of a node relative to the rest of the graph. This line of reasoning eventually resulted in the introduction of modularity (Newman, 2006) as a global statistic to relate the observed number of edges between groups to their expected number under the configuration model outlined in Section A.1. Spectral clustering methods can also be applied to the task of approximately maximizing modularity, in a manner that enables both group size and number to vary. A wide variety of alternative maximization approaches have been applied as well: Both Wang *et al.* (2007) and Brandes *et al.* (2008) review the computational difficulties associated with maximization of the modularity statistic, and relate this to known combinatorial optimization problems. Fortunato and Castellano (2007) review many recently proposed maximization routines and contrast them with traditional methods.

A.3.2. *Evaluation of efficacy*

Approximate procedures in turn require some way to evaluate the departure from exact likelihood maximization. Thus far, a clear way to evaluate partitions found through the various heuristics cited above has not yet emerged, though many different approaches have been proposed. Both Massen and Doye (2006) and Karrer *et al.* (2008) have explored ways to test the statistical significance of the output of graph partitioning algorithms. Their methods attempt to determine whether a model which lacks structure could equally well explain the group structure inferred from the data. These approaches, though distinct from one another, are both akin to performing a permutation test – a method known to be effective when applied to more general cases of clustering. Carley and Banks (1993) apply this exact idea to test for structure when group memberships are given.

Other researchers have attempted a more empirical approach to the problem of partition evaluation by adopting a metric to measure the distance between found and "true" partitions. Such distances are then examined for a variety of data sets and simulated cases for which the true

partition is assumed known. In this vein Danon *et al.* (2005) specified an explicit probability model for structure and compared how well different graph partitioning schemes recovered the true subgroups of data, ranking them by both execution time as well as average distance between true and found partitions. Gustafsson *et al.* (2006) performed a similar comparison, along with a study of differences in "found" partitions between algorithms for several well-known data sets, including the karate club data of Section 1.4.1. They found that standard clustering algorithms (e.g., k-means) sometimes outperform more specialized network partition algorithms. Finally, Fortunato and Barthélemy (2007) have undertaken theoretical investigations of the sensitivity and power of a particular partitioning algorithm to detect subgroups below a certain size.

References

Adamic, L. A. and Huberman, B. A. (2000). Power-law distribution of the World Wide Web, *Science* **287**, p. 2115.

Airoldi, E. M., Blei, D. M., Fienberg, S. E. and Xing, E. P. (2007). Combining stochastic block models and mixed membership for statistical network analysis, in E. M. Airoldi, D. M. Blei, S. E. Fienberg, A. Goldenberg, E. P. Xing and A. X. Zheng (eds), *Papers from the ICML 2006 Workshop on Statistical Network Analysis* (Springer, Berlin), pp. 57–74.

Airoldi, E. M., Blei, D. M., Fienberg, S. E. and Xing, E. P. (2008). Mixed membership stochastic block models, *J. Machine Learn. Res.* **9**, pp. 1981–2014.

Altman, D. G., Lausen, B., Sauerbrei, W. and Schumacher, M. (1994). Dangers of using "optimal" cutpoints in the evaluation of prognostic factors, *J. Natl. Cancer Inst.* **86**, pp. 829–835.

Anderson, C. J., Wasserman, S. and Crouch, B. (1999). A p^* primer: Logit models for social networks, *Social Networks* **21**, pp. 37–66.

Banks, D. and Constantine, G. M. (1998). Metric models for random graphs, *J. Classificat.* **15**, pp. 199–223.

Barabási, A. L. (2002). *Linked: The New Science of Networks* (Perseus Publishing, Cambridge, MA).

Barabási, A. L. and Albert, R. (1999). Emergence of scaling in random networks, *Science* **286**, pp. 509–512.

Batada, N. N., Reguly, T., Breitkreutz, A., Boucher, L., Breitkreutz, B.-J., Hurst, L. D. and Tyers, M. (2006). Stratus not altocumulus: A new view of the yeast protein interaction network, *PLoS Biology* **4**, pp. 1720–1731.

Belabbas, M.-A. and Wolfe, P. J. (2009). Spectral methods in machine learning and new strategies for very large data sets, *Proc. Natl. Acad. Sci. USA* **106**, pp. 369–374.

Blitzstein, J. and Diaconis, P. (2011). A sequential importance sampling algorithm for generating random graphs with prescribed degrees, *Internet Math.* **6**, pp. 489–522.

Bollobás, B. and Scott, A. D. (2004). Max Cut for random graphs with a planted partition, *Combinat. Probab. Comput.* **13**, pp. 451–474.

Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z. and Wagner, D. (2008). On modularity clustering, *IEEE Trans. Knowl. Data Eng.* **20**, pp. 172–188.

Carley, K. and Banks, D. (1993). Nonparametric inference for network data, *J. Math. Sociol.* **18**, pp. 1–26.

Ceyhan, E., Priebe, C. E. and Marchette, D. J. (2007). A new family of random graphs for testing spatial segregation, *Canad. J. Statist.* **35**, pp. 27–50.

Chung, F., Lu, L. and Vu, V. (2003). Spectra of random graphs with given expected degrees, *Proc. Natl. Acad. Sci. USA* **100**, pp. 6313–6318.

Chung, F. R. K. (1997). *Spectral Graph Theory* (American Mathematical Society, Providence, RI).

Clauset, A., Moore, C. and Newman, M. E. J. (2008). Hierarchical structure and the prediction of missing links in networks, *Nature* **453**, pp. 98–101.

Danon, L., Diaz-Guilera, A., Duch, J. and Arenas, A. (2005). Comparing community structure identification, *J Statist. Mech.* **9**, p. P09008.

de Solla Price, D. J. (1965). Networks of scientific papers, *Science* **149**, pp. 510–515.

Dunne, J. A., Williams, R. J. and Martinez, N. D. (2002). Food-web structure and network theory: The role of connectance and size, *Proc. Natl. Acad. Sci. USA* **99**, pp. 12917–12922.

Eagle, N., Pentland, A. and Lazer, D. (2008). Mobile phone data for inferring social network structure, in H. Liu, J. J. Salerno and M. J. Young (eds), *Social Computing, Behavioral Modeling, and Prediction* (Springer, New York, NY), pp. 79–88.

Eckmann, J.-P. and Moses, E. (2002). Curvature of co-links uncovers hidden thematic layers in the World Wide Web, *Proc. Natl. Acad. Sci. USA* **99**, pp. 5825–5829.

Erdös, P. and Rényi, A. (1959). On random graphs, *Publicat. Mathemat.* **6**, pp. 290–297.

Fiedler, M. (1973). Algebraic connectivity of graphs, *Czech. Math. J.* **23**, pp. 298–305.

Fortunato, S. and Barthélemy, M. (2007). Resolution limit in community detection, *Proc. Natl. Acad. Sci. USA* **104**, pp. 36–41.

Fortunato, S. and Castellano, C. (2007). Community structure in graphs, Unpublished manuscript. Available at: http://arxiv.org/abs/0712.2716. Accessed 09.09.13.

Gilbert, E. N. (1959). Random graphs, *Ann. Math. Stat.* **30**, pp. 1141–1144.

Girvan, M. and Newman, M. E. J. (2002). Community structure in social and biological networks, *Proc. Natl. Acad. Sci. USA* **99**, pp. 7821–7826.

Gustafsson, M., Hörnquist, M. and Lombardi, A. (2006). Comparison and validation of community structures in complex networks, *Physica A: Statist. Mech. Appl.* **367**, pp. 559–576.

Handcock, M. S. and Gile, K. J. (2010). Modeling social networks from sampled data, *Ann. Appl. Statist.* **4**, pp. 5–25.

Handcock, M. S., Raftery, A. E. and Tantrum, J. M. (2007). Model-based clustering for social networks, *J. Roy. Statist. Soc. A* **170**, pp. 301–354.

Hoff, P. D., Raftery, A. E. and Handcock, M. S. (2002). Latent space approaches to social network analysis, *J. Amer. Statist. Assoc.* **97**, pp. 1090–1098.

Hofman, J. M. and Wiggins, C. H. (2008). Bayesian approach to network modularity, *Phys. Rev. Lett.* **100**, pp. 258701(1–4).

Holland, P. W. and Leinhardt, S. (1981). An exponential family of probability distributions for directed graphs, *J. Amer. Statist. Assoc.* **76**, pp. 33–50.

Hunter, D. R., Goodreau, S. M. and Handcock, M. S. (2008). Goodness of fit of social network models, *J. Amer. Statist. Assoc.* **103**, pp. 248–258.

Jin, R. K.-X., Parkes, D. C. and Wolfe, P. J. (2007). Analysis of bidding networks in eBay: Aggregate preference identification through community detection, in C. Geib and D. Pynadath (eds), *Plan, Activity, and Intent Recognition (PAIR): Papers from the 2007 AAAI Workshop* (AAAI Press, Menlo Park, CA), pp. 66–73.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S. and Saul, L. K. (1999). An introduction to variational methods for graphical models, *Machine Learn.* **37**, pp. 183–233.

Karrer, B., Levina, E. and Newman, M. E. J. (2008). Robustness of community structure in networks, *Phys. Rev. E* **77**, pp. 46119(1–9).

Krishnamurthy, V., Faloutsos, M., Chrobak, M., Cui, J. H., Lao, L. and Percus, A. G. (2007). Sampling large Internet topologies for simulation purposes, *Comput. Networks* **51**, pp. 4284–4302.

Li, L., Alderson, D., Doyle, J. C. and Willinger, W. (2005). Towards a theory of scale-free graphs: Definition, properties, and implications, *Internet Math.* **2**, pp. 431–523.

Marchette, D. J. and Priebe, C. E. (2008). Predicting unobserved links in incompletely observed networks, *Computat. Statist. Data Anal.* **52**, pp. 1373–1386.

Massen, C. P. and Doye, J. P. K. (2006). Thermodynamics of community structure, Unpublished manuscript. Available at: http://arxiv.org/abs/cond-mat/0610077. Accessed 09.09.13.

Moreno, J. L. and Jennings, H. H. (1938). Statistics of social configurations, *Sociometry* **1**, pp. 342–374.

Newman, M. E. J. (2003). The structure and function of complex networks, *SIAM Rev.* **45**, pp. 167–256.

Newman, M. E. J. (2006). Modularity and community structure in networks, *Proc. Natl. Acad. Sci. USA* **103**, pp. 8577–8582.

Newman, M. E. J. and Leicht, E. A. (2007). Mixture models and exploratory analysis in networks, *Proc. Natl. Acad. Sci. USA* **104**, pp. 9564–9569.

Newman, M. E. J., Strogatz, S. H. and Watts, D. J. (2001). Random graphs with arbitrary degree distributions and their applications, *Phys. Rev. E* **64**, pp. 26118(1–17).

Palla, G., Derényi, I., Farkas, I. and Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society, *Nature* **435**, pp. 814–818.

Pemantle, R. (2007). A survey of random processes with reinforcement, *Probab. Surv.* **4**, pp. 1–79.

Pothen, A., Simon, H. D. and Liou, K.-P. (1990). Partitioning sparse matrices with eigenvectors of graphs, *SIAM J. Matrix Anal. Appl.* **11**, pp. 430–452.

Radicchi, F., Castellano, C., Cecconi, F., Loreto, V. and Parisi, D. (2004). Defining and identifying communities in networks, *Proc. Natl. Acad. Sci. USA* **101**, pp. 2658–2663.

Reichardt, J. and Bornholdt, S. (2004). Detecting fuzzy community structures in complex networks with a Potts model, *Phys. Rev. Lett.* **93**, pp. 218701(1–4).

Rosvall, M. and Bergstrom, C. T. (2007). An information-theoretic framework for resolving community structure in complex networks, *Proc. Natl. Acad. Sci. USA* **104**, pp. 7327–7331.

Rosvall, M. and Bergstrom, C. T. (2008). Maps of random walks on complex networks reveal community structure, *Proc. Natl. Acad. Sci. USA* **105**, pp. 1118–1123.

Snijders, T. A. B. (1981). The degree variance: An index of graph heterogeneity, *Social Networks* **3**, pp. 163–223.

Snijders, T. A. B., Pattison, P. E., Robins, G. L. and Handcock, M. S. (2006). New specifications for exponential random graph models, *Sociolog. Methodol.* **36**, pp. 99–153.

Stutzbach, D., Rejaie, R., Duffield, N., Sen, S. and Willinger, W. (2006). On unbiased sampling for unstructured peer-to-peer networks, in *Proc. 6th ACM SIGCOMM Conference on Internet Measurement*, pp. 27–40.

Thompson, S. K. (2006). Adaptive web sampling, *Biometrics* **62**, pp. 1224–1234.

Viger, F. and Latapy, M. (2005). Efficient and simple generation of random simple connected graphs with prescribed degree sequence, in L. Wang (ed.), *Proc. 11th Annual International Computing and Combinatorics Conference* (Springer, Berlin), pp. 440–449.

von Luxburg, U. (2007). A tutorial on spectral clustering, *Statist. Comput.* **17**, pp. 395–416.

von Luxburg, U., Belkin, M. and Bousquet, O. (2008). Consistency of spectral clustering, *Ann. Statist.* **36**, pp. 555–586.

Wang, R., Wang, Y., Zhang, X. and Chen, L. (2007). Detecting community structure in complex networks by optimal rearrangement clustering, in J. G. Carbonell and J. Siekmann (eds), *Papers from the PAKDD 2007 International Workshops* (Springer, Berlin), pp. 119–130.

Wang, Y. J. and Wong, G. Y. (1987). Stochastic block models for directed graphs, *J. Amer. Statist. Assoc.* **82**, pp. 8–19.

Wasserman, S. and Faust, K. (1994). *Social Network Analysis: Methods and Applications* (Cambridge University Press, Cambridge).

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of "small-world" networks, *Nature* **393**, pp. 440–442.

White, S. and Smyth, P. (2005). A spectral clustering approach to finding communities in graphs, in *Proc. SIAM International Conference on Data Mining*, pp. 274–285. Available at: http://www.cs.ucr.edu/%7Eeamon/HOT%20SAX%20%zolong-ver.pdf. Accessed 04.09.13.

Zachary, W. W. (1977). An information flow model for conflict and fission in small groups, *J. Anthropolog. Res.* **33**, pp. 452–473.

Zheng, T., Salganik, M. J. and Gelman, A. (2006). How many people do you know in prison?: Using overdispersion in count data to estimate social structure in networks, *J. Amer. Statist. Assoc.* **101**, pp. 409–423.

This page intentionally left blank

Chapter 2

Rapid Detection of Attacks in Computer Networks by Quickest Changepoint Detection Methods

Alexander G. Tartakovsky

University of Southern California, Department of Mathematics
3620 S. Vermont Avenue, KAP-108, Los Angeles, CA 90089-2532, USA
tartakov@usc.edu

University of Connecticut, Department of Statistics
215 Glenbrook Road, U-4120, Sorrs, CT 06269, USA
a.tartakov@uconn.edu

Changepoint problems deal with detecting changes in a process that occur at unknown points in time. The gist of the quickest changepoint problem is to design a detection procedure that minimizes the expected detection delay of a real change subject to a bound on the false alarm rate. In this chapter, we argue that network anomaly detection can be efficiently performed using changepoint detection methods. More specifically, we propose an anomaly intrusion detection system that exploits score-based versions of cumulative sum (CUSUM) and Shiryaev–Roberts detection algorithms. These algorithms are robust, computationally simple, and efficient for the detection of a wide variety of network intrusions that lead to relatively abrupt changes in network traffic. We also devise a novel hybrid anomaly-spectral-signature intrusion detection system that integrates change detection-based anomaly and spectral-based signature detection systems. This hybrid system allows for efficient filtering of false detections and confirmation of true attacks, ensuring very high speeds of detection with extremely low false alarm rates. The results are illustrated for several real data sets with denial-of-service flooding attacks on backbone links as well as for detecting spam campaigns.

2.1. Introduction

Cyber-security has evolved into a critical 21st-century problem that affects governments, businesses and individuals. Recently, cyber-threats have become more diffuse, more complex, and harder to detect.

Malicious activities and intrusion attempts such as spam campaigns, phishing, personal data theft, worms, distributed denial-of-service (DDoS)

attacks, address resolution protocol man-in-the-middle attacks, fast flux, etc. occur every day, have become commonplace in contemporary computer networks, and pose enormous risks to users for a multitude of reasons. These threats can incur significant financial damage and severely compromise the integrity of personal information. It is therefore essential to devise automated techniques to detect such events as quickly as possible so as to respond appropriately and eliminate the negative consequences for the users.

Current ultra-high-speed networks carry massive aggregate data flows. Malicious events usually produce (relatively) abrupt changes in network traffic profiles, which must be detected and isolated rapidly while keeping a low false alarm rate (FAR). Both requirements are important. However, rapid intrusion detection with minimal FAR and the capability to detect a wide spectrum of attacks is a challenge for modern ultra-high-speed networks. This problem is compounded by the increasing dimensionality in terms of the sheer number and complexity of attacks and the myriad of available detection systems, which appear non-commensurate due to the lack of unified performance metrics for unbiased performance evaluation. Moreover, even routine behavior of users could generate anomalous events requiring the attention of network operators and managers. A good example would be flash-crowds. Efficient operation and management of computer networks depend heavily on a relatively precise analysis of anomalies produced by both malicious and legitimate behavior. As a result, the challenges of intrusion detection in high-speed networks constantly outstrip our ability to detect, track, and interpret anomalies. This combination of massive complex data and the difficulty of extracting relevant information overwhelms human operators.

Detection of traffic anomalies in computer networks is performed by employing *Intrusion Detection Systems* (IDS). Such systems in one way or another capitalize on the fact that *maltraffic* is noticeably different from *legitimate* traffic. Depending on the principle of operation there are two categories of IDSs: signature-based or anomaly-based. For an overview see: Debar *et al.* (1999); Ellis and Speed (2001); Kent (2000). A signature-based IDS (SbIDS) inspects passing traffic to find matches against already known malicious patterns. Examples of SbIDSs are Snort (Roesch, 1999) and Bro (Paxson, 1999). An anomaly-based IDS (AbIDS) is first trained to recognize normal network behavior and then watches for any deviation from the normal profile, classifying deviations as potential attacks (Kent, 2000; Tartakovsky *et al.*, 2006a,b, 2013).

As an example, consider DDoS attacks, which lead to abrupt changes in network traffic (Loukas and Öke, 2010; Mirkovic *et al.*, 2004). Such attacks typically involve many traffic streams resulting in a large number of packets aimed at congesting the target's server or network. Such attacks can be detected by noticing a change in the average number of packets sent through the victim's link per unit time. Intuitively, it is appealing to formulate the problem of detecting DDoS as a *quickest changepoint detection* problem. That is, to detect changes in statistical models as rapidly as possible (with minimal *average delays*) while maintaining the FAR at a given low level.

Previous publications (Polunchenko *et al.*, 2012; Tartakovsky *et al.*, 2006a,b, 2013) have showed that certain quickest changepoint detection methods (Basseville and Nikiforov, 1993; Lai, 1998; Pollak and Tartakovsky, 2009) can be effectively used for designing AbIDSs for the early detection of intrusions in high-speed computer networks. Changepoint detection theory allows us to develop solutions that are easily implemented and have certain optimality properties.

However, AbIDSs and SbIDSs are clearly complementary, and neither alone is sufficient to detect and isolate the anomalies generated by attacks or non-malicious events. The reason is that both these types of IDSs, when working independently, are plagued by a high rate of false positives and are susceptible to carefully crafted attacks that "blend" themselves into normal traffic. The ability of changepoint detection techniques to run at high speeds and with small detection delays presents an interesting opportunity. What if one could combine these techniques with signature-type methods that offer very low FAR but are too heavy to use at line speeds? Do such synergistic IDSs exist, and if so, how can they be integrated?

Such an approach is explored in this chapter. The main idea is to integrate two substantially different detection techniques – anomaly changepoint detection-based methods and signature–spectral detection techniques. We demonstrate that the resulting hybrid anomaly–signature IDS is synergistic and performs better than any individual system. More specifically, the resulting IDS is based on a two-stage (cascade) *hybrid* approach that combines changepoint AbIDS and "flow-based" SbIDS to simultaneously improve detection performance and lower FAR. This two-stage hybrid approach allows augmenting hard detection decisions with profiles that can be used for further analysis, e.g., for filtering false positives and confirming real attacks both at single-sensor and network levels. The hybrid IDS is tested on real attacks, and the results demonstrate the benefits of carefully combining anomaly and signature IDSs.

The chapter is organized as follows. In Section 2.2 we outline certain theoretical aspects of quickest detection methods. In Section 2.3 we transition from changepoint detection theory to cyber-security and formulate the principles of the anomaly IDS. In Section 2.4 we describe the novel hybrid anomaly–signature IDS that integrates anomaly- and signature-based detection systems and allows for efficient false alarm filtering and true attack confirmation. In Subsections 2.3.2 and 2.4.2 we present the results of experimental studies for real-world data with several challenging attacks.

2.2. Quickest Changepoint Detection

Computer intrusions produce either abrupt or at least relatively abrupt changes in network traffic. The observations are obtained sequentially and, as long as their behavior is consistent with the "normal state," one is content to let the process continue. If the state changes, one should detect the change as soon as possible. In other words, the change occurs at an unknown instant, and the practitioners' goal is to detect it as quickly as possible while avoiding frequent false alarms. Evidently, the desire to detect a change quickly causes one to be trigger-happy, which brings about many false alarms if there is no change. On the other hand, attempting to avoid false alarms too strenuously will lead to a long delay between the time of occurrence of a real change and its detection. Thus, the design of the quickest changepoint detection procedures involves optimizing the *tradeoff* between two kinds of performance measures, one being a measure of detection delay, the other a measure of the false alarm frequency. A good decision procedure depends on what is known about the stochastic behavior of the observations, both pre- and post-change.

These ideas can be used for designing an efficient AbIDS based on changepoint detection schemes. See Section 2.3 for a detailed discussion.

We now provide a brief overview of changepoint detection approaches and two main competing detection procedures – CUSUM (Cumulative Sum) and the Shiryaev–Roberts procedure.

Suppose a series $\{X_n\}_{n\geq 1}$ of random variables, not necessarily independent and identically distributed (i.i.d.), is observed sequentially, in a one-at-a-time manner. At first, the series is "in-control" and each X_n is distributed according to conditional density $f(X_n|\mathbf{X}_1^{n-1})$, the *pre-change* density, where we used the notation $\mathbf{X}_1^{\ell} = (X_1, X_2, \ldots, X_\ell)$ for the vector of first ℓ observations. At an unknown time instant ν something unusual

happens and the series "runs out of control" by altering its statistical properties so that beginning from the time moment $\nu+1$ the conditional density of each X_n is $g(X_n|\mathbf{X}_1^{n-1})$, the *post-change* density. Note that in general these densities may depend on n, and the post-change density may also depend on the changepoint ν, i.e., $f(X_n|\mathbf{X}_1^{n-1}) = f_n(X_n|\mathbf{X}_1^{n-1})$ for $n \geq 1$ and $g(X_n|\mathbf{X}_1^{n-1}) = g_{n,\nu}(X_n|\mathbf{X}_1^{n-1})$ for $n > \nu$. After the change occurs, an alarm should be raised as soon as possible and with few false detections so that an appropriate action is taken.

Statistically, having observed the sample $\mathbf{X}_1^n = (X_1, \ldots, X_n)$, the problem is to test the hypothesis $\mathsf{H}_k\colon \nu = k \geq 0$ that the change has occurred at time k somewhere in the starch of n observations against the alternative $\mathsf{H}_\infty\colon \nu = \infty$ that there is never a change. The densities of $\mathbf{X}_1^n$ given the hypotheses H_k and H_∞ are

$$p(\mathbf{X}_1^n|\mathsf{H}_k) = \prod_{i=1}^{k} f(X_i|\mathbf{X}_1^{i-1}) \times \prod_{i=k+1}^{n} g(X_i|\mathbf{X}_1^{i-1}),$$

$$p(\mathbf{X}_1^n|\mathsf{H}_\infty) = \prod_{i=1}^{n} f(X_i|\mathbf{X}_1^{i-1}).$$

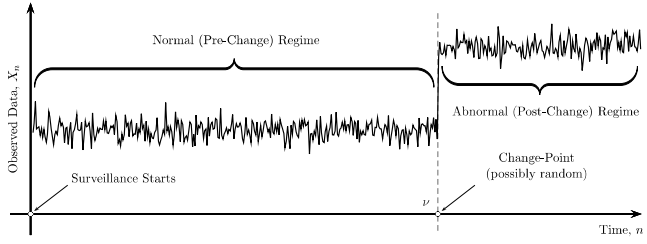
According to hypothesis testing theory an optimal solution should be based on the likelihood ratio (LR) between these hypotheses given by

$$\Lambda_n^k = \frac{p(\mathbf{X}_1^n|\mathsf{H}_k)}{p(\mathbf{X}_1^n|\mathsf{H}_\infty)} = \prod_{j=k+1}^{n} L_j \quad \text{for} \quad k < n, \quad L_j = \frac{g(X_j|\mathbf{X}_1^{j-1})}{f(X_j|\mathbf{X}_1^{j-1})}, \tag{2.1}$$

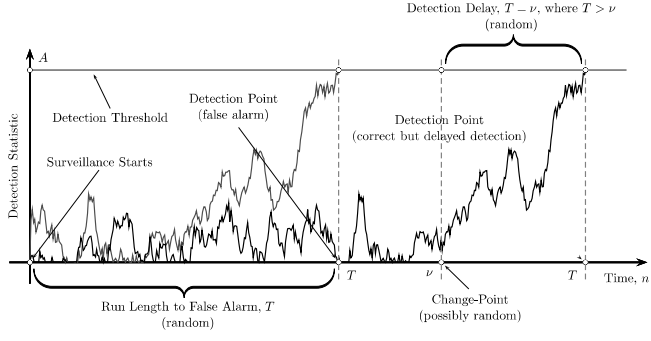
where $\prod_{j=\ell}^{n} L_j = 1$ for $\ell > n$.

In the simplest setting, it is assumed that the observations are *independent* throughout the entire period of surveillance, so that $X_1, X_2, \ldots, X_\nu$ are distributed according to a common pre-change density f, while $X_{\nu+1}, X_{\nu+2}, \ldots$ have a common density $g \not\equiv f$. We refer to this case as the *i.i.d. case*. Note that due to the i.i.d. assumption, the LR for the nth observation simplifies to $L_n = g(X_n)/f(X_n)$.

The LRs $\{\Lambda_n^k\}_{n\geq 1}$ are then used by a *sequential detection procedure* to decide in favor of one of the hypotheses. Given the sequence $\{X_n\}_{n\geq 1}$, a sequential detection procedure is defined as a *stopping time* T adapted to the observations, i.e., the event $\{T \leq n\}$ is measurable with respect to $X_1, \ldots, X_n$. The stopping time T is nothing but the time instant at which the detection procedure stops and declares that a change is in effect. If the change is declared prematurely, i.e., $T \leq \nu$, then a *false alarm* is



(a) Process of interest with a change in the mean.



(b) Two possible detection scenarios: false alarm (gray) and correct detection (black).

Fig. 2.1. Illustration of sequential changepoint detection.

raised. Figure 2.1(a) shows a typical example of change in the mean of the process and its detection. The gray trajectory in Figure 2.1(b) illustrates the false alarm situation when the detection statistic exceeds the detection threshold prior to the change occurring, in which case T can be regarded as the (random) run length to the false alarm. The black trajectory in Figure 2.1(b) illustrates true detection when the detection statistic exceeds the detection threshold past the changepoint. Note that the detection delay characterized by the difference $T - \nu$ is random.

There are various formulations of the optimum tradeoff problem, which differ in the way the terms "detection delay" and "false alarm rate" are defined (see Polunchenko and Tartakovsky (2012) for a detailed overview). We now outline two optimality approaches that are of greatest interest for our purposes.

A *minimax formulation* was proposed by Lorden's (1971) and later by Pollak (1985), who suggested a different measure of detection speed. This formulation regards the changepoint ν as *unknown*, but *not random*. The

goal is to minimize the worst-case average delay to detection, subject to a lower bound on the mean time to false alarm. The second is a *Bayesian formulation*, introduced by Shiryaev (1963). In contrast to the minimax formulation, the Bayesian formulation assumes that the changepoint ν is a *random variable* with a *known* (prior) distribution. The objective is to minimize the expected delay, subject to an upper bound on the weighted false alarm probability. Since the prior distribution of the point of occurrence of network traffic anomaly is not feasible to model, we will assume that the changepoint has improper uniform distribution on the positive line, leading to a generalized Bayesian solution (also related to a more practical setting of detecting changes occurring in a distant future (Pollak and Tartakovsky, 2009)).

Hereafter, let P_k and P_∞ denote the probability measures when a change takes place at $0 \le \nu = k < \infty$ and when no change ever occurs and let E_k and E_∞ be the corresponding expectations.

Lorden's (1971) minimax changepoint detection theory measures the false alarm rate in terms of $\mathsf{E}_\infty T$, the average run length (ARL) to false alarm (ARL2FA). Specifically, define $\mathbf{C}_\gamma = \{T\colon \mathsf{E}_\infty T \ge \gamma\}$ the class of procedures for which the ARL2FA is no less than the desired chosen level $\gamma > 1$. To measure the detection speed, Lorden's (1971) suggested the following "worst-worst-case" essential supremum average detection delay (ESADD)

$$\mathsf{ESADD}(T) = \sup_{0\le\nu<\infty} \operatorname{ess\,sup} \mathsf{E}_\nu[(T-\nu)^+ | X_1, \ldots, X_\nu], \tag{2.2}$$

where $x^+ = \max\{0, x\}$. In other words, the detection delay is maximized over both the changepoint and the trajectory of observations.

A more suitable for practical purposes measure of detection delay was proposed by Pollak (1985), who instead suggested to use

$$\mathsf{SADD}(T) = \sup_{0\le\nu<\infty} \mathsf{E}_\nu(T-\nu | T > \nu), \tag{2.3}$$

i.e., the maximal (supremum) conditional average detection delay (SADD), provided a false alarm has not sounded. In the following this detection speed measure will be used along with the stationary average detection delay (ADD) ($\mathsf{STADD}(T)$) introduced below.

In the minimax problem, the goal is to find procedures that would minimize $\mathsf{ESADD}(T)$ and $\mathsf{SADD}(T)$ subject to $\mathsf{E}_\infty T \ge \gamma$, i.e., in the class $\mathbf{C}_\gamma$.

Note that ideally one would prefer to find a procedure that would minimize $\mathsf{E}_\nu(T-\nu|T>\nu)$ for all $\nu \geq 0$, when the frequency of false alarms is kept at a desired level. However, no such uniformly optimal procedure exists in the class $\mathbf{C}_\gamma$, so we have to resort to minimax or other settings.

In the i.i.d. setting, Moustakides (1986) showed that Page's (1954) CUSUM procedure is strictly minimax with respect to Lorden's measure of delay to detection ESADD given by (2.2). The CUSUM procedure is based on a maximum likelihood principle: $\max_{k\geq 0}\Lambda_n^k$, which leads (after taking logarithms) to the detection statistic

$$W_n = (0, W_{n-1}+Z_n)^+, \quad n \geq 1, \quad W_0 = 0, \tag{2.4}$$

where, as usual, $x^+ = \max(0,x)$ denotes a positive part of x, $W_n = \log(\max_{k\geq 0}\Lambda_n^k)$ and $Z_n = \log L_n$ is the log-likelihood ratio (LLR) for the nth observation. The procedure stops and declares that the change has occurred at

$$T_{\mathrm{CS}}(h) = \min\{n \geq 1\colon W_n \geq h\}, \quad \min\{\varnothing\} = \infty, \tag{2.5}$$

i.e., as soon as the detection statistic W_n exceeds h, a positive detection threshold preset so as to achieve the desired level of false alarms $\gamma > 1$. This can be done by choosing $h = h_\gamma = \log\gamma$ since $\mathsf{E}_\infty T_{\mathrm{CS}}(h) \geq e^h$ for any $h > 0$ (Lorden's, 1971). This choice of the detection threshold guarantees $T_{\mathrm{CS}}(h) \in \mathbf{C}_\gamma$, but this lower bound is extremely conservative. For relatively large values of γ a connection of h and γ is given in Tartakovsky (2005): $\mathsf{E}_\infty T_{\mathrm{CS}}(h) \sim v^{-1}e^h$ as $h \to \infty$, where $0 < v < 1$ is a computable constant depending on the model. Thus, setting $h = \log(v\gamma)$ guarantees $\mathsf{E}_\infty T_{\mathrm{CS}}(h) \approx \gamma$. Hereafter $x_b \sim y_b$ means that $\lim_{b\to\infty}(x_b/y_b) = 1$, which can also be written as $x_b = y_b(1+o(1))$, where $o(1) \to 0$ as $b \to \infty$.

A competitive detection procedure, the so-called Shiryaev–Roberts (SR) procedure, is based on the averaging of the likelihood ratio over the uniform prior distribution rather than maximizing it over the unknown changepoint. Specifically, define

$$R_n = \sum_{k=1}^{n}\prod_{i=k}^{n} L_i, \quad T_{\mathrm{SR}}(A) = \min\left\{n \geq 1\colon R_n \geq A\right\}, \tag{2.6}$$

where A is a positive threshold that controls the FAR. A connection between A and the ARL2FA $\mathsf{E}_\infty T_{\mathrm{SR}}(A)$ is given in Pollak (1987): $\mathsf{E}_\infty T_{\mathrm{SR}}(A) \geq A$ for every $A > 0$ and $\mathsf{E}_\infty T_{\mathrm{SR}}(A) \sim \zeta^{-1}A$ as $A \to \infty$, where the constant $0 < \zeta < 1$ is subject of renewal theory. Hence, taking $A_\gamma = \gamma\zeta$ yields $\mathsf{ARL2FA}(T_{\mathrm{SR}}) \approx \gamma$ for sufficiently large γ.

Note the recursion

$$R_{n+1} = (1 + R_n)L_{n+1}, \quad n \geq 0, \quad R_0 = 0 \tag{2.7}$$

(with the null initial condition).

Pollak (1985) tweaked the procedure by starting it off at a random $R_0^{\mathsf{Q}_A}$ whose distribution is the quasi-stationary distribution $\mathsf{Q}_A = \lim_{n\to\infty} \mathsf{P}_\infty(R_n \leq x | T_{\mathrm{SR}}(A) > n)$ of the SR statistic R_n and showed that the detection procedure that stops and raises an alarm at the first time when the statistic $R_n^{\mathsf{Q}_A}$ crosses the level A,

$$T_A^{\mathsf{Q}_A} = \min\left\{n \geq 1\colon R_n^{\mathsf{Q}_A} \geq A\right\}, \tag{2.8}$$

minimizes (asymptotically as $\gamma \to \infty$) to within $o(1)$ the maximal expected delay $\mathsf{SADD}(T)$ over all stopping times that satisfy $\mathsf{E}_\infty T \geq \gamma$, where A is such that $\mathsf{E}_\infty T_A^{\mathsf{Q}_A} = \gamma$. We will refer to this randomized SR procedure as the Shiryaev–Roberts–Pollak (SRP) procedure.

Until recently, the question of whether the SRP procedure is exactly minimax with respect to Pollak's expected delay measure $\mathsf{SADD}(T)$ (in the class of procedures $\mathbf{C}_\gamma$) was open. Moustakides *et al.* (2011) presented numerical evidence that uniformly better procedures exist. They proposed to start the original SR procedure at a fixed (but specially designed) point $R_0^r = r$, $0 \leq r < A$, showing that, for certain values of r, the expected conditional delay $\mathsf{E}_\nu(T_{A_*}^r - \nu | T_{A_*}^r > \nu) < \mathsf{E}_\nu(T_A^{\mathsf{Q}_A} - \nu | T_A^{\mathsf{Q}_A} > \nu)$ for all $\nu \geq 0$, where A_* and A are such that $\mathsf{E}_\infty T_A^{\mathsf{Q}_A} = \mathsf{E}_\infty T_{A_*}^r$. We will refer to the procedure T_A^r that starts from r to as the SR$-r$ procedure. Tartakovsky *et al.* (2012) showed that the SR$-r$ procedure with a specially designed $r = r(\gamma)$ is third-order asymptotically minimax (i.e., to within $o(1)$) in the class of procedures $\mathbf{C}_\gamma$ with $\mathsf{E}_\infty T \geq \gamma$ as $\gamma \to \infty$. Polunchenko and Tartakovsky (2010, 2012) provided examples where the SR$-r$ procedure is strictly minimax.

In a variety of surveillance applications, including intrusion detection, the detection procedure should be applied *repeatedly*. This requires specification of a renewal mechanism after each alarm (false or true). The simplest renewal strategy is to restart from scratch, in which case the procedure becomes multi-cyclic with similar cycles (in a statistical sense) if the process is homogeneous. Furthermore, the most interesting scenario for our applications is when a change (attack) occurs at a distant time horizon, i.e., long after surveillance begins. This naturally leads to a problem of detecting a distant change in a stationary background of false alarms (stationary

regime), assuming the detection procedure is applied anew after each time the detection statistic exceeds the threshold. Pollak and Tartakovsky (2009) showed that if a change takes place after many successive re-runs of a stopping time T, the expected delay is minimized asymptotically as $\nu \to \infty$ over all multi-cyclic procedures with $\mathsf{E}_\infty T \geq \gamma$ for every $\gamma > 1$ by the original (multi-cyclic) SR procedure. In other words, the SR procedure is strictly optimal in the i.i.d. case with respect to the stationary average delay to detection (STADD) given by

$$\mathsf{STADD}(T) = \lim_{\nu\to\infty} \mathsf{E}_\nu(N_1 + N_2 + \cdots + N_{J_\nu} - \nu) \tag{2.9}$$

for every $\gamma > 1$. Since computer intrusions most likely start long after surveillance begins, the SR type statistics are preferable to CUSUM. Figure 2.2 illustrates this multi-cyclic scenario, where $N_1, N_2, \ldots$ are sequential independent repetitions of a stopping time T, e.g., the CUSUM or SR detection algorithm. There are multiple false alarms and the detection statistic is renewed from scratch each time. Note also that recent research shows that the difference in performance between SR and CUSUM is visible only when detecting dim changes (Tartakovsky *et al.*, 2012), as we also verified in our experiments with real attacks.

For the purpose of illustration, Figure 2.3 compares the behavior of the CUSUM and SR statistics for a simulated trajectory from the Gaussian i.i.d. model with a change in the mean from 0 to 1 and standard deviation 1. The thresholds in both procedures were selected to guarantee the same ARL2FA. The change occurs at $\nu = 200$, i.e., relatively far from the beginning. In this particular case, the CUSUM statistic exits over the threshold a little later than the SR statistic, as expected from our previous discussion. We plot $\log(R_n)$ to be on the same scale with CUSUM W_n.

The plots of the conditional average delay to detection $\mathsf{ADD}_\nu(T) = \mathsf{E}_\nu(T - \nu | T > \nu)$ versus the changepoint ν for several detection procedures described above are shown in Figure 2.4. These plots were obtained solving

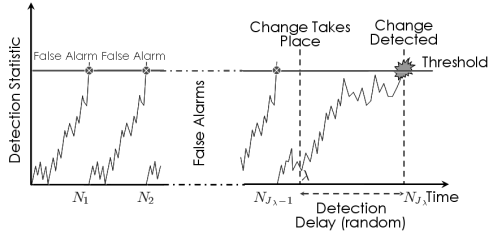


Fig. 2.2. Illustration of the typical multi-cyclic surveillance scenario.

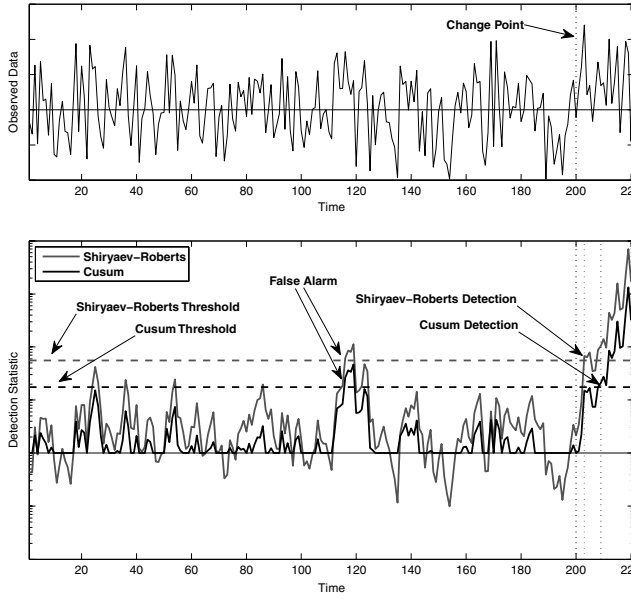


Fig. 2.3. The CUSUM statistic and log of the SR statistic for the Gaussian model (pre-change mean $\mu = 0$, post-change mean $\theta = 1$, standard deviation $\sigma = 1$, changepoint $\nu = 200$).

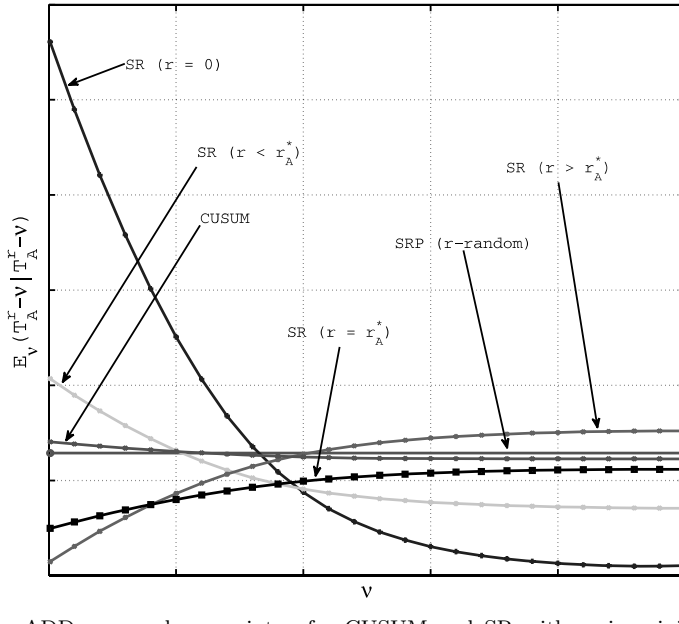


Fig. 2.4. ADD versus changepoint ν for CUSUM and SR with various initialization strategies.

numerically integral equations for operating characteristics (Moustakides *et al.*, 2011). The SRP and SR-r procedures outperform both SR and CUSUM with respect to SADD. However, this comes with the additional effort of finding an appropriate initialization – not a trivial task. Besides, the difference is not dramatic. For this reason, in this chapter, we will focus on the "pure" CUSUM and SR procedures with zero initialization as well as on their semiparametric and nonparametric modifications more suitable for intrusion detection applications. Note also that the SR procedure outperforms all its counterparts when the change occurs at a distant time horizon.

The following theorem, whose proof may be found in Tartakovsky *et al.* (2014, Ch. 9), establishes asymptotic properties of the CUSUM and SR procedures for the low FAR (large γ, i.e., as $h, A \to \infty$). We need additional notation: $\lambda_n = \sum_{i=1}^n Z_i$,

$$\tau_a = \min\{n : \lambda_n \geq a\}, \quad \varkappa = \lim_{a\to\infty} \mathsf{E}_0(\lambda_{\tau_a} - a), \quad \zeta = \lim_{a\to\infty} \mathsf{E}_0 e^{-(\lambda_{\tau_a}-a)},$$

$$\mathsf{Q}^{\mathrm{CS}}(y) = \mathsf{P}_0\left(\min_{n\geq 0}\lambda_n \leq y\right), \quad \mathsf{Q}^{\mathrm{SR}}(y) = \mathsf{P}_0\left\{\log\left(1+\sum_{n=1}^{\infty} e^{-\lambda_n}\right) \leq y\right\},$$

$$\mathsf{Q}^{\mathrm{CS}}_{\mathrm{st}}(y) = \lim_{n\to\infty} \mathsf{P}_\infty(W_n \leq y), \quad \mathsf{Q}^{\mathrm{SR}}_{\mathrm{st}}(y) = \lim_{n\to\infty} \mathsf{P}_\infty(R_n \leq y),$$

$$C_0^{\mathrm{CS}} = -\int_{-\infty}^0 y\,\mathrm{d}\mathsf{Q}^{\mathrm{CS}}(y), \quad C_0^{\mathrm{SR}} = \int_0^\infty y\,\mathrm{d}\mathsf{Q}^{\mathrm{SR}}(y),$$

$$C_\infty^{\mathrm{CS}} = -\int_0^\infty\int_{-\infty}^{-z} y\,\mathrm{d}\mathsf{Q}^{\mathrm{CS}}(y)\mathrm{d}\mathsf{Q}^{\mathrm{CS}}_{\mathrm{st}}(z), \quad C_\infty^{\mathrm{SR}} = \int_0^\infty\int_0^z y\,\mathrm{d}\mathsf{Q}^{\mathrm{SR}}(y)\mathrm{d}\mathsf{Q}^{\mathrm{SR}}_{\mathrm{st}}(z).$$

Define also the Kullback–Leibler (KL) information number

$$I = \mathsf{E}_0 Z_1 = \int \log\left(\frac{g(x)}{f(x)}\right) g(x)\,\mathrm{d}\mu(x).$$

Theorem 2.1. *Consider the i.i.d. case. Assume that* $\mathsf{E}_0|Z_1|^2 < \infty$ *and that* Z_1 *is* P_0*-nonarithmetic.*

(i) *If* $h = \log(I\zeta^2\gamma)$, *then* $\mathsf{ARL2FA}(T_{\mathrm{CS}}) \sim \gamma$ *as* $\gamma \to \infty$ *and*

$$\mathsf{SADD}(T_{\mathrm{CS}}) = \frac{1}{I}(\log\gamma + C_{\mathrm{CS}}) + o(1),$$

$$\mathsf{STADD}(T_{\mathrm{CS}}) = \frac{1}{I}(\log\gamma + \widetilde{C}_{\mathrm{CS}}) + o(1),$$

where $C_{\mathrm{CS}} = \varkappa - \log(1/I\zeta^2) - C_0^{\mathrm{CS}}$ *and* $\widetilde{C}_{\mathrm{CS}} = \varkappa - \log(1/I\zeta^2) - C_\infty^{\mathrm{CS}}$.

(ii) *If* $A = \log(\zeta\gamma)$, *then* $\mathsf{ARL2FA}(T_{\mathrm{SR}}) \sim \gamma$ *as* $\gamma \to \infty$ *and*

$$\mathsf{SADD}(T_{\mathrm{SR}}) = \frac{1}{I}(\log\gamma + C_{\mathrm{SR}}) + o(1),$$
$$\mathsf{STADD}(T_{\mathrm{SR}}) = \frac{1}{I}(\log\gamma + \widetilde{C}_{\mathrm{SR}}) + o(1),$$

where $C_{\mathrm{SR}} = \varkappa - \log(1/\zeta) - C_0^{\mathrm{SR}}$ *and* $\widetilde{C}_{\mathrm{CS}} = \varkappa - \log(1/\zeta) - C_\infty^{\mathrm{SR}}$.

Neglecting the constants yields

$$\mathsf{SADD}(T_{\mathrm{CS}}) \approx \mathsf{SADD}(T_{\mathrm{SR}}) \approx \mathsf{STADD}(T_{\mathrm{CS}}) \approx \mathsf{STADD}(T_{\mathrm{SR}}) \approx \frac{\log\gamma}{I}.$$

The latter approximation is not especially accurate but can still be used as a preliminary estimate for average detection delays.

It is worth mentioning that the i.i.d. assumption is rather restrictive for intrusion detection applications where the observed data is usually correlated and non-stationary, even "bursty" due to substantial temporal variability. Recent advances in general changepoint detection theory imply that the CUSUM and SR procedures are asymptotically optimal for general non-i.i.d. statistical models when the FAR is low ($\gamma \to \infty$) (Fuh, 2003, 2004; Lai, 1998; Tartakovsky and Veeravalli, 2004, 2005). Specifically, if we assume that

$$\lim_{n\to\infty} \frac{1}{n}\mathsf{E}_k[\log\Lambda_{k+n}^k] = I \quad \text{for all } k \geq 0,$$

where I is a positive and finite number (a prototype of the KL number), and the strong law of large numbers (SLLN) holds for the LLR, i.e.,

$$\frac{1}{n}\log\Lambda_{k+n}^k \xrightarrow[n\to\infty]{\mathsf{P}_k-\text{a.s.}} I \quad \text{for all } k \geq 0$$

and additionally will require a certain rate of convergence in the SLLN (cf. Tartakovsky *et al.*, 2014), then both detection procedures, CUSUM and SR, with thresholds $h_\gamma = \log\gamma$ and $A_\gamma = \gamma$ are asymptotically first-order minimax:

$$\inf_{T\in\mathbf{C}_\gamma} \mathsf{SADD}(T) \sim \mathsf{SADD}(T_{\mathrm{CS}}) \sim \mathsf{SADD}(T_{\mathrm{SR}}) \sim \frac{\log\gamma}{I} \quad \text{as } \gamma \to \infty.$$

(cf. Theorem 1 in Tartakovsky *et al.*, 2006a). The same asymptotic optimality result holds for the stationary average delay to detection, i.e.,

$$\inf_{T\in\mathbf{C}_\gamma} \mathsf{STADD}(T) \sim \mathsf{STADD}(T_{\mathrm{CS}}) \sim \mathsf{STADD}(T_{\mathrm{SR}}) \sim \frac{\log\gamma}{I} \quad \text{as } \gamma \to \infty.$$

Finally, note that typically the post-change distribution is known only up to some unknown parameter $\theta \in \Theta$ (at best), that is $g(X_n|\mathbf{X}_1^n) = g_\theta(X_n|\mathbf{X}_1^n)$. For example, the attack intensity is never known exactly. In this case, $\mathsf{SADD}_\theta(T) = \sup_{\nu\geq 1} \mathsf{E}_{\nu,\theta}(T-\nu|T>\nu)$ depends on this parameter, and the same is true for $\mathsf{STADD}_\theta(T)$. Here $\mathsf{E}_{\nu,\theta}$ is the corresponding expectation operator when the parameter value is θ. Then, the CUSUM and SR procedures tuned to a putative value $\theta = \theta_1$ are optimal or asymptotically optimal only if the true parameter value is θ_1, but they are not optimal for other parameter values.

The two conventional methods of overcoming this parametric uncertainty are either the generalized likelihood ratio (GLR) approach based on the GLR statistic $\sup_{\theta\in\Theta} \Lambda_n^k(\theta)$ or the mixture-based approach based on the weighted LR $\int_\Theta \Lambda_n^k(\theta)\mathrm{d}\pi(\theta)$, where $\pi(\theta)$ is some positive weight (prior distribution) and

$$\Lambda_n^k(\theta) = \prod_{i=k+1}^{n} \frac{g_\theta(X_n|\mathbf{X}_1^n)}{f(X_n|\mathbf{X}_1^n)}, \quad k<n.$$

Using the GLR statistic in CUSUM leads to the generalized CUSUM procedure

$$T_{\mathrm{GCS}}(A) = \min\left\{n\geq 1 : \max_{0\leq k\leq n} \sup_{\theta\in\Theta} \Lambda_n^k(\theta) \geq A\right\}, \tag{2.10}$$

while using the weighted LR statistic in SR leads to the weighted (mixture) SR procedure

$$T_{\mathrm{WSR}}(A) = \min\left\{n\geq 1 : \sum_{k=0}^{n-1} \int_\Theta \Lambda_n^k(\theta)\mathrm{d}\pi(\theta) \geq A\right\}. \tag{2.11}$$

Selecting $A=\gamma$ guarantees that both procedures belong to the class $\mathbf{C}_\gamma$, i.e., the ARL2FA for both procedures is lower-bounded by γ for every $\gamma>1$. With this threshold, both detection procedures are uniformly first-order asymptotically minimax under quite general conditions: as $\gamma\to\infty$ for all $\theta\in\Theta$

$$\inf_{T\in\mathbf{C}_\gamma} \mathsf{SADD}_\theta(T) \sim \mathsf{SADD}_\theta(T_{\mathrm{GCS}}) \sim \mathsf{SADD}_\theta(T_{\mathrm{WSR}}) \sim \frac{\log\gamma}{I_\theta},$$

where $I_\theta = \lim_{n\to\infty} \frac{1}{n}\mathsf{E}_{0,\theta}[\log\Lambda_n^0]$ (cf. Tartakovsky *et al.*, 2014).

In particular, consider the i.i.d. case and a multivariate exponential family of distributions with density (with respect to some non-degenerate

σ-finite measure $\mu(\mathrm{d}x)$)

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \exp\left\{\boldsymbol{\theta}^\top \mathbf{x} - \psi(\boldsymbol{\theta})\right\}, \quad \left\{\boldsymbol{\theta} \in \Theta : \psi(\boldsymbol{\theta}) = \log \int_\Theta e^{\boldsymbol{\theta}^\top \mathbf{x}}\, \mu(\mathrm{d}\mathbf{x}) < \infty\right\},$$

assuming that both $\mathbf{X} = (X_1, \ldots, X_\ell) \in \mathbb{R}^\ell$ and $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_\ell) \in \Theta \subset \mathbb{R}^\ell$ are ℓ-dimensional vectors. The pre-change parameter is $\boldsymbol{\theta} = 0$ and the post-change parameter is $\boldsymbol{\theta} \in \Theta_\varepsilon = \Theta - B_\varepsilon$, where B_ε is the ball of radius ε in $\mathbb{R}^\ell$. It can be shown (Tartakovsky *et al.*, 2014) that

$$\mathsf{ARL2FA}(T_{\mathrm{GCS}}) \geq h^{-\ell/2} e^h / K_\varepsilon \big(1 + o(1)\big) \quad \text{as } h \to \infty$$

with

$$K_\varepsilon = (2\pi)^{-\ell/2} \int_{\Theta_\varepsilon} \zeta_{\boldsymbol{\theta}} \sqrt{\det[\nabla^2 \psi(\boldsymbol{\theta})]/I_{\boldsymbol{\theta}}^\ell}\, \mathrm{d}\boldsymbol{\theta}.$$

Therefore, setting $h = \log[K_\varepsilon \gamma (\log \gamma)^{\ell/2}\ ]$ yields $\mathsf{ARL2FA}(T_{\mathrm{GCS}}) \geq \gamma(1 + o(1))$ and, as $\gamma \to \infty$,

$$\mathsf{SADD}_{\boldsymbol{\theta}}(T_{\mathrm{GCS}}) = \frac{1}{I_{\boldsymbol{\theta}}} \left(\log \gamma + \frac{\ell}{2} \log\log \gamma + C_{\mathrm{GCS}}(\boldsymbol{\theta})\right) + o(1),$$

where

$$C_{\mathrm{GCS}}(\boldsymbol{\theta}) = -\frac{\ell}{2}[1 + \log(2\pi)] + \log\left(\int_{\Theta_\varepsilon} \zeta_t \sqrt{\det[\nabla^2 \psi(t)]/I_t^\ell}\, \mathrm{d}t\right) - \mu_{\boldsymbol{\theta}} + \varkappa_{\boldsymbol{\theta}}$$

(cf. Tartakovsky *et al.*, 2014). Here $I_{\boldsymbol{\theta}} = \boldsymbol{\theta}^\top \nabla\psi(\boldsymbol{\theta}) - \psi(\boldsymbol{\theta})$ is the KL number, ∇ is the gradient, ∇^2 is the Hessian, $\mu_{\boldsymbol{\theta}} = \mathsf{E}_{\boldsymbol{\theta}}[\min_{n \geq 0} \lambda_n(\boldsymbol{\theta})]$, $\lambda_n(\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \sum_{k=1}^n \mathbf{X}_k - \psi(\boldsymbol{\theta}) n$ and $\varkappa_{\boldsymbol{\theta}} = \lim_{a \to \infty} \mathsf{E}_{\boldsymbol{\theta}}[\lambda_{\tau_a}(\boldsymbol{\theta}) - a]$ is the limiting average overshoot in the one-sided test $\tau_a = \min\{n : \lambda_n(\boldsymbol{\theta}) \geq a\}$. A similar approximation holds for the weighted SR procedure with an arbitrary continuous weight (of course with a different constant $C_{\mathrm{WSR}}(\boldsymbol{\theta})$). The second term $\frac{\ell}{2} \log\log \gamma$ that goes to infinity when γ becomes large is the price one pays for prior uncertainty.

While these procedures have a nice uniform asymptotic optimality property, it is difficult to implement them online, since even in the i.i.d. case there is no recursive structure, and there are computational difficulties. Luckily, in most computer intrusion applications the minimal intensity of attacks is rather high. For example, DoS attacks attempt to overflow servers, so a quite intense data flow is transmitted. Then one may tune the IDS to an expected minimal intensity, and the real intensity will be typically much higher, in which case further optimization is unnecessary.

2.3. Anomaly-based IDS

2.3.1. *CUSUM and SR score-based algorithms*

Changepoint detection theory is straightforward to implement in the context of intrusion detection in computer networks. As discussed above, network anomalies take place at unknown points in time and produce abrupt changes in statistical properties of traffic data. In network monitoring, one can observe various informative features from packet headers, e.g., packet size, source IP address, destination IP address, source port, destination port, types of protocols (e.g., ICMP, UDP, TCP), etc. In the case of user diagram protocol (UDP) flooding attacks, potentially useful observables include packet sizes, source ports, destination ports, and destination prefix. In the case of transmission control protocol (TCP) flooding attacks, conceivably we could have multiple channels that record counts of different flags (SYN, ACK, PUSH, RST, FIN, URG) from TCP header. Another plausible observable is a number of half-open connections for the detection of SYN flooding attacks. We could also have channels that keep track of the discrepancies in TCP SYN-FIN or TCP SYN-RST pairs. Furthermore, in order to detect file-sharing, we could monitor arrival (e.g., packet, byte or flow) counts, port numbers, and source-destination prefixes.

Hence, we consider the problem of anomaly detection in computer networks a quickest changepoint detection problem: to detect changes in network traffic as rapidly as possible while maintaining a tolerable level of false alarms.

In network security, however, the behavior of both pre- and post-attack traffic is poorly understood; as a result, typically neither the pre- nor post-change distributions are known. Consequently, one can no longer rely on the LR (2.1), demanding an alternative approach based on replacing the LR with appropriate statistics that we refer to as scores. To understand the idea behind constructing score-based statistics, consider the typical behavior of the LR-based CUSUM and SR procedures. It suffices to consider CUSUM W_n since $\log R_n$ evolves similarly. As long as the observed process $\{X_n\}_{n\geq 1}$ is "in-control", W_n fluctuates not far from the zero reflection barrier as if it were "afraid" of approaching the detection threshold h, although W_n could cross the threshold sooner or later, raising a false alarm. However, as soon as $X_{\nu+1}$, the first "out-of-control" measurement, is observed, the behavior of W_n makes a complete 180° turn – now it eagerly tries to hit the level h. Figures 2.2 and 2.3 illustrate this typical behavior, guaranteed by the fact that $\mathsf{E}_\infty Z_n < 0$, i.e., the detection statistic has a negative drift in the

normal regime, while $\mathsf{E}_\nu Z_n > 0$ for $\nu < n$, i.e., the drift is positive in the abnormal regime.

That said, consider the following score-based modification of the CUSUM and SR algorithms

$$W_n^{\mathrm{sc}} = \max\{0, W_{n-1}^{\mathrm{sc}} + S_n\}, \quad T_{\mathrm{CS}}^{\mathrm{sc}} = \min\{n \geq 1 : W_n^{\mathrm{sc}} \geq h\} \tag{2.12}$$

and

$$R_n^{\mathrm{sc}} = (1 + R_{n-1}^{\mathrm{sc}})e^{S_n}, \quad T_{\mathrm{SR}}^{\mathrm{sc}} = \min\{n \geq 1 : R_n^{\mathrm{sc}} \geq A\}, \tag{2.13}$$

where $W_0^{\mathrm{sc}} = 0 = R_0^{\mathrm{sc}}$ and $h, A > 0$ are *a priori* chosen detection thresholds which determine FAR. Here $S_n(\mathbf{X}_1^n)$ is a score function sensitive to a change. Clearly, as long as the score function has negative pre-change mean $\mathsf{E}_\infty S_n < 0$ and positive post-change mean $\mathsf{E}_\nu S_n > 0$, the resulting score-based (semiparametric or nonparametric) CUSUM and SR algorithms will work, though they are no longer guaranteed to be optimal.

Let Q be a positive and finite number and assume that

$$\lim_{n\to\infty} \frac{1}{n} \mathsf{E}_\nu \left[\sum_{i=\nu+1}^{\nu+n} S_i \right] = Q \quad \text{for all } \nu \geq 0.$$

Further, assume that the SLLN holds for the score S_n:

$$\frac{1}{n} \sum_{i=\nu+1}^{\nu+n} S_i \xrightarrow[n\to\infty]{\mathsf{P}_\nu-\text{a.s.}} Q \quad \text{for all } \nu \geq 0.$$

If, in addition, we postulate a certain rate of convergence in the SLLN, it can be shown that

$$\mathsf{SADD}(T_{\mathrm{CS}}^{\mathrm{sc}}) \sim \mathsf{STADD}(T_{\mathrm{CS}}^{\mathrm{sc}}) \sim h/Q \quad \text{as } h \to \infty, \tag{2.14}$$

and similar asymptotic approximations hold for the score-based SR procedure with h replaced by $\log A$. See Theorem 3 in Tartakovsky *et al.* (2006a). In general, however, it is impossible to approximate ARL2FA unless S_n is connected to the LLR. So it is unclear how to select thresholds h and A to guarantee the given FAR level. In general, Monte Carlo simulations seem to be the only way.

The score function S_n can be chosen in a number of ways, each particular choice depending crucially on the expected type of change. For example, detecting a shift in the mean value and a change in the variance requires considering different score functions. In the applications of interest, the problem

can be usually reduced to detecting changes in mean values or in variance or in both mean and variance. In Tartakovsky *et al.* (2006a,b), a linear memoryless score was proposed for detecting changes in the mean, and in Tartakovsky *et al.* (2013) this score was generalized to linear-quadratic to simultaneously handle changes in both mean and variance.

Specifically, let $\mu_\infty = \mathsf{E}_\infty X_n$, $\sigma^2_\infty = \mathsf{var}_\infty[X_n]$ and $\mu = \mathsf{E}_0 X_n$, $\sigma^2 = \mathsf{var}_0[X_n]$ denote the pre- and post-anomaly mean values and variances, respectively. Write $Y_n = (X_n - \mu_\infty)/\sigma_\infty$ for the centered and scaled observation at time n. In the real-world applications, the pre-change parameters μ_∞ and σ^2_∞ are estimated from the training data and periodically re-estimated due to the non-stationarity of network traffic; they can therefore be assumed known. Introduce the following memoryless linear-quadratic score

$$S_n(Y_n) = C_1 Y_n + C_2 Y_n^2 - C_3, \tag{2.15}$$

where C_1, C_2 and C_3 are non-negative design numbers, assuming for concreteness that the change leads to an increase in both mean and variance. In the case where the variance either does not change or changes relatively insignificantly compared to the change in the mean, the coefficient C_2 may be set to zero. In the opposite case where the mean changes only slightly compared to the variance, we may take $C_1 = 0$. The first linear case is typical for many cyber-security applications such as internet control message protocol (ICMP) and UDP DDoS attacks. However, in certain cases, such as the TCP SYN attacks considered in Polunchenko *et al.* (2012) and Tartakovsky *et al.* (2013), both the mean and variance change significantly.

Note that the score given by (2.15) with

$$C_1 = \delta q^2, \quad C_2 = (1 - q^2)/2, \quad C_3 = \delta^2 q^2/2 - \log q, \tag{2.16}$$

where $q = \sigma_\infty/\sigma$, $\delta = (\mu - \mu_\infty)/\sigma_\infty$, is optimal if pre- and post-change distributions are Gaussian with known putative values μ and σ^2. This is true because in the latter case S_n is the log-likelihood ratio for the nth observation. If one accepts the Gaussian model (which is sometimes the case), it follows from the discussion in Section 2.2 that selecting $q = q_0$ and $\delta = \delta_0$ with some design values q_0 and δ_0 provides reasonable operating characteristics for $q < q_0$ and $\delta > \delta_0$ and optimal characteristics for $q = q_0$ and $\delta = \delta_0$. However, it is important to emphasize that the proposed score-based CUSUM and SR procedures do not assume that the observations have Gaussian (or any other) pre- and post-change distributions.

Further improvement may be achieved by using either mixtures or adaptive versions with generalized likelihood ratio-type statistics similar to (2.10) – (2.11). Also, an improvement can be obtained by running several CUSUM (or SR) algorithms in parallel, each tuned to its own value of (q, δ). This multichart CUSUM and SR procedures are robust and very efficient (Tartakovsky and Polunchenko, 2007, 2008).

In Tartakovsky *et al.* (2006a,b), we conjectured that in certain conditions splitting packets in "bins" and considering multichannel detectors helps localize and detect attacks more rapidly. Consider the multichannel scenario where the vector data $X_n^{(1)}, \ldots, X_n^{(N)}$ are used to decide on the presence of anomalies. Here $X_n^{(i)}$ is a sample obtained at time n in the ith channel. For example, in the case of UDP flooding attacks the channels correspond to packet sizes (size bins), while for TCP SYN attacks they correspond to IP addresses (IP bins).

Similarly to the single-channel case, for $i = 1, \ldots, N$, introduce the score functions $S_n^{(i)} = C_1^i Y_n^i + C_2^i (Y_n^i)^2 - C_3^i$ (or any other reasonable scores in channels) and the corresponding score-based CUSUM and SR statistics

$$\begin{aligned} W_n^{(i)} &= \max\left\{0, W_{n-1}^{(i)} + S_n^{(i)}\right\}, \quad W_0^{(i)} = 0; \\ R_n^{(i)} &= (1 + R_n^{(i)}) \exp\left\{S_n^{(i)}\right\}, \quad R_0^{(i)} = 0. \end{aligned} \tag{2.17}$$

Typically, the statistics $W_n^{(i)}$ and $\log R_n^{(i)}$ $(i = 1, \ldots, N)$ remain close to zero in normal conditions; when the change occurs in the jth channel, the jth statistics $W_n^{(j)}$ and $\log R_n^{(j)}$ start rapidly drifting upward. The "MAX" algorithm previously proposed by Tartakovsky *et al.* (2006a,b) is based on the maximal statistic $W_{\max}(n) = \max_{1 \le i \le N} W_n^{(i)}$, which is compared to a threshold h that controls FAR, i.e., the algorithm stops and declares the attack at

$$T_{\max}(h_N) = \min\left\{n \ge 1 : W_{\max}(n) \ge h_N\right\}. \tag{2.18}$$

This method shows very high performance and is the best one can do when attacks are visible in one or very few channels. The latter conclusion can be explained as follows. If the attack is visible in the ith channel (and only in this channel), then analogously to (2.14) the average delay to detection $\mathsf{SADD}_i(T_{\max}) = \sup_{\nu \ge 0} \mathsf{E}_{\nu,i}(T_{\max} - \nu | T_{\max} > \nu)$ is approximated as $\mathsf{SADD}_i(T_{\max}) \approx h_N / Q_i$, where $Q_i = \lim_{n\to\infty} \frac{1}{n} \mathsf{E}_{0,i}\left[\sum_{k=1}^n S_k^{(i)}\right]$ is the "signal-to-noise" ratio (related to the attack intensity relative to the background traffic) in the ith channel. In the N-channel system, the threshold

h_N should be increased roughly by $\log N$ compared with the threshold h in the single-channel system to have about the same FAR (see Lemma 1 in Tartakovsky *et al.* (2006a)). We thus obtain the estimate

$$\mathsf{SADD}_i(T_{\max}) \approx (h + \log N)/Q_i, \quad i = 1, \ldots, N. \tag{2.19}$$

For the sake of concreteness, consider the Gaussian model with a change in the mean $\mu_\infty \to \mu_i$ in the ith channel and constant variance $\sigma^2_\infty = \sigma^2$, in which case the linear score $S_n^{(i)} = \mu_i X_n^{(i)} - \delta_i^2/2$ is optimal, where $\delta_i = \mu_i/\sigma$. Then $Q_i = \delta_i^2/2$. Now, for a single-channel system where all packets are mixed in a single statistic but the attack is only visible in the ith bin, we have

$$\mathsf{SADD}_i(T^{\mathrm{sc}}_{\mathrm{CS}}) \approx h/(\delta_i^2/2N). \tag{2.20}$$

Therefore, for large enough h, which in this case can be taken $h = \log\gamma$, using (2.19) and (2.20), we obtain

$$\frac{\mathsf{SADD}_i(T^{\mathrm{sc}}_{\mathrm{CS}})}{\mathsf{SADD}_i(T_{\max})} \approx N.$$

This estimate is very approximate but shows how poorly a single-channel procedure may perform.

Assuming the attack is visible in many channels, the following "SUM" decision statistic that combines scores from all the channels will be efficient:

$$\overline{W}_n = \max\left\{0, \overline{W}_{n-1} + \sum_{i=1}^{N}\left[S_n^{(i)}\right]^+\right\}, \quad \overline{W}_0 = 0.$$

The detection procedure $T_{\mathrm{SUM}} = \min\{n : \overline{W}_n \geq h\}$ outperforms the previous one when the anomaly due to the attack occurs in many channels.

However, the most general case is where the number of affected channels is *a priori* unknown and may vary from small to large. In this case, the reasonable detection statistic is $W_n^c = \sum_{i=1}^N W_n^{(i)}$, or if the maximal percentage, p, of the affected channels is *a priori* known, then $W_n^{c,p} = \sum_{i=1}^{pN} W_n^{(i)}$, where $W_n^{(i)}$, $i = 1, \ldots, N$ are ordered versions, i.e., $W_n^{(1)} \leq W_n^{(2)} \leq \cdots \leq W_n^{(N)}$. Such an LR-based algorithm was considered in Mei (2010). A similar approach can be used to form SR-type multichannel detection procedures (Siegmund, 2013).

Monte Carlo simulations and experiments with real data show that the multichannel score-based CUSUM and SR procedures defined above are very efficient at detecting anomalies of arbitrary nature and structure.

Yet another approach is to exploit a nonparametric algorithm with binary quantization and optimization of the quantization threshold. In this case, it is possible to implement optimal binary quantized CUSUM and SR algorithms that are based on true likelihood ratios for Bernoulli sequences at the output of quantizers. Specifically, the observations X_n, $n \geq 1$ are quantized as follows: $V_n = 1$ if $X_n > t$ and 0 otherwise, where t is a quantization threshold. After quantization, the LLR-based CUSUM or SR algorithms are applied to the binary (Bernoulli) sequence $\{V_n\}_{n\geq 1}$. Let β_0 and β_1 denote the probabilities that $V_n = 1$ under the normal and the attack conditions, respectively. It is easily seen that the LLR $Z_n^b = \log[\mathsf{P}(V_n|\mathsf{H}_0)/\mathsf{P}(V_n|\mathsf{H}_\infty)]$ for the binary data between the post-change and pre-change hypotheses is given by $Z_n^b = a_1 V_n + a_0$, where

$$a_1 = \log \frac{\beta_1 (1-\beta_0)}{\beta_0 (1-\beta_1)}, \quad a_0 = \log \frac{1-\beta_1}{1-\beta_0}.$$

The binary CUSUM and SR procedures are

$$\begin{aligned} T_{\mathrm{CS}}^b(h_b) &= \min\{n \geq 1 : W_n^b \geq h_b\}, \quad W_n^b = (0, W_{n-1}^b + Z_n^b)^+, \\ T_{\mathrm{SR}}^b(h_b) &= \min\{n \geq 1 : R_n^b \geq A_b\}, \quad R_n^b = (1 + R_{n-1}^b) e^{Z_n^b}, \end{aligned} \tag{2.21}$$

where $W_0^b = R_0^b = 0$. To optimize the performance, one should choose the threshold t to maximize the KL number $I_b(t) = \beta_1(t)a_1(t) + a_0(t)$: $t_{\mathrm{opt}} = \arg\max_{t>0} I_b(t)$. See Tartakovsky *et al.* (2006a, Section 4) for details.

Finally, it is worth mentioning that the development of an adaptive structure that allows for an efficient online estimation of both pre-change (normal) and post-change (abnormal) parameters is an important task. Preliminary experiments show that in certain conditions these adaptive algorithms are more efficient then the previous ones.

2.3.2. *Experimental study*

2.3.2.1. *Detection of DDoS attacks*

TCP SYN DoS Attack (CAIDA). To validate usefulness of a multichannel AbIDS as opposed to the single-channel system, we used the eight-hour real backbone data captured by the Cooperative Association for Internet Data Analysis (CAIDA). A bidirectional link from San Jose, CA to Seattle, WA belonging to the US backbone Internet service provider (ISP) was monitored. This data set contains a TCP SYN flooding attack.

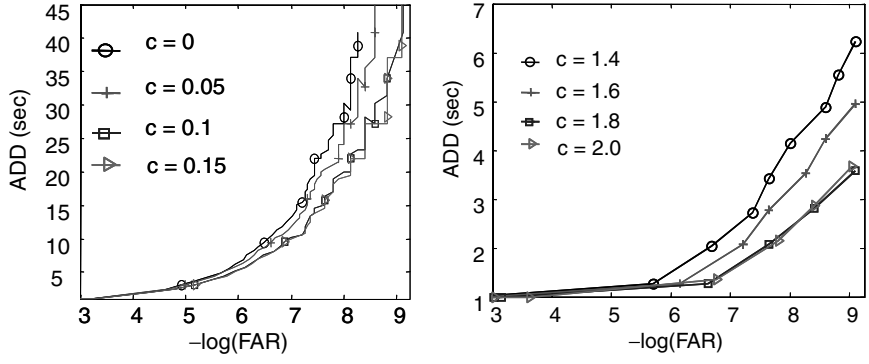
The attack's aim is to congest the victim's link with a series of SYN requests.

We compare the single-channel CUSUM (SC-CUSUM) to the multichannel CUSUM (MC-CUSUM) algorithm in terms of the maximal average detection delay (ADD) given by (2.3) and FAR expressed via the average run length to false alarm $\mathsf{ARL2FA}(T) = \mathsf{E}_\infty T$.

In the case of SYN attack detection, one can observe the number of SYNs that would cause denial of service. Since we are interested in the SYN arrival rate at a destination, we divide the channels based on their destination IP addresses. There are many ways to divide the IP address space. We take the following approach to set up the multichannel detection problem. One specific group of IP addresses that belong to the same 8-bit prefix (the first 8 bits of the IP address are the same) is considered. This group of IP addresses (/8) is further subdivided into 256 channels, each containing all the IP addresses that have the same first 16 bits (/16), so we have $N = 256$ channels. In each channel, we monitor the number of SYN packets sent per second for the entire eight-hour duration. In the single-channel case, the time series is formed by monitoring the total number of SYNs per second for all the IP addresses that have the same 8-bit prefix; the pre-change mean value $\mu_\infty = 3$ SYNs/sec and the post-change (attack) mean $\mu = 19$ SYNs/sec with the same (approximately) variance. Thus, a linear score has been used ($C_2 = 0$ in (2.15)). In the multichannel case, the attack occurs in the channel $i = 113$ with mean values $\mu_\infty^{113} = 0.0063$ SYNs/sec and $\mu^{113} = 15.3$ SYNs/sec. It is therefore obvious that localizing the attack with the multichannel MAX algorithm (2.18) based on the thresholding of the statistic $W_{\max}(n) = \max_{1\le i\le 256} W_n^{(i)}$ enhances the detection capability.

Figures 2.5(a) and 2.5(b) illustrate the relation between the ADD and $-\log(\mathrm{FAR}) = \log \mathsf{ARL2FA}$ for the SC-CUSUM and MAX MC-CUSUM detection algorithms, respectively. The optimal value of the design parameter $C_3 = c$ is $c_{\mathrm{opt}} = 0.1$ for the single-channel case and $c_{\mathrm{opt}} = 1.8$ for the multichannel case. In the extreme right of the plot, we achieve the ARL2FA of 8103 sec, i.e., 2.25 hours ($-\log(\mathrm{FAR}) = 9$). For this FAR, the ADD for the MAX MC-CUSUM is 3.5 sec, while the ADD for the SC-CUSUM is 45 sec, about 13 times higher. Since the ADD dramatically increases as the FAR decreases, for the lower FAR the SC-CUSUM algorithm may be unable to detect short attacks.

We therefore conclude that in certain scenarios the use of multichannel intrusion detection systems may be very important.



(a) The SC-CUSUM algorithm (b) The MAX MC-CUSUM algorithm

Fig. 2.5. ADD versus $-\log(\text{FAR})$ for the SC-CUSUM and MAX MC-CUSUM algorithms.

TCP SYN DDoS Attack (LANDER Project). Next, we present the results of testing the single-channel (score-based) CUSUM and SR detection algorithms with respect to the STADD introduced in (2.9) when the attack occurs long after surveillance starts and is preceded by multiple false alarms. This testing was performed for a real data set containing a DDoS SYN flood attack. The data is courtesy of the Los Angeles Network Data Exchange and Repository (LANDER) project (see http://www.isi.edu/ant/lander). Specifically, the trace is flow data captured by Merit Network Inc. (see http://www.merit.edu) and the attack is on a University of Michigan IRC server. It starts at approximately 550 seconds into the trace and lasts for ten minutes. Figure 2.6 shows the number of attempted connections (the connections birth rate) as a function of time. While the attack can be seen to the naked eye, it is not completely clear when it starts. In fact, there is fluctuation (a spike) in the data before the attack.

The observations $\{X_n\}_{n\geq 1}$ represent the number of connections during 20 msec batches. The estimated values of the connections birth rate mean and standard deviation for legitimate and attack traffic are: $\mu_\infty \approx 1669, \sigma_\infty \approx 114$ and $\mu \approx 1888, \sigma \approx 218$ (connections per 20 msec). Therefore, this attack leads to a considerable increase in both the mean and standard deviation of the connections birth rate.

Statistical analysis of this data set shows that the distribution of the number of attempted connections for legitimate traffic is very close to Gaussian, but for attack traffic it is not (see Tartakovsky *et al.* (2013) for

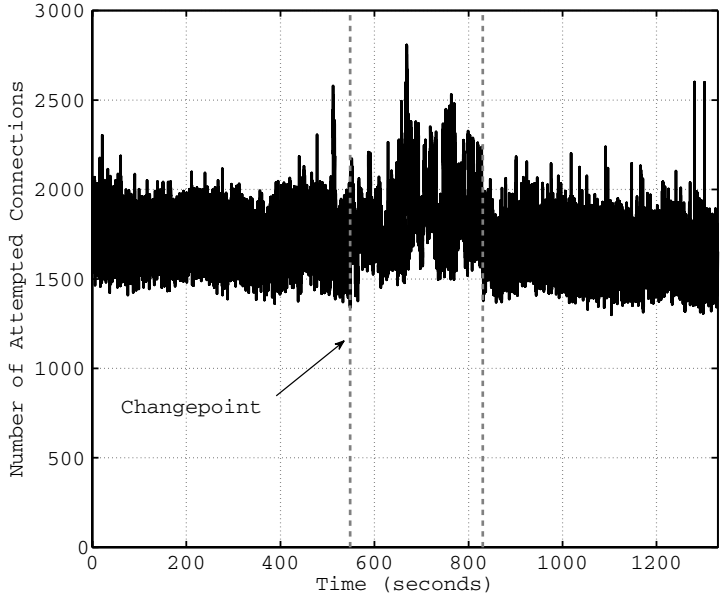


Fig. 2.6. The connections birth rate for LANDER data.

details). We implement the score-based multi-cyclic SR and CUSUM procedures with the linear-quadratic memoryless score (2.15). When choosing the design parameters C_1, C_2, C_3 we assume the Gaussian pre-attack model, i.e., the parameters C_1, C_2, and C_3 are chosen according to formulas (2.16) with $q_0 = q \approx 0.52$ and to allow for detection of dimmer attacks $\delta_0 \approx 1.5$ (versus the estimated attack value $\delta \approx 1.9$). We set the detection thresholds $A \approx 1.9 \times 10^3$ and $h \approx 6.68$ so as to ensure the same level of ARL2FA, which in the multi-cyclic setup characterizes the mean time between false alarms, at approximately 500 samples (i.e., 10 sec) for both procedures. The thresholds are estimated using Monte Carlo simulations assuming the empirical pre-change distribution learned from the data.

The results are illustrated in Figures 2.7 and 2.8. Figure 2.7 shows a relatively long run of the SR statistic with several false alarms and then the true detection of the attack with a very small detection delay (at the expense of raising many false alarms prior to the correct detection). Recall that the idea of minimizing the STADD is to set the detection thresholds low enough in order to detect attacks very quickly, which unavoidably leads to multiple false alarms prior to the attack starts. These false alarms should

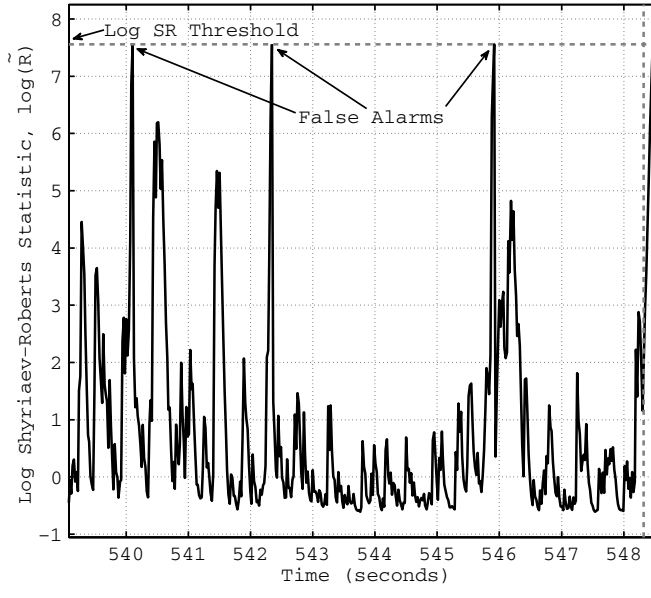
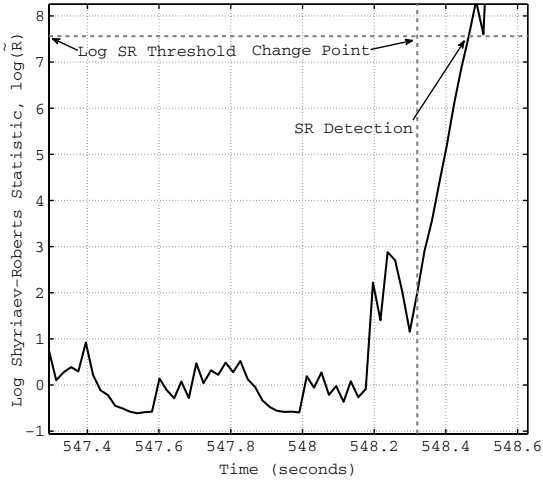


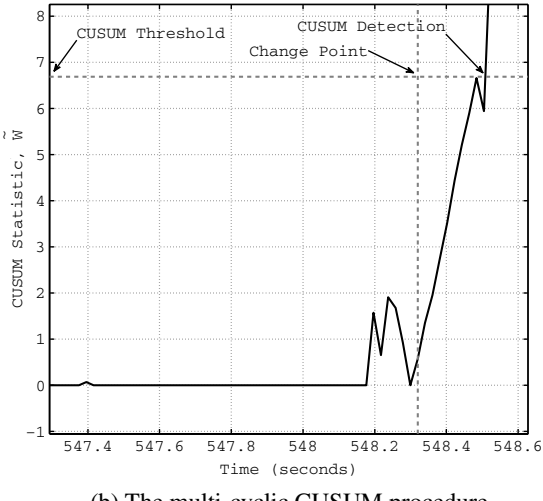
Fig. 2.7. Long run of the SR procedure (logarithm of the SR statistic versus time) for SYN flood attack.

be filtered by a specially designed algorithm, as has been suggested by Pollak and Tartakovsky (2009) and will be further discussed in Section 2.4. Figure 2.8(a) shows the behavior of $\log R_n^{\mathrm{sc}}$ shortly prior to the attack and right after the attack starts until detection. Figure 2.8(b) shows the CUSUM score-based statistic W_n^{sc}. Both procedures successfully detect the attack with very small delays and raise about seven false alarms per 1000 samples. The detection delay is approximately 0.14 seconds (seven samples) for the repeated SR procedure, and about 0.21 seconds (ten samples) for the CUSUM procedure. As expected, the SR procedure is slightly better.

UDP DoS Flooding Attack (CAIDA). Finally, we validate the feasibility of the binary CUSUM detection algorithm (2.21) on backbone data with the one-hour packet traces captured on SONET OC-48 links by CAIDA monitors. This data set, shown in Figure 2.9, contains the UDP flooding attack. Figure 2.9 shows the time series of the total number of UDP packets in a sample period of 0.015 msec. The attack is not visible to the naked eye, but an offline examination revealed that it consists of a Trojan horse called *trojan.dasda* sent from one source on port 10100 to one destination on port 44097. *Trojan.dasda* is a Trojan horse that can download and execute



(a) The multi-cyclic SR procedure



(b) The multi-cyclic CUSUM procedure

Fig. 2.8. Detection of the SYN flood attack by the multi-cyclic SR and CUSUM procedures.

remote files and open a back-door on an infected computer. Careful estimation shows that there is a change from the pre-change mean $\mu_\infty = 87$ packets per sample period to the post-change mean $\mu = 94$ packets per sample period. Thus, the parameter differentiating the normal traffic from an abnormal one is changed from 87 to 94 packets per sample period.

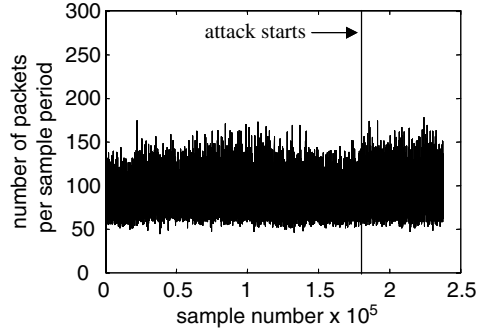
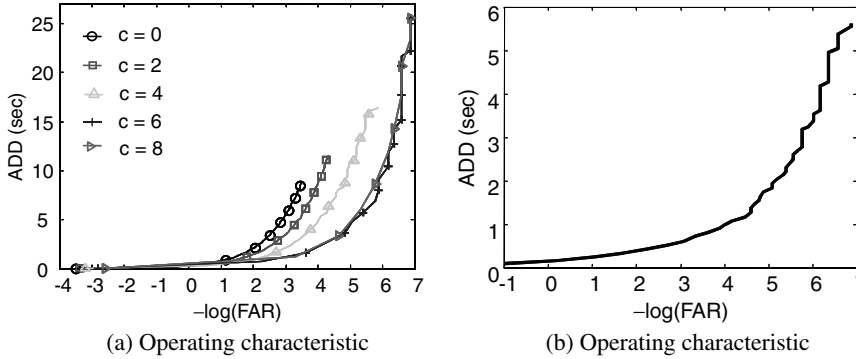


Fig. 2.9. Time series of the total number of UDP packets in a sample period 0.015 msec. Observe that the attack is not visible to the naked eye.



(a) Operating characteristic of score CUSUM.

(b) Operating characteristic of binary CUSUM.

Fig. 2.10. SADD (sec) versus −log FAR for the score-based and binary CUSUM algorithms.

Operating characteristics (SADD *versus* $-\log \mathsf{FAR} = \log \mathsf{ARL2FA}$) of the score-based and binary CUSUM algorithms are shown in Figure 2.10. Specifically, Figure 2.10(a) illustrates the operating characteristic of the linear score-based CUSUM procedure (i.e., $C_1 = 1$ and $C_2 = 0$ in (2.15)) for different values of $C_3 = c$, the tuning parameter. Note that the range of $-\log \mathsf{FAR} = \log \mathsf{ARL2FA}$ from -4 to 7 is equivalent to the frequency of false alarms from every 0.018 sec to every 1096 sec ($\approx$ 18 min). Note that in the left-most region of Figure 2.10(a) the ADD is very small but this results in a very large FAR. On the other hand, the right-most region in Figure 2.10(a) has the lowest FAR and, hence, a bigger ADD. For example, the ADD is 0.015 sec for $\mathsf{ARL2FA} = 0.018$ sec (the left-most region) and

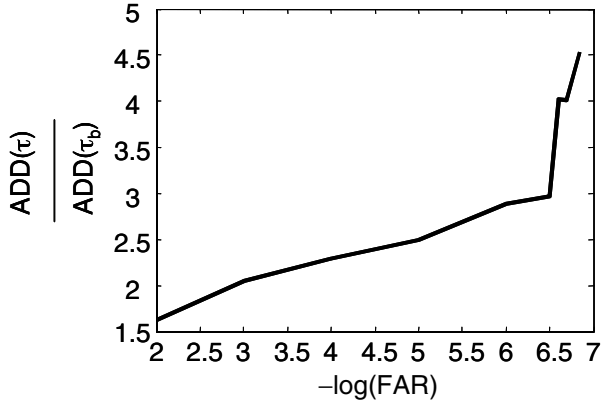


Fig. 2.11. Relative efficiency *versus* −log FAR.

the ADD is 26 sec for $\mathsf{ARL2FA} = 945$ sec (the right-most region). Varying the value of c allows us to optimize the algorithm. The optimal value of c is $c_{\rm opt} = 6$. For this c, we get the best performance in the sense that, for the same FAR, we obtain the smallest ADD as compared to other values of c.

Figure 2.10(b) shows the operating characteristic of the binary CUSUM algorithm defined in (2.21). The optimal quantization threshold is $t_{\rm opt} = 105$ and the corresponding maximum KL number $I_b(t_{\rm opt}) = 0.163$.

Figure 2.11 illustrates the efficiency of the binary CUSUM detection procedure with respect to the optimized score-based CUSUM procedure (with $c_{\rm opt} = 6$) in terms of the relative efficiency, which is defined as $\mathsf{SADD}(T^{\rm sc}_{\rm CS})/\mathsf{SADD}(T^{b}_{\rm CS})$. In this case, the binary CUSUM algorithm performs the same as the optimized score-based CUSUM algorithm for small $-\log \mathsf{FAR}$. However, the binary CUSUM shows performance improvement by a factor of 1.5 to 4.5 as compared to the optimized score-based CUSUM algorithm for larger values of $-\log \mathsf{FAR}$. This can be explained by the fact that in addition to the mean, variance also changes, in which case the use of linear-quadratic score (2.15) is more appropriate, requiring further optimization of the parameter C_2.

This allows us to conclude that the binary CUSUM algorithm is robust and efficient. Moreover, when the quantization threshold is optimized, it is optimal in the class of binary change detection procedures. As a result, it may outperform the score-based CUSUM algorithm, especially if attacks lead to changes not only in mean values but also in the entire distribution. The binary detection algorithms (CUSUM and SR) also have a simple lower

bound on the ARL2FA, $\mathsf{E}_\infty T^b(A) \geq A$, which is convenient in the design of intrusion detection systems.

2.3.2.2. *Rapid detection of spam at the network level*

Most organizations run spam filters at their local networks, such as Bayesian filters or a block list. These filters examine the content of each message as well as the IP address. If the message matches known spam signatures, it is marked as spam. These techniques work quite well but have high operational costs. Block lists rely on information gathered ahead of time and thus perhaps stale. Bayesian approaches, while quite good, are not infallible and require examining all message content.

Another approach to fighting spam is to monitor traffic at the network level looking for spam behavior. Detecting spammers at the network level has several advantages such as no privacy issues related to message content examination, near real-time detection based on network behavior, and minimizing collateral damage because dynamic addresses released by spammers can be removed from block lists quickly.

What features are useful for detecting spammers? Prior work has shown that the autonomous system the IP address belongs to, message size, blocked connections, and message length are important. All these features can be determined from network traffic and used in conjunction with changepoint detection to detect when traffic patterns from a particular host match known spammer patterns. Examples include:

(1) ***Message size.*** Most spam campaigns attempt to deliver the same (or similar) message to many recipients. With the exception of the receiver's IP address the size of the message does not vary significantly. Thus, one may use changepoint detection methods to detect hosts that send email blocks of a similar size.
(2) ***Dropped connections.*** Block lists that refuse connections from suspected spammers will be detected in network traces. Keeping track of such events can help detect spammers, and changepoint detection techniques can detect a change in dropped connections from a particular IP address.
(3) ***Connection patterns.*** Spammers typically send very few emails to a particular domain to avoid being detected. Network monitoring, however, which monitors many domains at once, can detect this pattern. Changepoint detection can detect a spammer touching many different domains.

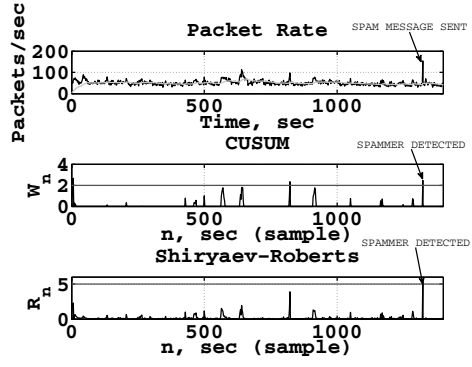


Fig. 2.12. Spam detection. Top – raw data; middle – CUSUM statistic; bottom – SR statistic.

We now illustrate rapid spam detection for a particular real-world data set to prove the feasibility of change detection methods. The data set was obtained from a regional ISP. The trace contains email flows to a mail server from a number of hosts. The records are sorted by the source IP address. Our objective is to isolate suspicious hosts and extract the typical behavioral pattern. Examining how the email size changes with time shows that it is very stable, with some occasional bursts. The individual producing such bursts is very likely to be a spammer. Figure 2.12 shows the detection of a real-world spammer using the AbIDS. The email (SMTP) traffic generated by a certain host is under surveillance. Ordinarily, SMTP traffic produced by a user sending legitimate messages is characterized by a relatively steady intensity, i.e., the number of messages sent per unit time remains more or less constant, with no major bursts or drops. However, the behavior changes abruptly once a spam attack begins: the number of sent messages explodes, possibly for a very short period of time. The top-most plot in Figure 2.12 illustrates this typical behavior. The spike in the traffic intensity that appears in the far right of the plot can easily be detected by changepoint detection methods. The behavior of the linear score-based CUSUM and SR procedures (i.e., $C_2 = 0$ in (2.15)) is plotted in the middle and bottom pictures, respectively. Both statistics behave similarly – an alarm is raised as soon as the traffic intensity blunder caused by the spam attack is encountered. The spammer is detected immediately after he/she starts activity. The difference is mainly prior to the attack – there are two false detections produced by CUSUM, while none by SR.

2.4. Hybrid Anomaly–Signature IDS

2.4.1. *IDS structure*

Since in real life legitimate traffic dominates, comparing various AbIDSs using the multi-cyclic approach and the stationary average detection delay (2.9) is the most appropriate method for cyber-security applications. However, even an optimal changepoint detection method is subject to large detection delays if the FAR is maintained at a low level. Hence, as we have already mentioned, employing one such scheme alone will lead to multiple false detections, or if detection thresholds are increased to lower the FAR, the delays will be too large, and attacks may proceed undetected.

Could one combine changepoint detection techniques with other methods that offer very low FAR but are too time-consuming to use at line speeds? Do such synergistic anomaly detection systems exist, and if so, how can they be fused?

In this section, we answer these questions by devising a novel approach to intrusion detection based on a two-stage hybrid anomaly–signature IDS (HASIDS) with profiling, false alarm filtering, and true attack confirmation capabilities. Specifically, consider complementing a changepoint detection-based AbIDS with a flow-based signature IDS that examines the traffic's spectral profile and reacts to changes in spectral characteristics of the data. The main idea is to integrate anomaly- and spectral-signature-based detection methods so that the resulting HASIDS overcomes the shortcomings of current IDSs. We propose using "flow-based" signatures in conjunction with anomaly-based detection algorithms. In particular, Fourier and wavelet spectral signatures and related *spectral analysis techniques* can be exploited, as shown in Hussain *et al.* (2003, 2006) and He *et al.* (2009). This approach is drastically different from traditional signature-based systems because it depends not on packet content but on communication patterns alone.

At the first stage, we use either CUSUM or SR multi-cyclic (repeated) changepoint detection algorithm to detect traffic anomalies. Recall that in network security applications it is of utmost importance to detect attacks that may occur in a distant future very rapidly (using a repeated application of the same anomaly-based detection algorithm), in which case the true detection of a real change is preceded by a long interval with frequent false alarms that should be filtered (rejected) by a separate algorithm. This latter algorithm is based on spectral signatures, so at the second stage we exploit a spectral-based IDS that filters false detections and confirms true attacks.

In other words, the methodology is based on using the changepoint detection method for preliminary detection of attacks with low threshold values and a discrete Fourier (or wavelet) transform to reveal periodic patterns in network traffic to confirm the presence of attacks and reject false detections produced by the anomaly IDS. When detection thresholds are low, the AbIDS produces an intense flow of false alarms. However, these false alarms can be tolerated at the level of minutes or even seconds, since they do not lead to real false alarms in the whole system. An alarm in the AbIDS triggers a spectral analyzer. This alarm will either be rejected or confirmed, in which case a final alarm will be raised. Schematically, the system is shown in Figure 2.13.

To summarize, the HASIDS is based on the following principles:

- **Anomaly IDS – Quick Detection with High FAR:** In order to detect attacks quickly, the detection threshold in the changepoint detection module is lowered leading to frequent false alarms that are filtered by a separate algorithm.
- **Signature IDS – False Alarm Filtering:** A spectral-based approach, e.g., Fourier or wavelet spectral analysis module, is used to reject false detections.
- **Changepoint Detection Module:** For quick detection with relatively high FAR and triggering spectral analysis algorithms.
- **Spectral Analysis Module:** For false alarm filtering/rejection and true attack confirmation.

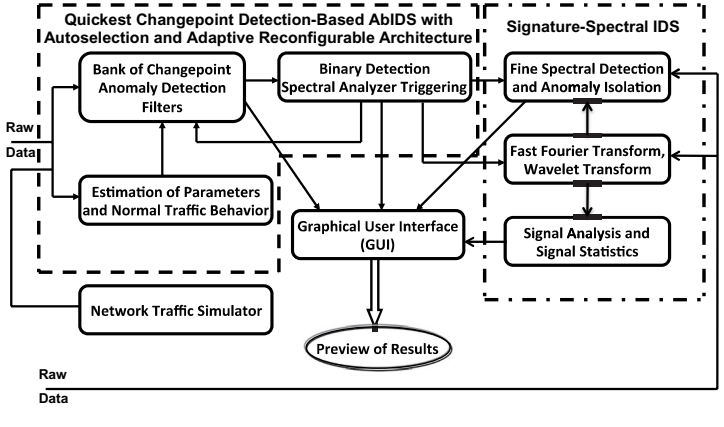


Fig. 2.13. Block diagram of the hybrid anomaly–signature intrusion detection system.

This approach allows us not only to detect attacks with small delays and a low FAR but also to isolate/localize anomalies precisely, e.g., low-rate pulsing attacks. See Figure 2.15 in Subsection 2.4.2 for further details. The results of experiments presented below show that such combining of the changepoint anomaly- and spectral-signature-based detectors significantly improves the system's overall performance, reducing the FAR to the minimum and simultaneously guaranteeing very small detection delays.

2.4.2. *Experimental study*

We now present sample testing results that illustrate efficiency of the HASIDS for LANDER data sets.

ICMP Attack (LANDER). The first data set is a tcpdump trace file containing a fragment of real-world malicious network activity, identified as an ICMP attack. The trace was captured on one of the Los Nettos private networks (a regional ISP in Los Angeles).

Figure 2.14 demonstrates the attack detection. It shows raw data (top), the behavior of the multi-cyclic CUSUM statistic (middle), and the power spectral density (PSD) of the data (bottom). The hybrid IDS filtered all false alarms (shown by green circles) and detected the attack very rapidly after its occurrence. Note that the spectral analyzer is triggered only when a threshold exceedance occurs. None of the false alarms passed the hybrid system since the peak in spectrum appeared only after the attack began. This allowed us to set a very low threshold in the anomaly IDS, resulting in a very small delay to detection of the attack.

UDP Flooding Attack (LANDER). The next experiment demonstrates the supremacy of the hybrid anomaly–signature approach to intrusion detection over the anomaly-based approach by applying HASIDS and AbIDS to detect and isolate a real-world "double-strike" UDP DDoS attack. The attack is composed of two consecutive "pulses" in the traffic intensity, shown in Figure 2.15(a). Each pulse is a sequence of seemingly insignificant packets (roughly 15 bytes in size) sent to the victim's UDP port 22 at a rate of about 180 Kbps. This is approximately thrice the intensity of the background traffic (about 53 Kbps). Although each individual packet may appear to be harmless due to its small size, each pulse's combined power is sufficient to knock the machine down almost instantaneously. One would think that because the attack is so strong, any IDS will be able to detect it quickly. However, the challenge is that each pulse is rather short, the

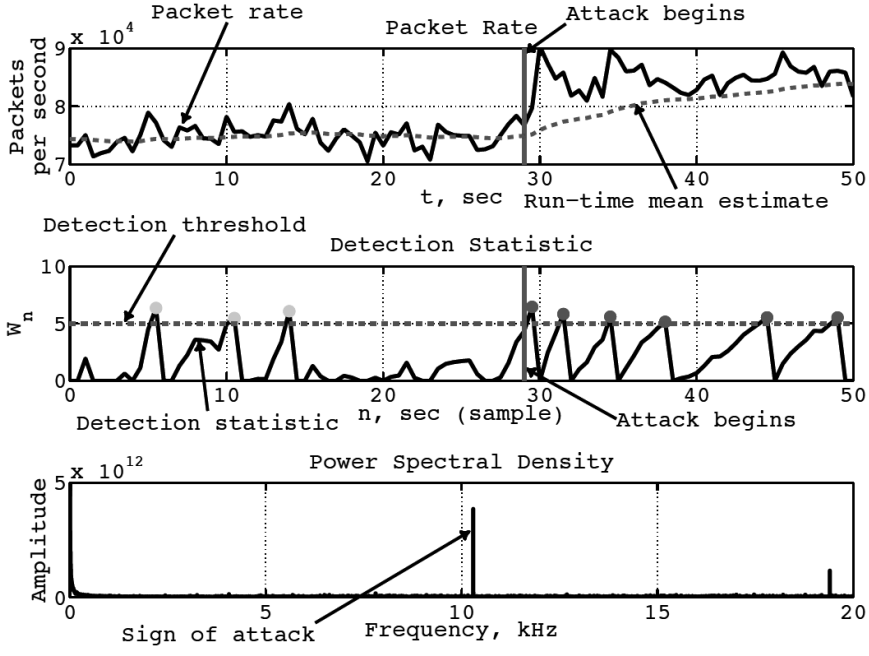
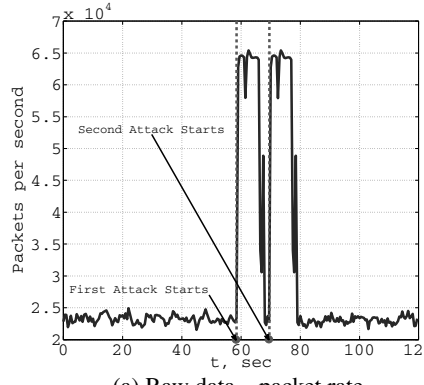


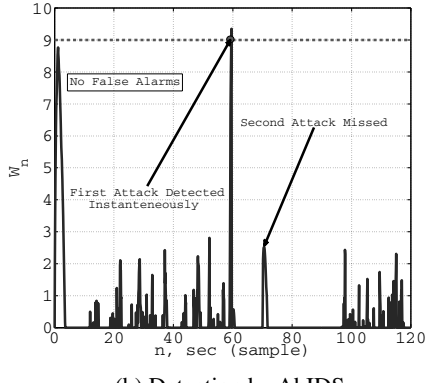
Fig. 2.14. Detection of the ICMP DDoS attack with HASIDS.

gap between the two pulses is very short, and the source of the attack packets for each pulse is different. Therefore, if the detection speed is comparable to the short duration of the pulses, the attack will get through undetected. Furthermore, if the renewal time (i.e., the interval between the most recent detection and the time the IDS is ready for a new detection) is longer than the distance between the pulses, then even though the first pulse may be detected, the second one is likely to be missed. Hence, this scenario is challenging and illustrates the efficiency of the proposed HASIDS not only for detecting attacks quickly but also for isolating closely located anomalies.

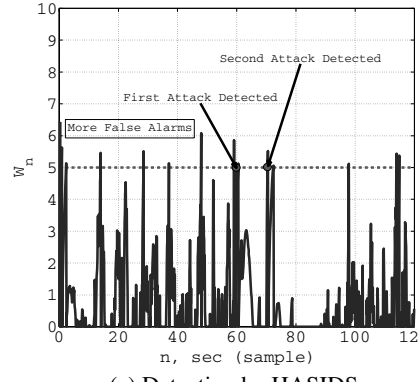
Figure 2.15 illustrates the detection by the change detection-based AbIDS (Figure 2.15(b)) and by the HASIDS (Figure 2.15(c)). Figure 2.15(b) shows that the anomaly IDS detects the first pulse but misses the second one because the threshold in the anomaly detector is high to guarantee the given low FAR. With the selected threshold, this trace shows no false alarms. If the threshold is lowered, as it is in Figure 2.15(c),



(a) Raw data – packet rate.



(b) Detection by AbIDS.



(c) Detection by HASIDS.

Fig. 2.15. Detection of a short UDP DoS attack with AbIDS and HASIDS. The second "pulse" is missed by the AbIDS but not by the HASIDS.

both segments of the attack are perfectly detected and localized. However, this brings many false alarms at the output of the changepoint detection based AbIDS. What can be done? The hybrid IDS offers an answer. The FFT (fast Fourier transform) spectral module is triggered by the AbIDS when detections (false or true) occur. In this particular experiment, all false alarms were filtered by the FFT spectral analyzer. This allowed us to lower the detection threshold in the AbIDS and, as a result, detect both pulses with a very small detection delay and with no increase of FAR, since in the hybrid system false alarms were filtered by the spectral module (see Figure 2.15(c)).

2.5. Conclusion

The experimental study shows that the proposed AbIDS, which exploits score-based CUSUM and SR changepoint detection methods, is robust and efficient for detecting a multitude of computer intrusions, e.g., UDP, ICMP, and TCP SYN DDoS attacks as well as spammers. More importantly, devising the hybrid anomaly–signature IDS that fuses quickest change detection techniques with spectral signal processing methods solves both aspects of the intrusion detection problem. It achieves unprecedented speeds of detection and simultaneously has a very low FAR in ultra-high-speed networks. In addition to achieving high performance in terms of the tradeoff between delays to detection, correct detections and false alarms, the hybrid system allows one to estimate anomaly length and distinguish between anomalies, i.e., efficient isolation-localization of anomalies.

Acknowledgments

This work was supported in part by the U.S. National Science Foundation under grants CCF-0830419 and DMS-1221888 and the U.S. Army Research Office under MURI grant W911NF-06-1-0044 at the University of Southern California, Department of Mathematics as well as the U.S. Department of Energy Office of Science SBIR contract at ARGO SCIENCE CORP.

The author would like to thank Christos Papadopoulos (Colorado State University, Department of Computer Science) and John Heidemann (University of Southern California, Information Sciences Institute) for help with obtaining real data and for useful discussions. The author is also grateful to Aleksey Polunchenko, Kushboo Shah, and Greg Sokolov for the help with simulations and processing real data.

References

Basseville, M. and Nikiforov, I. V. (1993). *Detection of Abrupt Changes – Theory and Application*, Information and System Sciences Series (Prentice Hall, Inc, Englewood Cliffs, NJ).

Debar, H., Dacier, M. and Wespi, A. (1999). Toward a taxonomy of intrusion detection systems, *Comput. Net.* **31**, 8, pp. 805–822.

Ellis, J. and Speed, T. (2001). *The Internet Security Guidebook: From Planning to Deployment* (Academic Press, Amsterdam).

Fuh, C.-D. (2003). SPRT and CUSUM in Hidden Markov Models, *Ann. Stat.* **31**, 3, pp. 942–977.

Fuh, C.-D. (2004). Asymptotic operating characteristics of an optimal change point detection in Hidden Markov Models, *Ann. Stat.* **32**, 5, pp. 2305–2339.

He, X., Papadopoulos, C., Heidemann, J., Mitra, U. and Riaz, U. (2009). Remote detection of bottleneck links using spectral and statistical methods, *Comput. Net.* **53**, 3, pp. 279–298.

Hussain, A., Heidemann, J. and Papadopoulos, C. (2003). A framework for classifying denial of service attacks, in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures and Protocols for Computer Communications* (Karlsruhe, Germany), pp. 99–110. Available at: http://doi.acm.org/ 10.1145/863955.863968.

Hussain, A., Heidemann, J. and Papadopoulos, C. (2006). Identification of repeated denial of service attacks, in *Proceedings of the 25th IEEE International Conference on Computer Communications* (IEEE, Barcelona, Spain), pp. 1–15, doi:10.1109/INFOCOM.2006.126.

Kent, S. (2000). On the trail of intrusions into information systems, *IEEE Spectrum* **37**, 12, pp. 52–56.

Lai, T. L. (1998). Information bounds and quick detection of parameter changes in stochastic systems, *IEEE T. Inform. Theory* **44**, 7, pp. 2917–2929.

Lorden, G. (1971). Procedures for reacting to a change in distribution, *Ann. Math. Stat.* **42**, 6, pp. 1897–1908.

Loukas, G. and Öke, G. (2010). Protection against denial of service attacks: A survey, *Comput. J.* **53**, 7, pp. 1020–1037.

Mei, Y. (2010). Efficient scalable schemes for monitoring a large number of data streams, *Biometrika* **97**, 2, pp. 419–433.

Mirkovic, J., Dietrich, S., Dittrich, D. and Reiher, P. (2004). *Internet Denial of Service Attack and Defense Mechanisms* (Prentice Hall PTR, Indianapolis, IN).

Moustakides, G. V. (1986). Optimal stopping times for detecting changes in distributions, *Ann. Stat.* **14**, 4, pp. 1379–1387.

Moustakides, G. V., Polunchenko, A. S. and Tartakovsky, A. G. (2011). A numerical approach to performance analysis of quickest change-point detection procedures, *Stat. Sinica* **21**, 2, pp. 571–596.

Page, E. S. (1954). Continuous inspection schemes, *Biometrika* **41**, 1–2, pp. 100–114.

Paxson, V. (1999). Bro: A system for detecting network intruders in real-time, *Comput. Net.* **31**, 23–24, pp. 2435–2463.

Pollak, M. (1985). Optimal detection of a change in distribution, *Ann. Stat.* **13**, 1, pp. 206–227.

Pollak, M. (1987). Average run lengths of an optimal method of detecting a change in distribution, *Ann. Stat.* **15**, 2, pp. 749–779.

Pollak, M. and Tartakovsky, A. G. (2009). Optimality properties of the Shiryaev–Roberts procedure, *Stat. Sinica* **19**, 4, pp. 1729–1739.

Polunchenko, A. S. and Tartakovsky, A. G. (2010). On optimality of the Shiryaev–Roberts procedure for detecting a change in distribution, *Ann. Stat.* **38**, 6, pp. 3445–3457.

Polunchenko, A. S. and Tartakovsky, A. G. (2012). State-of-the-art in sequential change-point detection, *Methodol. Comput. Appl. Prob.* **14**, 3, pp. 649–684, doi:10.1007/s11009-011-9256-5.

Polunchenko, A. S., Tartakovsky, A. G. and Mukhopadhyay, N. (2012). Nearly optimal change-point detection with an application to cybersecurity, *Sequential Anal.* **31**, 4, pp. 409–435, doi:10.1080/07474946.2012.694351.

Roesch, M. (1999). Snort — lightweight intrusion detection for networks, in *Proceedings of the 13th Systems Administration Conference (LISA), Seattle, Washington, USA* (USENIX), pp. 229–238.

Shiryaev, A. N. (1963). On optimum methods in quickest detection problems, *Theor. Probab. Appl.* **8**, 1, pp. 22–46.

Siegmund, D. (2013). Change-points: From sequential detection to biology and back, *Sequential Anal.* **32**, 1, pp. 2–14.

Tartakovsky, A. G. (2005). Asymptotic performance of a multichart CUSUM test under false alarm probability constraint, in *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC'05), Seville, Spain*, IEEE (Omnipress CD-ROM), pp. 320–325.

Tartakovsky, A. G., Nikiforov, I. V. and Basseville, M. (2014). *Sequential Analysis: Hypothesis Testing and Change-Point Detection*, Statistics (Chapman & Hall/CRC, Boca Raton, FL).

Tartakovsky, A. G., Pollak, M. and Polunchenko, A. S. (2012). Third-order asymptotic optimality of the generalized Shiryaev–Roberts changepoint detection procedures, *Theor. Probab. Appl.* **56**, 3, pp. 457–484.

Tartakovsky, A. G. and Polunchenko, A. S. (2007). Decentralized quickest change detection in distributed sensor systems with applications to information assurance and counter terrorism, in *Proceedings of the 13th Annual Army Conference on Applied Statistics* (Rice University, Houston, TX).

Tartakovsky, A. G. and Polunchenko, A. S. (2008). Quickest changepoint detection in distributed multisensor systems under unknown parameters, in *Proceedings of the 11th IEEE International Conference on Information Fusion* (Cologne, Germany).

Tartakovsky, A. G., Polunchenko, A. S. and Sokolov, G. (2013). Efficient computer network anomaly detection by changepoint detection methods, *IEEE J. Sel. Top. Signal Process.* **7**, 1, pp. 4–11.

Tartakovsky, A. G., Rozovskii, B. L., Blaźek, R. B. and Kim, H. (2006a). Detection of intrusions in information systems by sequential change-point methods, *Stat. Method.* **3**, 3, pp. 252–293.

Tartakovsky, A. G., Rozovskii, B. L., Blaźek, R. B. and Kim, H. (2006b). A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods, *IEEE Tran. Signal Proc.* **54**, 9, pp. 3372–3382.

Tartakovsky, A. G. and Veeravalli, V. V. (2004). Change-point detection in multichannel and distributed systems, in N. Mukhopadhyay, S. Datta and S. Chattopadhyay (eds), *Applied Sequential Methodologies: Real-World Examples with Data Analysis, Statistics: a Series of Textbooks and Monographs*, Vol. 173 (Marcel Dekker, Inc, New York), pp. 339–370.

Tartakovsky, A. G. and Veeravalli, V. V. (2005). General asymptotic Bayesian theory of quickest change detection, *Theor. Probab. Appl.* **49**, 3, pp. 458–497.

Chapter 3

Statistical Detection of Intruders Within Computer Networks Using Scan Statistics

Joshua Neil, Curtis Storlie, Curtis Hash and Alex Brugh

Los Alamos National Laboratory
PO BOX 1663, Los Alamos, New Mexico, 87545, USA
jneil@lanl.gov

We introduce a computationally scalable method for detecting small anomalous subgraphs in large, time-dependent graphs. This work is motivated by, and validated against, the challenge of identifying intruders operating inside enterprise-sized computer networks with 500 million communication events per day. Every observed edge (time series of communications between each pair of computers on the network) is modeled using observed and hidden Markov models to establish baselines of behavior for purposes of anomaly detection. These models capture the bursty, often human-caused, behavior that dominates a large subset of the edges. Individual edge anomalies are common, but the network intrusions we seek to identify always involve coincident anomalies on multiple adjacent edges. We show empirically that adjacent edges are primarily independent and that the likelihood of a subgraph of multiple coincident edges can be evaluated using only models of individual edges. We define a new scan statistic in which subgraphs of specific sizes and shapes (out-stars and 3-paths) are tested. We show that identifying these building-block shapes is sufficient to correctly identify anomalies of various shapes with acceptable false discovery rates in both simulated and real-world examples.

3.1. Introduction

In this chapter, we consider the problem of detecting locally anomalous activity in a set of time-dependent data having an underlying graph structure. While the method proposed can be applied to a general setting in which data is extracted from a graph over time, and in which anomalies occur in connected subgraphs, we will focus exclusively on the detection of attacks within a large computer network. Specifically, we are interested in detecting those attacks that create connected subgraphs within which the communications have deviated from historic behavior in some window of time.

We start with a discussion of computer network data, and the underlying graph induced by this network.

3.1.1. *Basic graph concepts and computer network data*

A graph consists of nodes and edges (Kolaczyk, 2009). In the example of a computer network, nodes are computers and edges are a time series of directed communications between computers. In general, data can be collected over time from both nodes and edges. For this chapter, however, we will only consider data extracted from network communications, with node data a subject of future work. The data we will focus on was obtained from NetFlow records (Bensley *et al.*, 1997; Brownlee *et al.*, 1997; Phaal *et al.*, 2001) gathered from one of Los Alamos National Laboratory's (LANL's) internal networks, over 30 days in 2010. Internet protocol (IP) addresses define nodes, and counts of connections per minute between IPs define a time series on the directed edge between those nodes, resulting in a total of 558,785 edges. Each edge is directed, in the sense that communications are marked with a source and destination, and an edge with reversed source and destination nodes is considered a separate edge from the forward direction. The data is observed every minute, and in a 30-minute period the network graph has in the order of 20,000 active nodes and 90,000 active edges.

This is a fairly difficult data set to come by, since collection is a challenge for many reasons. At many universities, for example, data from personal machines is not collected for privacy reasons. In addition, sensors in the network that measure this data need to be distributed well to have access to all of the connections, and when they are, the data rate can be very high, so that significant engineering is required to collect and store these measurements. For these reasons, data of this type is not widely available, and there is little published work using it. The LANL has invested significant resources in collection over the past ten years, and since attacks can and are observed via edge data, we feel it is an extremely fruitful data stream for researchers to collect and analyze.

A plot of one of these edges is given in Figure 3.1. There is an enormous variety among the edges in a typical network. Those that are driven by human presence, such as the edge created between a desktop and an email server when a user checks his email, tend to be similar to Figure 3.1. Others, such as edges between load-balancing servers, tend to be much smoother. Several of the edges exhibit periodic behavior, at multiple timescales.

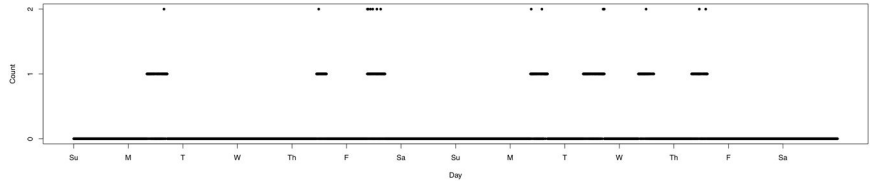


Fig. 3.1. Counts of connections per minute between two machines in LANL's internal network. The connections originate at a specific user's machine, and the destination is a server providing virtual desktop services.

In order to establish a baseline of behavior for each edge, modeling is used. Section 3.4 presents three models, one of which, the hidden Markov model, attempts to capture this human behavior, by explicitly accounting for the burstiness apparent on this edge. A more thorough modeling effort is required to accurately reflect the variety of edges seen in this data, an effort that is ongoing, but the three models presented represent a first step.

Attacks create deviations not just on single edges, but across multiple connected edges, a subject discussed in the next section. Under an independence assumption among the edges in the subgraph, models for edges lead to models for subgraphs as discussed in Section 3.3.

3.1.2. *Example traversal attack*

A common initial stage of attack on computer networks is to infect a machine on the network using malicious software. One method for initial infection is known as a phishing attack, where an email that includes a link to a malicious website is sent to a set of users on a network. When the user clicks on the link, their machine becomes infected, giving the attacker some form of access to the machine.

The attacker generally cannot dictate which machine is infected, and the initial host is usually not the ultimate target of the attack, if there even is an ultimate target. Instead, the attacker may wish to move to other machines in order to locate and exfiltrate valuable data, escalate privileges, or to establish broad presence in the network for later exploitation. Therefore, from this initial host, the attacker may proceed to other hosts, hopping from one to the next; see Figure 3.2.

In order to maximize the true positive rate, and minimize the false one, statistical testing is performed at the subgraph level, not at that of each edge. The task then is to form a subgraph that simultaneously captures as many attack edges as possible, and as few non-attack edges as possible.

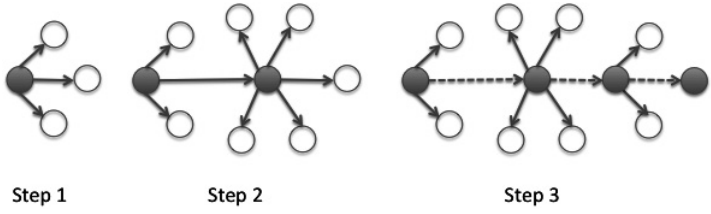


Fig. 3.2. A traversal attack. Step 1: Initial infection and local search. Step 2: First traversal has occurred, and further search is performed. Step 3: A full traversal has occurred. This shape is denoted as a *caterpillar*. The data on each edge in this graph is potentially anomalous. The filled nodes and dotted edges in Step 3 form a 3-path, which is one type of shape used to capture this behavior.

3.1.3. *Attack shapes in the graph*

Our institution is under regular attack by many entities, from simple automated tools scanning our firewalls through a spectrum of attack types including dedicated, sophisticated teams of attackers. This chapter was motivated by two canonical types of attacks seen against our site. While the details cannot be discussed for security reasons, forensic expertise has been brought to bear to describe these attacks to the authors.

Scanning Behavior: Out-stars. Attackers may wish to search locally around a single node for vulnerabilities in neighboring computers. This generates traffic emanating from a central node, to a set of destination nodes, forming a star in the graph. Out-stars, introduced by Priebe *et al.* (2005), are defined as the set of edges whose source is a given central node; see Figure 3.3. Since some computers, such as email servers, communicate with large numbers of hosts, these shapes can contain large numbers of edges, and an attack that only used a few of the edges in the star may be lost in edges not part of the local search behavior. This suggests enumerating subsets of stars, which is the subject of future work. For this discussion, all edges emanating from a node are examined.

Traversal Behavior: *k*-Paths. Another type of behavior commonly seen in real attacks is that of attacker traversal, as depicted by the filled nodes and dotted edges in Figure 3.2, Step 3. To capture this behavior, we suggest the directed *k-path*. Informally, a k-path is a sequence of k edges where the destination node of the current edge in the sequence is the source node of the next edge in the sequence, and so on. In graph terms, a k-path is a directed subgraph where both size and diameter are equal to k (Kolaczyk, 2009).

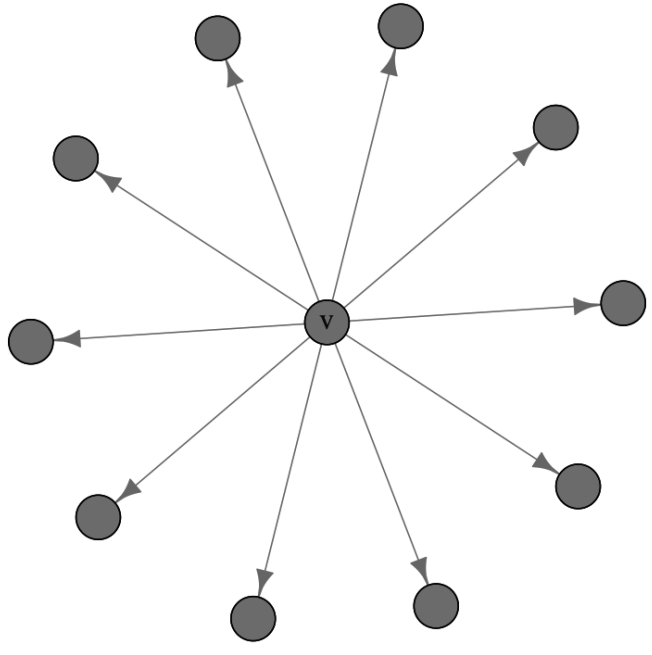


Fig. 3.3. Example out-star, centered at node v.

The k-path captures the core of many network attacks, which have a path through the network with additional edges as "fuzz" around this core path. In addition, the k-path is limited to k edges, allowing for the detection of very small anomalies. In the simulation (Section 3.5) and real-data (Section 3.6) studies, we choose to use 3-paths. 3-paths have the advantage of locality, but are also large enough to capture significant traversal. In a complete system, we forsee analyzing all 1-, 2- and 3-paths, but longer paths are less local, providing analysts with more alarmed edges to sort through.

3.1.4. *Related work*

We are interested in testing the null hypothesis that all edges in the graph are behaving as they have historically, versus the alternative that there are local shapes of altered activity among the edges. To accomplish this goal, we have developed a method based on scan statistics to examine each of these shapes in the graph over sliding windows of time. Scan statistics have been widely used to detect local clusters of events (Naus, 1982; Loader,

1991; Kulldorff, 1997; Glaz *et al.*, 2001). The idea is to slide a window over a period of time and/or space, calculating a local deviation statistic. The most extreme of these is known as the scan statistic, which is used to decide if there is any deviation from historic behavior in the local window.

Fast statistical anomaly detection on streaming data has become an important area of research given the proliferation of data over the past few decades, and the need to detect quickly the event that a process has changed significantly from past behavior. Applications can be found in many areas including engineering (Chandola *et al.*, 2009), computer science (Forrest *et al.*, 1996), and, specifically, in communications networks (Mukherjee *et al.*, 1994; Yeung and Ding, 2003; Lambert and Liu, 2006; Chandola *et al.*, 2009).

In many cases, the data can be represented as a graph (Kolaczyk, 2009). Nodes represent actors sending and receiving data, and edges represent communications between nodes. Anomalies can be detected in the changes to the structure of the graph (Noble and Cook, 2003; Collins and Reiter, 2007).

Scan statistics for communications graphs were established in Priebe *et al.* (2005), and used a star shape. Paths are compared with stars in Section 3.5.1. Similar methods that aggregate at the node, examining each node's behavior independently, include Yeung and Ding (2003) and Mukherjee *et al.* (1994). In none of this work are edges modeled. Yet, different edges may have significantly different behavior over time, and attacks between nodes must happen over edges. In addition, traversal cannot be captured by analyzing node behavior separately for each node. In these cases, modeling each edge is desirable. Additionally, all of these graph methods tend to lack fine-grained locality, which we address by using k-paths. Because of this locality, we have discovered attacks that are not specifically traversal or star-shaped.

In only one article identified, Heard *et al.* (2010), are the individual edges modeled. A Bayesian testing framework is proposed to test the anomalousness of each edge in a social network, without consideration of other local-edge anomalousness. These edges are then passed to a secondary analysis that examines the graph constructed from the edges that were detected in the initial pass. Interesting features of the anomalous edge graph can be detected in this way, but simultaneously testing multiple local sets of edges will have increased power to detect locally anomalous behavior. For example, if two anomalous edges were connected by a non-anomalous edge, this possible traversal path would likely be missed by the technique in Heard

et al. (2010), but is a valid anomaly in many settings. In addition, when data speeds are high, a fully Bayesian treatment may pose computational difficulties, unless the model is parsimonious enough for sequential Monte Carlo (Doucet *et al.*, 2001).

Hidden Markov models (HMMs) for communcations networks are discussed in Section 3.4. Salamatian and Vaton (2001) discuss using HMMs to examine packet loss in the internet. The data we work with, however, is internal network data, and the challenges of modeling edge data are entirely different than those of modeling highly aggregated flows over the internet. There is some interesting work on identifying groups in social networks using HMMs in the literature (Baumes *et al.*, 2004, 2006). While these methods are geared to social networks, do not use edge data, and use prelabeled examples, not anomaly detection, we feel this work to be inspirational, as it tackles many similar issues. Finally, Ye *et al.* (2000) present a Markov model for audit data from Unix machines to perform node-based anomaly detection. No network modeling is done, and focusing on a set of Unix machines is no longer of particular interest, since modern networks have a variety of operating systems, and a general approach to network anomaly detection cannot focus on a single operating system.

Paths have been examined in the context of vehicular traffic in Lu *et al.* (2009), using a similarity metric to compare paths, and then clustering to find outliers. This method, however, assumes we observe path values that can be clustered. On the contrary, in this chapter we propose a statistically rigorous method to infer anomalous shapes from the network without any prior knowledge about traversal by individual actors.

Many of the methods mentioned above are intended for much smaller graphs than our method proposes to address. We have a data set that is difficult to come by: a record of all of the communications between individual computers on a large corporate-sized network. These communications are recorded at fine timescales (1 second or finer), and have been archived for the past decade in some cases. The objects (paths) we monitor number in the hundreds of millions per 30-minute window, which we are able to test in under 5 seconds. With the exception of the telephone network literature (Lambert *et al.*, 2001; Lambert and Liu, 2006) (that does not monitor at a graph level, but at individual aggragation points), the sheer size of this endeavor separates it from most other work.

The formal statement of the scan statistic is given in Section 3.2. Independence between the edges within a path is covered in Section 3.3. Modeling of edge data is discussed in Section 3.4. Stars and paths are compared

on a variety of simulations in Section 3.5. Finally, we present results of scanning on actual computer network data in Section 3.6.

3.2. The Scan Statistic

In this section, we describe the methodology behind scanning for local anomalies in a graph over time. Windowing in this space is then discussed, followed by the definition of the scan statistic.

3.2.1. *Windows in the cross product space*

We are interested in examining sets of windows in the $Time \times Graph$ product space. We define these sets of windows as follows. We have a graph $G = (V, E)$ with node set V and edge set E. For each edge $e \in E$, at discrete time points $t \in \{1, \ldots, T\}$, we have a data process $X_e(t)$. We denote the set of time windows on edges e over discretized time intervals $(s, s+1, \ldots, k)$ as $\Omega = \{[e, (s, s+1, \ldots, k)] : e \in E, 0 \leq s < k \leq T\}$.

The set of all subsets of windows, $\Gamma = \{\{w_1, w_2, \ldots\} : w_j \in \Omega\}$, is usually very large, and we are normally interested in only a subset, $\Gamma_s \subset \Gamma$, that contains locality constraints in time and in graph space. We therefore restrict our attention to sets of windows $\gamma \in \Gamma_s$.

For convenience, we denote $\boldsymbol{X}(\gamma)$ as the data in the window given by γ. Next, we assume that for any time point t and edge e, we can describe $X_e(t)$ with a stochastic process (specific examples are given in Section 3.4) with parameter function given by $\theta_e(t)$. We denote the values of the parameter functions evaluated in the corresponding set of windows γ by $\boldsymbol{\theta}(\gamma)$. Finally, we denote the likelihood of the stochastic process on γ as $\mathcal{L}(\boldsymbol{\theta}(\gamma) \,|\, \boldsymbol{X}(\gamma))$.

At this point, it is worth returning to our discussion in Section 3.1.3 of the 3-path used to detect traversal. In this example, $X_e(t)$ are the directed time series of counts of connections between the pair of hosts that define each edge e. Then, Ω is the set of all (edge, time interval) pairs. We would like to combine edges to form shapes, so we take all subsets of Ω and call that Γ. For this example, we now restrict our set of shapes to sets consisting of three (edge, time interval) pairs such that the edges form a directed 3-path, and the time interval is selected to be the same on each edge. In the simulations and real network example, the time intervals are 30 minutes long, and overlap by ten minutes with the next time window, and are identical on each edge in the shape. These are then the windows γ that are used in the 3-path scan shape.

3.2.2. *A scan statistic for windows in the Time × Graph space*

We would like to examine whether the data in a window γ was likely to have been produced by some known function of the parameters $\boldsymbol{\theta}_0(\gamma)$, versus alternatives indicating that the parameters have changed. That is, given that we observe $\boldsymbol{X}(\gamma) = \boldsymbol{x}(\gamma)$, we would like to test whether $H_0 : \boldsymbol{\theta}(\gamma) = \boldsymbol{\theta}_0(\gamma)$ against alternatives that can be formed by restricting the overall parameter space, $\boldsymbol{\Theta}$, to a subset $\boldsymbol{\Theta}_A \subset \boldsymbol{\Theta}$. The generalized likelihood ratio test statistic (GLRT) is a natural statistic to use. Let

$$\lambda_\gamma = -2 \log \left(\frac{\mathcal{L}(\boldsymbol{\theta}_0(\gamma) \,|\, \boldsymbol{x}(\gamma))}{\sup_{\boldsymbol{\theta} \in \boldsymbol{\Theta}_A} \mathcal{L}(\boldsymbol{\theta}(\gamma) | \boldsymbol{x}(\gamma))} \right). \tag{3.1}$$

The size of λ_γ depends on the number of parameters being tested in the window, making it difficult to use directly. To address this issue, we normalize λ_γ by converting it into a p-value, p_γ. These p-values are discussed in detail in Section 3.4.5.

To scan for anomalies in the $Time \times Graph$ product space, we must slide over all windows γ, keeping track of the scan statistic $\Psi = \min_\gamma p_\gamma$. In practice, thresholding of the set of p-values is performed so more than just the minimum p-value can be considered. For online monitoring, we set a threshold on the p-values to control the false discovery rate (Benjamini and Hochberg, 1995). This threshold setting is described in more detail in Section 3.4.6, but we emphasize that generally, when a detection occurs, a set of windows (not just one) are detected, and so the union of these windows is the detected anomaly produced by the system.

3.3. Independence Among Edges in a Path

In order to scan for anomalous shapes, it is necessary to have models that describe the behavior of the data in the window γ under normal conditions. The number of enumerated subgraphs tends to scale exponentially with the number of nodes, however, and an assumption of independence among the edges in the shape facilitates scaling the computations required to process large graphs at line speeds, under reasonable memory requirements. While the general approach discussed in Section 3.2.1 does not require independence among the edges in the window γ, independence will be assumed among the stars and paths discussed in Section 3.1.3 and used in the simulation and real-data sections. Note that for those shapes, no edge is repeated.

Edge independence ensures the ability to scale, since models (and the storage of their parameters) for each edge are sufficient to construct models for subgraphs of edges under this assumption, whereas non-independence might require models for each *shape*, of which there may be many hundreds of millions, if not billions. Therefore, an examination of the assumption of independence among edges connected in a path is conducted.

Intuitively, independence among the edges in a 2-path (and in a 3-path) makes sense in the following way. In this chapter, the network is measured at the connection layer, layer 4 of the OSI model (Stallings, 1987), not at layer 3, which is the layer at which packets are routed. That is, end-to-end communications are measured. There is very little reason for a computer to generate connections to further computers as a result of being communicated with by some originating computer. There are exceptions, as will be apparent below, but these exceptions tend to be interesting in their own right. Detecting such flows is not a bad thing, it is a good thing, since attackers can tend to make correlated flows where there should be none. After the authors pointed out these highly correlated edges connected in a 2-path, network security personnel examined the behavior in detail.

This study utilized LANL NetFlow records over a 30-day period. For each edge, the data consists of a per-minute recording of the indicator variable that is 1 if there is activity on that edge in that minute, and 0 otherwise. This results in a sequence of 40,320 binary values for each edge. The sample correlation between pairs of edges connected in a 2-path was calculated. In this data set, there were a total of 311,411 2-paths.

Correlation was chosen as the measure used to gauge independence, since, in binary random variables, correlation equal to zero implies independence. Additionally, the independence assumption *is* being violated here, and the task then is to evaluate how severely the assumption is being violated in the data. It is clear from the results that this assumption is not far from reality, and it is likely that this model will suffice for this application.

In Figure 3.4 we plot the empirical cumulative distribution function (CDF) for the absolute values of these correlation statistics. Half of all correlations were less than 0.003 in absolute value, and only 1 in 1000 had R^2 value of larger than 6%.

Note that under the independence assumption, the path GLRT is expressed as

$$\lambda_\gamma = \sum_{e \in \gamma} \lambda_e \tag{3.2}$$

where λ_e are the GLRT scores on each edge in window γ.

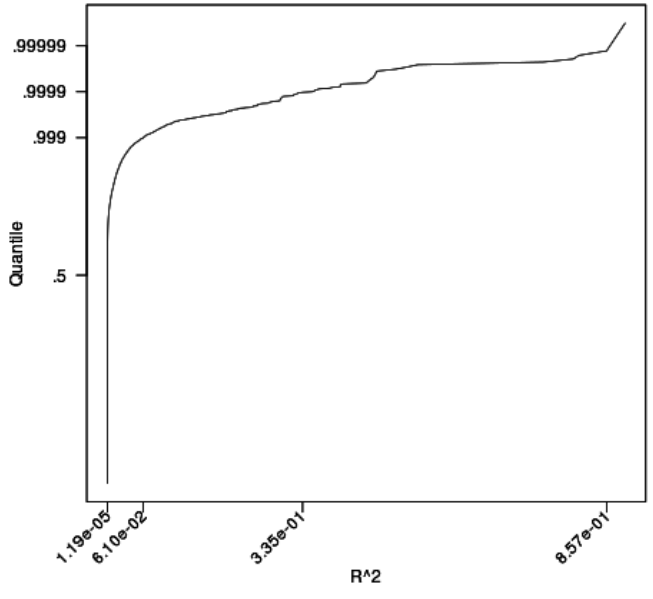


Fig. 3.4. Empirical CDF of absolute value of correlations between pairs of edges connected in a 2-path.

3.4. Modeling, Estimation, and Hypothesis Testing

As can be seen in Figure 3.1, it is common in communications between a pair of computers to observe a switching process. Intuitively, for many edges, this switching is caused by the human presence on the network. If a user is present at a machine, she may make nonzero counts on edges emanating from that machine. But in many minutes, even though the user may be present, she may not be making nonzero counts on this edge, since she may be communicating with some other machine, or not using the network at all. We only know that when she is not there, we will observe 0s on this edge. This presence/absence induces a switching process between a purely 0 count emission and one that admits positive counts. While, intuitively, there will be higher counts in the middle of the day than at night, in this chapter we use homogeneous models for the sake of simplicity. In this section, two models for capturing the switching behavior of the time series on each edge are discussed.

In addition, a model for establishing the probability that a connection is observed between two computers that have not communicated in the past is given. We denote this behavior as a *new edge*. While new edges are observed under non-attack conditions, it is a hallmark of many attacks, and

therefore an important behavior to model. Intuitively, many attackers are not aware of the normal communications patterns in a computer network, and tend to create many new edges as a result. This lack of awareness of normal behavior is a key difference between attackers and defenders, and one defenders must exploit to the fullest.

3.4.1. *Observed Markov model*

The first and simplest model is a two-state observed Markov model (OMM), which we denote B_t. If there was a nonzero count in time bin t, then $B_t = 1$, otherwise $B_t = 0$. This model has two parameters, $p_{01} = P(B_t = 1 \,|\, B_{t-1} = 0)$ and $p_{10} = P(B_t = 0 \,|\, B_{t-1} = 1)$. Its likelihood is given by

$$\mathcal{L}(p_{01}, p_{10} \,|\, b_1, \ldots, b_N) = (1 - p_{01})^{n_{00}} p_{01}^{n_{01}} p_{10}^{n_{10}} (1 - p_{10})^{n_{11}} \tag{3.3}$$

where n_{ij} is the number of times that the consecutive pair (b_i, b_j) was observed in the data. We assume that the initial state is fixed and known. Maximum likelihood estimates are given by $\hat{p}_{01} = \frac{n_{01}}{n_{00}+n_{01}}$ and $\hat{p}_{10} = \frac{n_{10}}{n_{10}+n_{11}}$.

While this model captures the burstiness, it ignores the distribution of the counts, and also does not reflect the underlying hidden process in many edges, which is that of a user being absent altogether (low state) versus the user being present (high state).

3.4.2. *Hidden Markov model*

To address the issues not covered by the OMM, we employ a two-state HMM (Rabiner, 1989) with a degenerate distribution at zero for the low state, and a negative binomial emission density in the high state. Negative binomial emission densities do not suffer from the equidispersion property of the Poisson (Pohlmeier and Ulrich, 1995), and a good justification for using them to monitor for anomalies in network counts is given in Lambert and Liu (2006). This model is similar to the hurdle models described in Heard *et al.* (2010) and elsewhere, with one important distinction: we allow the high state to emit zeros. We believe that this is important in modeling our data. Again, referring to Figure 3.1, we see that zero counts are interspersed with the nonzero data, but are still clearly a part of the "active" state. Intuitively, we think of the active state as "the user is present at the machine," and therefore likely to make communications, not as "the user is making a communication on this edge." Next, the estimation of the HMM is discussed in this setting.

Notation and Likelihood. At a set of T discrete time points we observe counts $\mathbf{x} = [x_1, \ldots, x_T]'$, with $x_t \in \{0, 1, \ldots\}$ for $t = 1, \ldots, T$. In this model, the counts are viewed as coming from one of two distributions, as governed by $\mathbf{Z} = [Z_1, \ldots, Z_T]'$, a latent two-state Markov process. Letting $p_{01} = \Pr(Z_n = 1 \,|\, Z_{n-1} = 0)$ and $p_{10} = \Pr(Z_n = 0 \,|\, Z_{n-1} = 1)$, we denote the latent transition matrix as

$$\mathbf{A} = \begin{bmatrix} 1 - p_{01} & p_{01} \\ p_{10} & 1 - p_{10} \end{bmatrix}.$$

The initial state distribution is denoted $\pi = \Pr(Z_1 = 1)$.

The marginal distribution of the count at time t, given that $Z_t = 0$ is degenerate at 0, i.e. $\Pr(X_t = x_t \,|\, Z_t = 0) = I(X_t = 0)$ where $I(\cdot)$ is the indicator function. When $Z_t = 1$, we assume that the counts are distributed according to a negative binomial distribution with mean and size parameters given by $\phi = [\mu, s]'$, i.e.

$$\Pr(X_t = x_t \,|\, Z_t = 1, \phi) = \frac{\Gamma(s + x_t)}{\Gamma(s)\Gamma(x_t + 1)} \left(\frac{s}{\mu + s}\right)^s \left(\frac{\mu}{\mu + s}\right)^{x_t}.$$

A useful fact is that the joint probability distribution over both latent and observed variables can be factored in a way that is useful for computation, since it separates the different parameter types:

$$\begin{aligned} &\Pr(\mathbf{X} = \boldsymbol{x}, \mathbf{Z} = \boldsymbol{z} \,|\, \boldsymbol{\theta}) \\ &\quad = \Pr(Z_1 = z_1 \,|\, \pi) \prod_{t=2}^{T} \Pr(Z_t = z_t \,|\, Z_{t-1} = z_{t-1}, \mathbf{A}) \\ &\qquad \times \prod_{t=1}^{T} \Pr(X_t = x_t \,|\, Z_t = z_t, \phi) \end{aligned}$$

where $\boldsymbol{\theta} = (\pi, \mathbf{A}, \phi)'$. Finally, the likelihood is

$$\Pr(\mathbf{X} = \boldsymbol{x} \,|\, \boldsymbol{\theta}) = \sum_{z_1=0}^{1} \cdots \sum_{z_t=0}^{1} \Pr(\mathbf{X} = \boldsymbol{X}, \mathbf{Z} = z \,|\, \boldsymbol{\theta}). \tag{3.4}$$

Maximum Likelihood Estimates. Equation (3.4) involves 2^T terms, making it computationally infeasible to work with directly, for even moderately large T. Hence, we look to expectation maximization (EM) as

an iterative approach for calculating the maximum likelihood estimates. EM starts with some initial selection for the model parameters, which we denote $\boldsymbol{\theta}^{old}$.

Initial Parameters. To obtain $\boldsymbol{\theta}^{old}$, we proceed by assuming that the high-state emission density, $\Pr(X_t = x_t \,|\, Z_t = 1, \phi)$ only emits positive counts. This, in effect, makes Z_t an observed random variable. Let $b_{t,0} = I(X_t = 0)$, $b_{t,1} = I(X_t > 0)$. We use initial transition probabilities defined by the maximum likelihood estimators (MLEs) of the observed Markov chain: $\tilde{p}_{01} = \frac{n_{01}}{n_{01}+n_{00}}$ $\tilde{p}_{10} = \frac{n_{10}}{n_{10}+n_{11}}$, where n_{ij} is the number of times that the consecutive pair $(b_{t-1,i}, b_{t,j})$ was observed in $\boldsymbol{x}$. An initial estimate for π is the steady-state probability given by $\tilde{\pi} = \frac{\tilde{p}_{01}}{\tilde{p}_{01}+\tilde{p}_{10}}$.

To obtain initial estimates for the high-state emission parameters ϕ, we collect the samples of $\boldsymbol{X}$ such that $X_t > 0$, and call that collection $\boldsymbol{Y}$. We then calculate $\tilde{\mu}$ and $\tilde{\sigma}^2$, the sample mean and variance of $\boldsymbol{Y}$. Finally, we reparameterize from $(\tilde{\mu}, \tilde{\sigma}^2)$ to $(\tilde{\mu}, \tilde{s})$ where $\tilde{s}$ is the initial size parameter, via $\tilde{s} = \frac{\tilde{\mu}^2}{\tilde{\sigma}^2 - \tilde{\mu}}$. This approach ignores the fact that the high-state distribution can emit zeros, but for our application these initial values were sufficient for the EM algorithm to converge in a reasonable number of iterations.

The E step. In the E step, we take these initial parameter values and find the posterior distribution of the latent variables $\Pr(\mathbf{Z} = z \,|\, \mathbf{X} = \boldsymbol{x}, \boldsymbol{\theta}^{old})$. This posterior distribution is then used to evaluate the expectation of the logarithm of the complete-data likelihood function, as a function of the parameters $\boldsymbol{\theta}$, to give the function $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$ defined by

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = \sum_{\mathbf{Z}} \Pr(\mathbf{Z} = z \,|\, \mathbf{X} = \boldsymbol{x}, \boldsymbol{\theta}^{old}) \log \Pr(\mathbf{X} = \boldsymbol{x}, \mathbf{Z} = z \,|\, \boldsymbol{\theta}). \quad (3.5)$$

It has been shown (Baum and Sell, 1968; Baker, 1975) that maximization of $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$ results in increased likelihood. To evaluate $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$, we introduce some notation. Let $\gamma(z_t)$ be the marginal posterior of z_t and $\xi(z_{t-1}, z_t)$ be the joint posterior of two successive latent variables, so

$$\gamma(z_t) = \Pr(Z_t = z_t \,|\, \mathbf{X} = \boldsymbol{x}, \boldsymbol{\theta}^{old})$$
$$\xi(z_{t-1}, z_t) = \Pr(Z_{t-1} = z_{t-1}, Z_t = z_t \,|\, \mathbf{X} = \boldsymbol{x}, \boldsymbol{\theta}^{old}).$$

Now for $k = 0, 1$, the two states of the Markov chain, we denote $z_{tk} = I(z_t = k)$, which is 1 if z_t is in state k and 0 otherwise. Let $\gamma(z_{tk})$

be the conditional probability that $z_{tk} = 1$, with a similar notation for $\xi(z_{t-1,j}, z_{tk})$. Since expectation of a binary random variable is just the probability that it takes value 1,

$$\gamma(z_{tk}) = Ez_{tk} = \sum_{\mathbf{Z}} \gamma(\mathbf{Z}) z_{tk}$$

$$\xi(z_{t-1,j}, z_{tk}) = Ez_{t-1,j}, z_{tk} = \sum_{\mathbf{Z}} \gamma(\mathbf{Z}) z_{t-1,j} z_{tk}.$$

If we substitute the joint distribution $\Pr(\mathbf{X} = \boldsymbol{x}, \mathbf{Z} = z \,|\, \boldsymbol{\theta})$, along with γ and ξ, into (3.5), we obtain

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) &= \gamma(z_{10}) \log(1-\pi) + \gamma(z_{11}) \log \pi \\ &\quad + \sum_{t=2}^{T} \sum_{j=0}^{1} \sum_{k=0}^{1} \xi(z_{t-1,j}, z_{tk}) \log A_{jk} \\ &\quad + \sum_{t=1}^{T} \gamma(z_{t0}) I(x_t = 0) + \gamma(z_{t1}) \log \Pr(X_t = x_t \,|\, Z_t = 1, \phi). \end{aligned} \tag{3.6}$$

Next, we seek an efficient procedure for evaluating the quantities $\gamma(z_{tk})$ and $\xi(z_{t-1,j}, z_{tk})$. The forward–backward algorithm (Baum and Eagon, 1967; Baum and Sell, 1968) is used to accomplish this. First, we define the *forward variable* as

$$\alpha(z_{t,k}) = \Pr(X_1 = x_1, \ldots, X_t = x_T, Z_t = k \,|\, \boldsymbol{\theta}) \quad k = 0, 1.$$

α can be solved for inductively:

(1) Initialization: $\alpha(z_{1,0}) = 1 - \pi$ $\alpha(z_{1,1}) = \pi \Pr(X_1 = x_1 \,|\, Z_1 = 1, \phi)$.
(2) Induction: For $k = 0, 1$ and $1 \le t \le T - 1$,

$$\alpha(z_{t+1,k}) = [\alpha(z_{t,0}) A_{0k} + \alpha(z_{t,1}) A_{1k}] \Pr(X_t = x_t \,|\, Z_t = k, \phi). \tag{3.7}$$

Below, we will use the fact that $\Pr(\mathbf{X} = \boldsymbol{x} \,|\, \boldsymbol{\theta}) = \alpha(z_{T,0}) + \alpha(z_{T,1})$. We next need to define the *backward variable*, the probability of the partial observation sequence from $t + 1$ to T:

$$\beta(z_t) = \Pr(X_{t+1} = x_{t+1}, \ldots, X_T = x_T \,|\, Z_t = z_t, \boldsymbol{\theta}).$$

$\beta(z_t)$ can be solved for inductively as follows:

(1) Initialization: $\beta(z_{T,k}) = 1 \; k = 0, 1.$
(2) Induction: For $k = 0, 1$ and $t = T-1, \ldots, 1,$

$$\begin{aligned}\beta(z_{t,k}) = A_{k,0} \Pr(X_{t+1} = x_{t+1} \,|\, Z_{t+1} = 0)\beta(z_{t+1,0}) \\ + A_{k,1} \Pr(X_{t+1} = x_{t+1} \,|\, Z_{t+1} = 1, \phi)\beta(z_{t+1,1}).\end{aligned} \tag{3.8}$$

Finally,

$$\gamma(z_t) = \frac{\alpha(z_t)\beta(z_t)}{\Pr(\mathbf{X} = \boldsymbol{x} \,|\, \boldsymbol{\theta})} \tag{3.9}$$

$$\xi(z_{t-1}, z_t) = \frac{\alpha(z_{t-1}) \Pr(X_t = x_t | Z_t = z_t, \phi) \Pr(Z_t = z_t | Z_{t-1} = z_{t-1})\beta(z_t)}{\Pr(\mathbf{X} = \boldsymbol{x} | \boldsymbol{\theta})}. \tag{3.10}$$

The M Step. In the M step, we maximize (3.6) with respect to $\boldsymbol{\theta}$. Maximization with respect to π and $\mathbf{A}$ is easily achieved using appropriate Lagrange multipliers. Taking the derivative with respect to μ results in a closed form update as well.

$$\begin{aligned}\hat{\pi} &= \frac{\gamma_{11}}{\sum_{j=0}^{1} \gamma(z_{1j})} \\ \hat{A}_{jk} &= \frac{\sum_{t=2}^{T} \xi(z_{t-1,j}, z_{tk})}{\sum_{l=0}^{1} \sum_{t=2}^{T} \xi(z_{t-1,j}, z_{tl})} \quad j = 0, 1 \; k = 0, 1 \\ \hat{\mu} &= \frac{\sum_{t=1}^{T} \gamma(z_{t1}) x_t}{\sum_{t=1}^{T} \gamma(z_{t1})}.\end{aligned}$$

The size parameter update is not closed form. From (3.6), we see that it comes down to maximizing $\log \Pr(X_t = x_t \,|\, Z_t = 1, \phi)$ with respect to s, which we achieve through a numerical grid optimization routine.

Scaling. For moderate lengths of chains, the forward and backward variables quickly get too small for the precision of the machine. One cannot work with logarithms, as is the case for independent and identically distributed (i.i.d) data, since here we have sums of products of small numbers. Therefore a rescaling has been developed, and is described in Bishop (2006). Define a normalized version of α as

$$\hat{\alpha}(z_t) = \Pr(Z_t = z_t \,|\, X_1 = x_1, \ldots, X_t = x_t) = \frac{\alpha(z_t)}{\Pr(\boldsymbol{X} = \boldsymbol{x} \,|\, \boldsymbol{\theta})}.$$

Equation (3.7) becomes

$$\hat{\alpha}(z_{t+1}) = \frac{\Pr(X_t = x_t \mid Z_t = z_t, \phi) \sum_{z_t} \hat{\alpha}(z_t) \Pr(Z_{t+1} = z_{t+1} \mid Z_t = z_t)}{c_t},$$

where

$$c_t = \sum_{k=0}^{1} \Pr(X_t = x_t \mid Z_t = k, \phi) \sum_{k} \hat{\alpha}(z_{t-1,k}) \Pr(Z_t = k \mid Z_{t-1} = k)$$

is the constant required to normalize $\hat{\alpha}(z_t)$. Equations (3.8), (3.9), and (3.10) become

$$\hat{\beta}(z_t) = \frac{\sum_{z_{t+1}} \hat{\beta}(z_{t+1}) \Pr(X_{t+1} = x_{t+1} \mid Z_{t+1} = z_{t+1}, \phi) \Pr(Z_{t+1} = z_{t+1} \mid Z_t = z_t)}{c_{t+1}}$$

$$\begin{aligned} \xi(z_{t-1}, z_t) &= c_t \hat{\alpha}(z_{t-1}) \Pr(X_t = x_t \mid Z_t = z_t, \phi) \\ &\quad \times \Pr(Z_t = z_t \mid Z_{t-1} = z_{t-1}) \hat{\beta}(z_t) \\ \gamma(z_t) &= \hat{\alpha}(z_t) \hat{\beta}(z_t). \end{aligned}$$

3.4.3. *New edges*

The approach described above discusses the stochastic modeling of existing edges, and seeks paths upon which the data have deviated from baseline models. Through examination of historic attacks, however, it is clear that new edges can be indicative of attacks.

The subject of new edges is related to link prediction in the social network literature (Liben-Nowell and Kleinberg, 2007), where suggesting "friends" is an important topic. In this chapter, however, we ask about the tail of the probability distribution. Instead of finding most probable new "friends," we seek the most unlikely new edges between pairs of computers, in order to identify anomalies.

For existing edges, one can associate a model with the observed behavior, and then estimate the parameters of the model given observed behavior. A more complicated problem is to estimate the probability of observing a new edge, since we have no existing behavior upon which to base our estimation. Instead, we borrow information from the frequency at which the source and destination nodes make and receive new edges from other nodes in the network.

Specifically, suppose that we observe source node x initiating a new edge to destination node y. To establish a probability of observing this

edge, we propose a logistic model:

$$logit(P_{xy}) = \alpha + \beta_x + \gamma_y,$$

where P_{xy} is the probability of the new edge initiated by x, bound for y; α is an effect for the overall rate at which new edges are produced in the network; β_x is an effect for how often x initiates new edges; and γ_y is an effect for how often y receives new edges.

Maximum likelihood estimation of the above model is computationally expensive. Instead, we use the method of moments estimation (Casella and Berger, 2001), significantly reducing estimation complexity. On large networks, method of moments provides high-quality estimation, since the sample size will be very large. In practice, the estimation requires that we have two graphs, an established graph that provides the existing edges, and another graph, disjoint in time, that allows us to estimate the rate at which new edges (those not in the established graph, but in the second graph) appear.

The new edge model above was not used in the simulation study in Section 3.5, but was used in the real-data section (3.6).

3.4.4. *Alternative hypotheses*

In order to obtain a GLRT, we need to restrict our overall parameter space to allow for alternatives that reflect the types of attacker behavior we wish to detect. These are intentionally kept general, in order to catch a variety of behaviors. We postulate that attacker behavior causes increases to the MLEs of parameters governing the models. This is due to the fact that the attacker must act *in addition to* the normal behavior on that edge. Specifically, referring to the OMM, we propose that attacker behavior causes an increase in the probability of transitioning from the inactive to the active state:

$$H_0 : p_{01} = \tilde{p}_{01} \quad \text{versus} \quad H_P : p_{01} > \tilde{p}_{01}, \tag{3.11}$$

where $\tilde{p}_{01}$ is the historic parameter value.

In the HMM setting, we have more options. We will test three combinations of parameter changes:

$$H_P : p_{01} > \tilde{p}_{01}, \quad H_M : \mu > \tilde{\mu}, \quad H_B : p_{01} > \tilde{p}_{01} \quad \text{and} \quad \mu > \tilde{\mu}. \tag{3.12}$$

In each case, the null hypothesis is that the parameter or two-parameter pair is equal to its historic parameter value.

3.4.5. *P-value calculation*

We seek a p-value for the observed GLRT statistic, λ_γ. Under mild regularity conditions, the GLRT is asymptotically χ^2 with degrees of freedom equal to the number of free parameters in $\mathbf{\Theta}$. However, this does not hold when the true parameters are not on the boundary of $\mathbf{\Theta}$ (see Casella and Berger, 2001, p. 516). If the true parameters are on the boundary, as in the restricted tests we perform (see Section 3.4.4), we will obtain a point mass at zero in the distribution of λ_γ.

Star p-values. We start with the simpler of the two shapes, the star. The number of stars in a graph is just the number of nodes, and therefore, for each node v, we can afford to model the distribution of the GLRT $\lambda_v = \sum_{e \in outedges(v)} \lambda_e$ for the star around v (see [3.2]).

Let Λ_v have the distribution of the λ_v. We model Λ_v as $\Lambda_v = B_v X_v$ where $B_v \sim \text{Bernoulli}(p_v)$ and $X_v \sim \text{Gamma}(\tau_v, \eta_v)$. Since all λ_e in the sum could be zero, Λ_v must have a point mass at zero. This is captured by B_v. To model the positive part of the distribution for Λ_v, the Gamma distribution is attractive since it is equal to a χ^2 distribution with degrees of freedom ν when $\tau_v = \frac{\nu}{2}$ and $\eta_v = 2$. The asymptotic distribution of λ_v is then the sum of independent zero-inflated χ^2 distributed random variables. Thus, we expect the zero-inflated Gamma to be able to model the distribution of λ_v fairly well. The log-likelihood of N i.i.d. samples is given by

$$l(p, \tau, \eta) = \sum_{i=1}^{N} I(\lambda_i = 0) \log(1 - p) + I(\lambda_i > 0)[(\tau - 1) \log \lambda_i - \lambda_i/\eta - \log \Gamma(\tau) - \tau \log \eta]. \quad (3.13)$$

To estimate τ_v and η_v, we use direct numerical optimization of (3.13) over 10 days of non-overlapping 30-minute windows, for each star centered at node v. We denote the MLEs as $(\hat{p}_v, \hat{\tau}_v, \hat{\eta}_v)$. Then for an observed λ_v, the upper p-value is calculated by $P(\Lambda_v > \lambda_v) = \hat{p}_v(1 - F_\Gamma(\lambda_v \mid \hat{\tau}_v, \hat{\eta}_v)$ where F_Γ is the Gamma CDF.

Path p-values. Unlike stars, the large number of paths makes modeling λ_γ for each path prohibitively expensive, both in computation time and memory requirements. Instead, we build a model for each individual edge, and then combine them during the path likelihood calculation. For each edge e, let Λ_e have the null distribution of e's GLRT scores, λ_e. Again, we use a zero-inflated Gamma distribution to model this. Now, however, it will be on a per-edge basis. Once again, this model is motivated by the fact that

asymptotically, the null distribution of λ_e is a zero-inflated χ^2 (with 50% mass at zero if testing one parameter).

Let $\Lambda_e = B_e X_e$ where $B_e \sim \text{Bernoulli}(p_e)$, and $X_e \sim \text{Gamma}(\tau_e, \eta)$, with edge-specific shape τ_e and shared scale η. That is, we have two free parameters for each edge, p_e and τ_e, and a common scale parameter for all edges, η. The importance of the common scale parameter will become clear shortly. We estimate MLEs $\hat{p}_e, \hat{\tau}_e$, and $\hat{\eta}$ using λ_es from non-overlapping 30-minute windows. The likelihood is similar to (3.13), but since each edge has its own τ_e, and a shared η, we have developed an iterative scheme that alternates between estimating η for all edges, and then, for that fixed η, estimating individual τ_e. Since each step of the iteration increases likelihood, the overall procedure increases likelihood.

Once the edge models are fitted, we have all of the information we need to calculate path p-values. Let $\Lambda_p = \sum_{e \in path} B_e X_e$. The 3-path exceedance p-value is the mixture exceedance given by

$$
\begin{aligned}
&P(\Lambda_p > \lambda_p) \\
&\quad = \sum_{b_1=0}^{1} \sum_{b_2=0}^{1} \sum_{b_3=0}^{1} P(B_1 = b_1) P(B_2 = b_2) P(B_3 = b_3) P(\Lambda_p > \lambda_p \mid b_1, b_2, b_3) \\
&\quad = \sum_{b_1=0}^{1} \sum_{b_2=0}^{1} \sum_{b_3=0}^{1} \left(\prod_{i=1}^{3} (1 - \hat{p}_i)^{1-b_i} \hat{p}_i^{b_i} \right) \left(1 - F_\Gamma \left(\lambda_p \mid \sum_{j=1}^{3} b_i \hat{\tau}_i, \hat{\eta} \right) \right)
\end{aligned}
$$

where we used the fact that the sum of Gamma random variables with common scale parameters is again Gamma.

3.4.6. *Threshold determination*

To obtain thresholds, we simulate ten days of per-minute counts for each edge with no anomalies introduced. We then slide 30-minute windows, offset by ten minutes, over the ten days, calculating the minimum p-value in each window, just as would be done in the full scanning procedure. See the scanning procedure discussion in Section 3.5 for a brief discussion of the time-window choices. To achieve a false discovery rate of one alarm per day, we might take the tenth smallest p-value in the resulting list of p-values. But since the windows overlap, we choose to be less conservative, by counting minimum p-values resulting from consecutive windows on the same path as a single p-value, and find the tenth-smallest minimum p-value associated with non-consecutive windows. In this way, alarms over several overlapping

windows only contribute one alarm to the threshold determination, which is exactly the way an analyst would view a series of consecutive alarms.

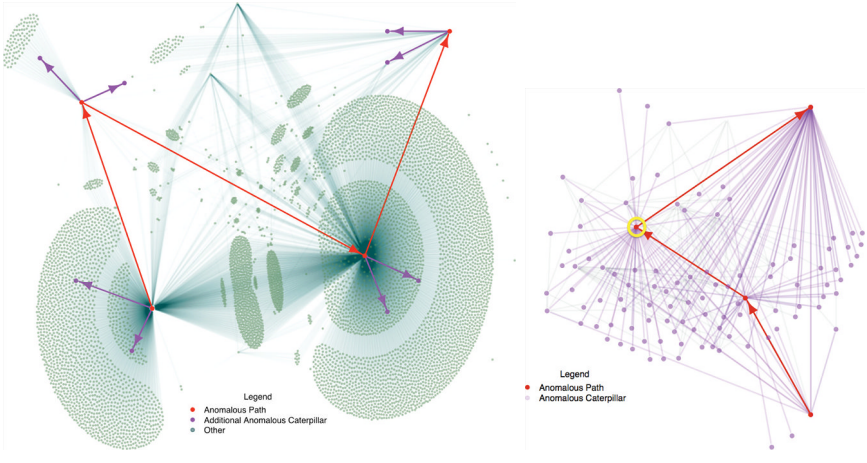
3.5. Simulation Study

In this section we describe a series of simulations. We use both star and path shapes to scan. Using both shapes allows us to directly compare paths with the method of Priebe *et al.* (2005), since the scan shape used in that work is the out-star. We will describe three anomaly shapes introduced into the simulation: the star anomaly, the path anomaly, and the caterpillar anomaly. The interplay between the shape of the true anomaly and the scan shape is significant. Not surprisingly, we will see that a path scan shape is better at detecting a path anomaly, and a star scan shape is better at detecting a star anomaly. On a mixed star/path shape, the caterpillar, stars tend to only identify parts of the anomaly, and paths generally discover the more complete anomalous shape, while both shapes tend to produce additional false edges.

Simulation Procedure. The steps for generating simulated data are:

(1) For each edge, estimate historic parameters from the full 30 days of LANL data (see Section 3.4).
(2) Fit models for the distribution of the λ_γ scores collected on the 30 days of data (see Section 3.4.5).
(3) Obtain a p-value threshold from ten days of simulated, non-anomalous data (see Section 3.4.6).
(4) Simulate 100 days of minute data on each edge according to the historic estimates, except for the set of anomalous edges, where the model parameters are adjusted to introduce an anomaly (see below).

Anomalous Shapes Introduced into the Simulation. Three anomalous shapes were used in the simulation, on two different areas of the LANL network. The shapes are visualized in Figure 3.5. On each edge in each shape, the model parameters were adjusted from their historic settings to mimic an attacker, while all other edges (approximately 550k edges) in the network are left at their historic settings. In both subgraphs, a red path is highlighted. These paths will form the *path anomalies*. The purple edges, in addition to the red edges, form a more general attack, the *caterpillar anomalies*, designed to mimic the attack described in Figure 3.2. Finally, the *star anomaly* was introduced as the set of outgoing edges from the yellow-circled node in subgraph B.



(a) Subgraph A (b) Subgraph B

Fig. 3.5. Star, path, and caterpillar anomalies. Each subgraph has a core path, around which anomalous shapes were introduced. For each core path, the anomaly is plotted, along with the directly connected edges not involved in the anomaly, which are provided for context. Red edges and nodes give the core path, and additional caterpillar nodes and edges are plotted in purple. Every edge in subgraph B is part of the caterpillar for this subgraph. The yellow circle indicates the node at the center of the anomalous out-star.

The additional green edges in subgraph A are to give context for the embedded anomalous subgraph within the larger network. One key difference between subgraph A and subgraph B is that in subgraph A, only two additional purple edges where chosen to be anomalous for the caterpillar anomaly, whereas in subgraph B, every outgoing edge of each node in the path was made anomalous. Yet the red path in subgraph A, call it path A, traverses a much more central part of LANL's network, whereas path B traverses a much less connected area. While these subgraphs do not come close to examining all of the possibilities of traversal in the network, we chose them as exemplars of the interplay between the underlying graph topology and the attack path taken over that topology.

Anomalous Parameter Settings. To insert an anomaly on a star, path, or caterpillar, we modify the historic parameters in each edge of the anomalous shape before simulating, but use the historic parameters for all other edges in the network.

In the OMM simulation, the parameters on the anomaly shape were adjusted letting $p_{01}^{anom} = \hat{p}_{01} + 0.2$ on each of the anomalous path edges

Table 3.1. Anomalous parameter change for simultations. P is an anomalous p_{01} change, M is an anomalous μ change, and B is a change to both p_{01} and μ.

Type	Anomalous Parameter Change
P	$p_{01}^{anom} = \hat{p}_{01} + 0.2$
M	$\mu^{anom} = \hat{\mu} + 1$
B	$p_{01}^{anom} = \hat{p}_{01} + 0.2, \mu^{anom} = \hat{\mu} + 1$

(see Table 3.1). This increase was arrived at after consulting with cybersecurity experts, whose intuition was that likely attacker behavior could be to transition to the active state once every two minutes. We choose to be more conservative, by inserting a one-in-five-minute anomaly.

In the HMM simulations, we introduce three types of anomalies, summarized in Table 3.1. The high-state mean was raised in the M and B anomaly types, reflecting the fact that an attacker may act in a way that increases the historic mean by one count per minute. All parameters not mentioned in each type are left at their historic settings.

Scanning Procedure. Once the data has been generated for a specific anomaly shape and model choice, for each of the 100 days of scanning:

(1) Slide a window of length 30 minutes over the day, offsetting each consecutive window by ten minutes. These choices were made after consulting with experts and examining real attacks. Thirty minutes is sufficient to capture many attack behaviors, but not so long that the true attack is buried in non-attack data. The ten-minute offset was chosen to balance processing time with quick response time, since shorter offsets require more processing, but longer offsets mean longer delays between alarms.
(2) Within each window, select the edges of the entire data set for which there was at least one nonzero count in the window. This creates a subgraph of the overall graph.
(3) For this subgraph, enumerate all 3-paths, and calculate their p-values.
(4) If any path in this window has a p-value below the threshold, record all such paths, and examine no further windows for this day.

The idea behind Step 4 is that once an anomaly is detected, the system would pass the results to an analyst. This analyst would possibly shut down the machines involved, and determine what, if any, true malicious activity was present, before allowing the machines back on the network. Therefore,

Table 3.2. Detection statistics on star and caterpillar anomaly shapes comparing path and star scanning shapes. AEF (average edge frequency) is the average number of true anomalous edges per number of detected edges. PAD (percent anomalous detected) is the average percentage of the truly anomalous edges that were detected. GS (graph size) is the average size of the detected subgraph, which may contain many false edges. Standard errors are given in parentheses.

Anomaly Type	Scan Type	AEF	PAD	GS
Star	Path	0.18(.02)	0.23(.03)	448.50(106.49)
Star	Star	1.00(.00)	1.00(.00)	43.02(.02)
Cat A	Path	0.01(.01)	0.79(.01)	3431.71(279.11)
Cat A	Star	0.02(.00)	0.19(.01)	62.42(4.06)
Cat B	Path	0.24(.01)	0.92(.01)	887.04(106.96)
Cat B	Star	1.00(.00)	1.00(.00)	134.02(.02)

these first detection graphs are the only graphs we analyze in the results, since for any further windows in the day, the anomaly would not be present in the data after forensic analysis was performed.

3.5.1. *A comparison of stars and paths*

As discussed above, a wide variety of simulations was performed for this chapter. We will focus on the differences between stars and paths, when the true anomaly is a star or a caterpillar.

In Table 3.2, we present several statistics related to the detection of the anomalous subgraph. Cat A and B refer to the caterpillar shapes inserted, and correspond to subgraphs A and B of Figure 3.5.

Star Anomaly. Referring to Table 3.2, it is clear that using star windows to scan a star anomaly is much more accurate than using paths. In fact, the star scan detected every true anomalous edge, and only those edges, for 99% of the days. Paths picked up some portion of the anomalous star, but at the cost of a much larger detected graph.

Caterpillar Anomaly. Recall from Figure 3.5 that Cat A is a very light anomaly (only 11 edges) whose core is a very well-connected path. Path scanning detected the anomaly on the first window, but stars had a non-trivial time to first detection, as seen in Figure 3.6. While the AEF value was fairly low using paths, on average nearly the entire anomaly was detected. The star scan, on the other hand, consistently detected only one of the three stars in the caterpillar. The other two stars, and core path edges, were not detected at all by the star scan.

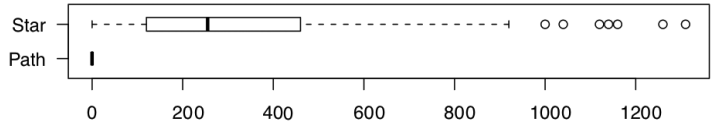


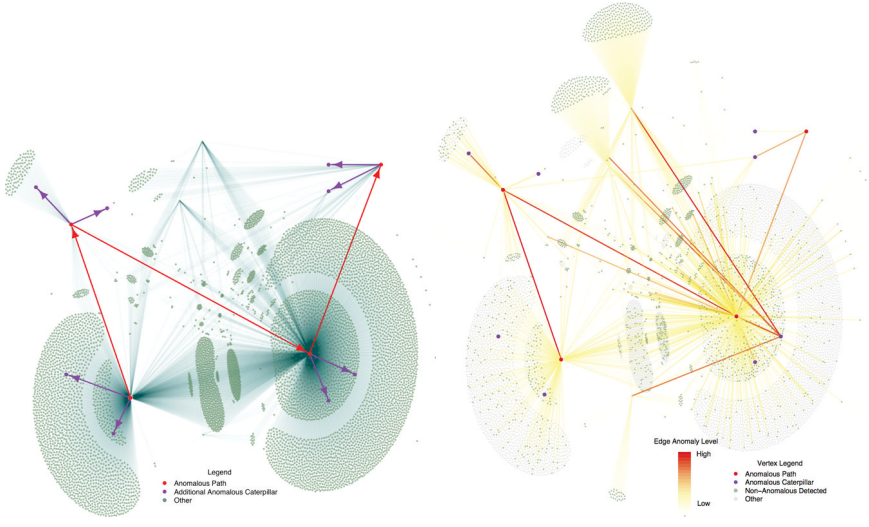
Fig. 3.6. Caterpillar A time to detection. The x-axis is in minutes from the beginning of the anomaly.

Cat B is a much heavier anomaly, involving every out edge of core Path B, for a total of 174 edges. But Path B is much more lightly connected in the graph, and therefore far fewer paths run through the anomaly than Path A. We might expect path scanning to suffer, as a result. However, path scanning performed even better than it did for Cat A, detecting more truly anomalous edges on average, and fewer falsely detected edges. Fewer false edges can be explained by the fact that fewer paths were inspected, but better detection of the true anomaly has to do with the difference between historic and anomalous parameters on the true anomaly. This is clear from looking at the historic versus anomalous parameter values, but since there were 174 sets of parameters to compare, we omit this analysis.

Next, we will discuss visualizations of the detected graphs. A detection using path scans corresponds to the union of every path that had a p-value smaller than the false discovery rate (FDR) threshold. Paths may overlap on a set of edges, and so for each detected graph we can count the number of times each edge appears in any detected path. This count can then be used to color edges in a heat map of the detection. A heat map resulting from using a path shape on the anomaly given by Caterpillar A is presented in Figure 3.7.

On the left, we see Caterpillar A embedded in its 1-hop containing graph (i.e., all edges emanating from the nodes in the caterpillar). On the right, we see the path-scan heat map of a single detected window. The core path is brightly colored, as these edges were detected very frequently. In addition, while some edges may be dim, for this detection at least, every true anomalous edge is present in this detection graph. These colors not only give the analyst an ordering of importance of the edges, but also provide an overall view of the structure of the anomaly. It additionally highlights the ability of paths to form more general shapes of detection than just the core shape.

To contrast with the visualization in Figure 3.7, in Figure 3.8 we provide a visualization produced by using a star-shaped window to scan the same Caterpillar A anomaly. One can see that most of the anomalous edges were



(a) Caterpillar A (b) Detection Heat Map

Fig. 3.7. Anomaly graph and heat map for Caterpillar A. The true anomaly is given on the left, with anomalous edges colored red and purple. Green nodes and edges are uninvolved in the anomaly, but are provided to give context. The more green edges, the more chance of false discovery. The detected heat map is displayed on the right, with darker red indicating more evidence of an anomaly.

missed, and many false edges were detected. Since star scans cannot overlap, there is no concept of heat in this visualization. Red indicates the edge was detected, and the light blue nodes are to provide the graph context.

3.6. Real Network Detections

Since our goal with this work is a system that runs in real time, on real networks such as LANL's internal network, we considered it an important milestone to run, at least in prototype form, a path scan on real data from such a network. Therefore, in this section we describe two path-scan analyses of data contained in LANL's historic data archives.

3.6.1. *Detection of user change*

The HMM models whose parameters were estimated from real data in 2011, ending 30 days later, were used for this study. We chose to test for an elevation in p_{01}. Initially, we attempted a test of both parameters, but we

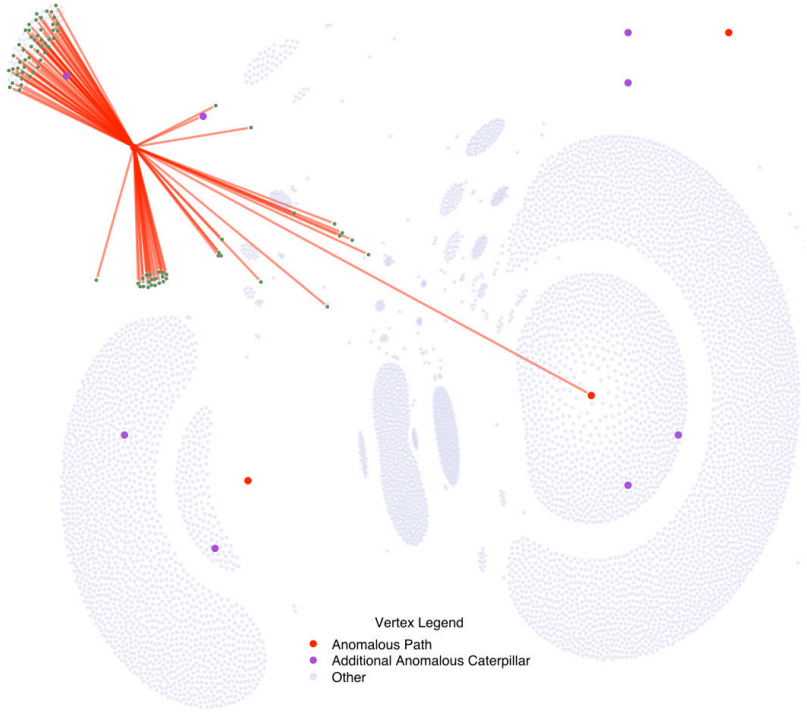


Fig. 3.8. Star scanning results for Caterpillar A. Detected edges are plotted in red. The light blue nodes are to provide context.

encountered several numerical problems with testing the high-state mean that could only be resolved with a more custom model for this data. In addition, testing for only a p_{01} change had good performance in simulation, especially when the mean was also anomalously high.

Since we used simulated data to set p-value thresholds in the simulations, we require new thresholds when preparing to scan on real data. Therefore, the next ten days of data, starting March 2 and ending March 12, were used to obtain these thresholds, using a discovery rate of one detection per day. Finally, the next 20 days were scanned using 3-paths.

Note that completely unestimated (new) edges did arise in this data set. For this example, we used these new edges in enumeration, allowing estimated edges to be "bridged" by the new edges in the paths. But we did not use the data on these new edges to contribute to the path GLRT score.

In these 20 days, 38 unique detections occurred, which is not unreasonable for this example. We would have expected 20 detections, but the larger number of detections can be attributed to estimation error in setting the threshold, random fluctuation in the number of detections, and/or some deterioration of the model fits over time. In practice, a larger sample would be used for setting the threshold, and these models would be updated over time.

While many of these detections look interesting, we choose to describe the most anomalous one, i.e., the detection that achieved the minimum p-value in the 20 days, in detail. A heat map of this detection is provided in Figure 3.9.

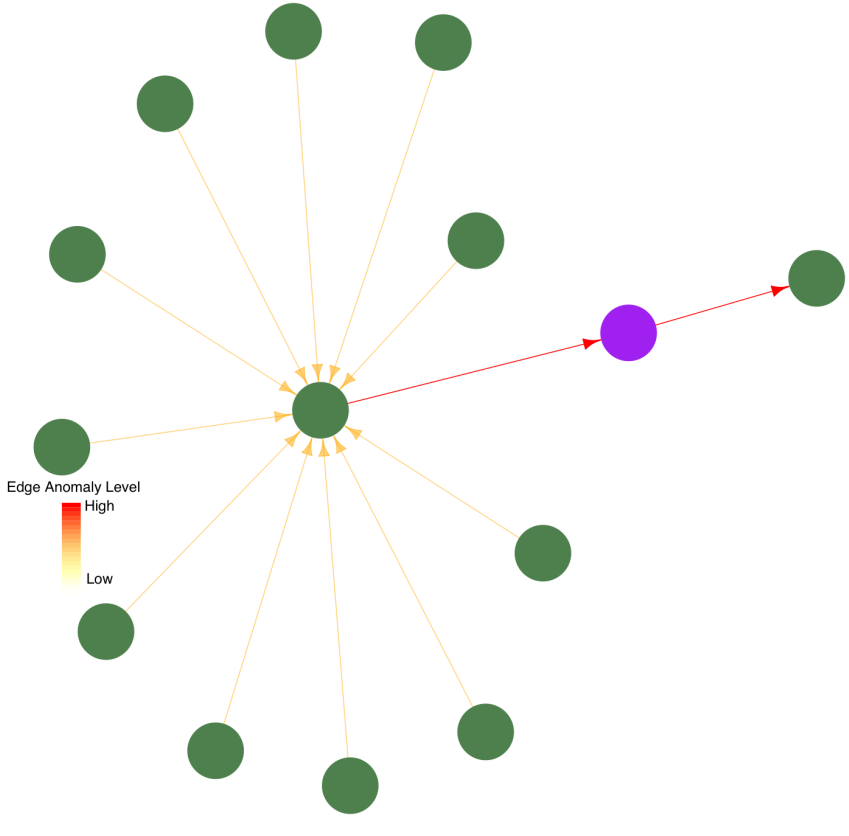


Fig. 3.9. User-change detection heat map.

In this figure, we see a star of 11 nodes around a central node, along with a 2-path (red) beginning at the central node. This central node is a calendaring server, and the star nodes around it are user machines making connections to it to get the updated meeting schedule. The red edge leading out from the calendaring server is an edge to a user machine, given in purple. The edge leading out from this user machine is an email server.

On March 22, at around 11:00 am, this graph was detected as anomalous. Each of the edges leading to the calendaring server were identified once in the detected graph, and the two red edges were detected 11 times. This implies that the 11 3-paths starting at each star node all passed through both red edges.

When we conducted a forensic analysis of this graph, two relevant facts emerged. First, the rate of counts on the two red edges increased significantly, while the edges leading into the star did not. This indicated an embedded anomalous 2-path in the 3-paths, which is apparent in Figure 3.9. Second, it was determined that the purple node changed significantly. Specifically, the purple machine's user changed.

Since the user changed, the settings of applications that accessed the network from this computer changed. While this event could be explained by normal network usage, it is nonetheless a very promising detection. Without the legitimate user change, this would be an extremely interesting anomaly, possibly indicating the presence of an attacker, and one that our security team would investigate thoroughly. Since our goal was to implement a practical monitoring system that detects just such anomalies, this finding is very encouraging.

3.6.2. *Detection of real attack*

While detecting a change in user was a compelling result, the real purpose of this work is to detect true attackers. Therefore, we present a detection of one such event. This event occurred over a period of nine days.

Paths were used as the scan shape and the OMM model was used. A 30-day period immediately preceding the attack was used as training data. The first 15 days were used to provide a baseline graph of existing edges, and the second 15 days were used to define new edges and estimate the new edge rates and OMM parameters for existing edges.

In Figure 3.10, we give the heat map of the most anomalous window. Again, this is the union of those paths that exceeded the threshold, and the minimum p-value path in this detection was the smallest p-value in the entire time period.

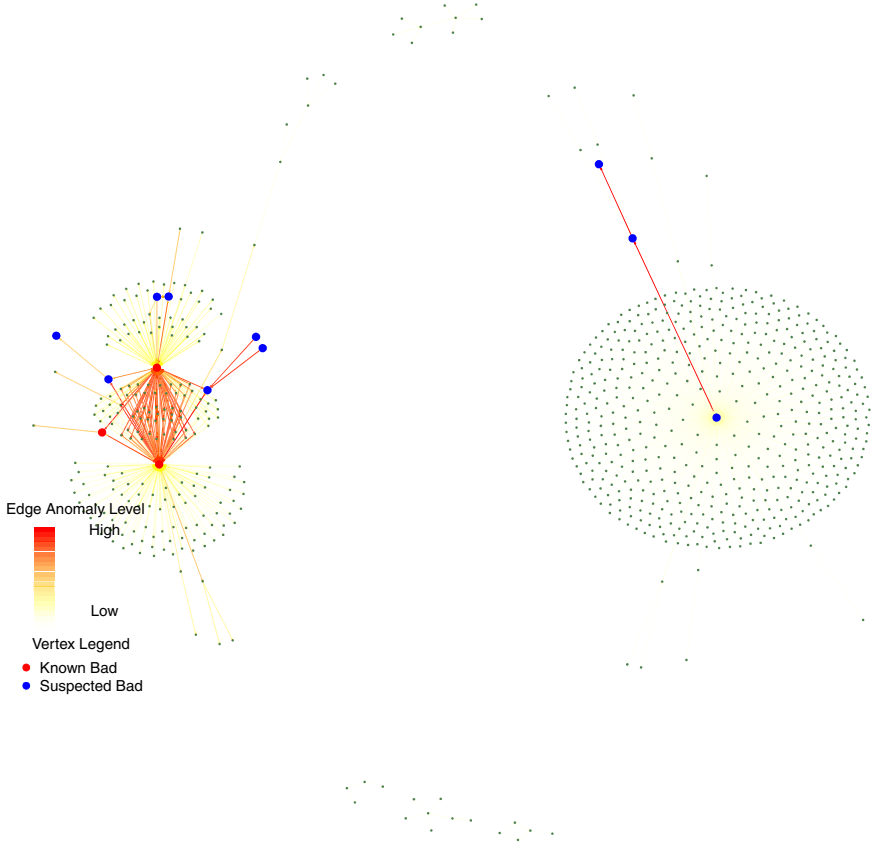


Fig. 3.10. Real detection heat map. Red nodes were independently verified as being compromised during the attack. Blue nodes were highly suspicious (low p-values on edges associated with these nodes).

3.7. Conclusions and Future Work

We have described a method for detecting anomalous activity where data is defined over time on edges in an underlying graph structure. We motivated the need for anomaly detection in this setting with the example of an attacker traversing a computer network. Attacks can be very localized, and so we introduce a method of windowing locally in the time $\times$ graph space. In each window, we calculate a scan statistic indicating whether or not the data in this local window is behaving according to a historic model. Many attacks have, at their core, a path that is the result of an attacker

traversing through the network. Therefore, we have introduced k-paths as a versatile type of local graph window.

We presented the results of simulations and real-data examples. The simulations provide insight into system performance on a variety of different anomalies and testing schemes. The real-data examples are exciting, since we have detected the very activity we set out to detect. We presented heat maps that should aid in the forensic investigation of detected graphs. This system is operational on LANL's computer networks, and has already identified several interesting events.

Most of our work so far has been focused on the general framework of scanning. Further work is focused on developing models that better represent computer edge data. The OMM and HMM presented here are not entirely sufficient to model the network data to our satisfaction. The new edge model appears to be very effective, and requires very lightweight estimation.

In addition, the data exhibit daily and weekly patterns. As seen in Figure 3.1, the middle part of the work day has higher counts than at night. Different schedules on different days can also be seen. Thus, the parameters of the model should also be allowed to change smoothly through the day, similar in spirit to the diurnal and weekly patterns modeled on telephone call networks presented in Lambert and Liu (2006).

Another avenue of investigation lies in the graph shapes. More general shapes are needed to avoid bias on our current knowledge of attack behavior. For example, one could enumerate all subgraphs with at most k edges. The issue then becomes one of limiting the number of subgraphs enumerated for computational reasons. We are investigating methods for allowing the data to suggest the enumerated subgraphs, rather than having fixed shapes such as paths or stars, but this work is very nascent.

Yet another research direction is the handling of data collected at each host. We believe there is strong signal in host data for identifying attacks. New processes and services, along with open files, all have promise. To collect this data requires substantial software engineering, something this team is developing currently. Once the data is collected, we will model it and include it in subgraph likelihoods just as we have done for the edges in this chapter.

To conclude, we feel that this approach is very promising. We have already identified attacks, yet we also feel that there is much to be done. Ultimately, we aim to reduce false positive rates to extremely low levels, allowing for highly accurate detection performance. The framework

of identifying local graph structures, followed by a measure of deviation from baseline models, we feel, is a very promising approach to the task of computer network attack detection.

Acknowledgments

This manuscript/publication has been authored by Los Alamos National Security, LLC under Contract No. DE-AC52-06NA25396 for Los Alamos National Laboratory with the U.S. Department of Energy. The United States Government retains, and the publisher, by accepting the article for publication, acknowledges that the United States Government retains, a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for the United States Government.

References

Baker, J. (1975). The dragon system–an overview, *IEEE T. Acoust. Speech* **23**, 1, pp. 24–29.

Baum, L. and Eagon, J. (1967). An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology, *Bull. Amer. Math. Soc.* **73**, 3, pp. 360–363.

Baum, L. and Sell, G. (1968). Growth functions for transformations on manifolds, *Pac. J. Math.* **27**, 2, pp. 211–227.

Baumes, J., Goldberg, M., Hayvanovych, M., Magdon-Ismail, M., Wallace, W. and Zaki, M. (2006). Finding hidden group structure in a stream of communications, *ISI*, pp. 201–212.

Baumes, J., Goldberg, M., Magdon-Ismail, M. and Wallace, W. (2004). Discovering hidden groups in communication networks, *ISI*, pp. 378–389.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing, *J. Roy. Stat. Soc. B Met.* **57**, 1, pp. 289–300.

Bensley, S., Amsden, P., Lyons, G., Amweg, J. and Calato, P. (1997). Cabletron's light-weight flow admission protocal specification version 1.0 (Network Working Group, Request for Comments: 2124).

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning* (Springer, New York, NY).

Brownlee, N., Mills, C. and Ruth, G. (1997). Traffic flow measurement: Architecture (Network Working Group, Request for Comments: 2722).

Casella, G. and Berger, R. (2001). *Statistical Inference* (Duxbury Press, Pacific Grove, CA).

Chandola, V., Banerjee, A. and Kumar, V. (2009). Anomaly detection: A survey, *ACM Comput. Surv.* **41**, 3, p. 15.

Collins, M. and Reiter, M. (2007). Hit-list worm detection and bot identification in large networks using protocol graphs, in *Recent Advances in Intrusion Detection*, Proceedings of the 10th International Symposium, RAID. 5–7 September 2007 (Springer, New York, NY), pp. 276–295.

Doucet, A., De Freitas, N. and Gordon, N. (2001). *Sequential Monte Carlo methods in practice* (Springer, New York, NY).

Forrest, S., Hofmeyr, S., Somayaji, A., Longstaff, T. *et al.* (1996). A sense of self for unix processes, *IEEE Symposium on Security and Privacy* (IEEE COMPUTER SOCIETY), pp. 120–128.

Glaz, J., Naus, J. and Wallenstein, S. (2001). *Scan Statistics* (Springer, New York, NY).

Heard, N., Weston, D., Platanioti, K. and Hand, D. (2010). Bayesian anomaly detection methods for social networks, *Ann. Appl. Stat.* **4**, 2, pp. 645–662.

Kolaczyk, E. (2009). *Statistical Analysis of Network Data*: *Methods and Models* (Springer, New York, NY).

Kulldorff, M. (1997). A spatial scan statistic, *Commun. Stat. Theor.* **26**, 6, pp. 1481–1496.

Lambert, D. and Liu, C. (2006). Adaptive thresholds: Monitoring streams of network counts online, *J. Am. Stat. Assoc.* **101**, 473, pp. 78–88.

Lambert, D., Pinheiro, J. and Sun, D. (2001). Estimating millions of dynamic timing patterns in real time. *J. Am. Stat. Assoc.* **96**, 453.

Liben-Nowell, D. and Kleinberg, J. (2007). The link-prediction problem for social networks, *J. Am. Soc. Inf. Sci. Tech.* **58**, 7, pp. 1019–1031.

Loader, C. (1991). Large-deviation approximations to the distribution of scan statistics, *Adv. Appl. Probab.* **23**, 4, pp. 751–771.

Lu, Q., Chen, F. and Hancock, K. (2009). On path anomaly detection in a large transportation network, *Comput. Environ. Urban.* **33**, 6, pp. 448–462.

Mukherjee, B., Heberlein, L. and Levitt, K. (1994). Network intrusion detection, *IEEE Network* **8**, 3, pp. 26–41.

Naus, J. (1982). Approximations for distributions of scan statistics, *J. Am. Stat. Assoc.* **77**, 377, pp. 177–183.

Noble, C. and Cook, D. (2003). Graph-based anomaly detection, in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM), Washington, DC, pp. 631–636.

Phaal, P., Panchen, S. and McKee, N. (2001). Inmon corporations sflow: A method for monitoring traffic in switched and routed networks (Network Working Group, Request for Comments: 3176), Tech. rep., Technical report, Internet Engineering Task Force (IETF).

Pohlmeier, W. and Ulrich, V. (1995). An econometric model of the two-part decisionmaking process in the demand for health care, *J. Hum. Resour.* **30**, 2, pp. 339–361.

Priebe, C. E., Conroy, J. M. and Marchette, D. J. (2005). Scan statistics on enron graphs, in *Workshop on Link Analysis, Counterterrorism and Security at the SIAM International Conference on Data Mining* (Newport Beach, CA), pp. 229–247.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition, *Proceedings of the IEEE* **77**, 2, pp. 257–286.

Salamatian, K. and Vaton, S. (2001). Hidden Markov modeling for network communication channels, *Perf. E. R.* **29**, pp. 92–101.

Stallings, W. (1987). *Handbook of Computer-communications Standards*: *The Open Systems Interconnection* (*OSI*) *Model and OSI-related Standards*, Volume 1 (Macmillan, New York, NY).

Ye, N. and Li, X. (2000). A Markov chain model of temporal behavior for anomaly detection, in *Proceedings of the 2000 IEEE Systems*, *Man*, *and Cybernetics Information Assurance and Security Workshop*, Vol. 166 (Oakland, CA), p. 169.

Yeung, D. and Ding, Y. (2003). Host-based intrusion detection using dynamic and static behavioral models, *Pattern Recogn.* **36**, 1, pp. 229–243.

Chapter 4

Characterizing Dynamic Group Behavior in Social Networks for Cybernetics

Sumeet Dua* and Pradeep Chowriappa

Program of Computer Science, Louisiana Tech University,
121, Nethken Hall, Dan Reneau Drive, Ruston, LA 71272, USA
sdua@coes.latech.edu

This chapter is an attempt to characterize users sentiments or opinions for the creation of ad hoc communities over social networks for effective situational awareness. We believe that the mined patterns in user opinions can act as indicators of potential threats in cyberspace. This chapter proposes a novel data-mining approach to identify ad hoc communities and track these communities to effectively identify users and topics that influence the dynamics of a community over time.

4.1. Introduction

The recent excrescence of social networking (SN) and social media (SM) as a medium of communication cannot be overlooked primarily for its far-reaching applications and outreach. Both SN and SM have developed from a means of casual communication to "virtual glue" that connects individuals over cyberspace. This constantly evolving and dynamic cyber system provides a deluge of data and information that can be exploited to enhance the situational awareness (SA). Existing techniques to achieve SA work by modeling the structure of user communities. These techniques employ standard visualization and quantitative tools to measure correlations between profiles of users within a community and users across communities. Though SA in SN lacks a clear definition and the use of a social network-based SA system is still in the conceptual form, this chapter is aimed at exploiting SN for effective SA.

*corresponding author.

The transition of the World Wide Web (Web 1.0) to the semantic web (Web 2.0) has enabled technologies to move away from just websites and portals to social networking and social media sharing. This transition has not only enabled online collaborations between individuals across the globe, but has laid the foundation for a more interoperable system, with diverse subsystems and organizations working synergistically. With the emergence of mobile networks, we move towards a more intelligent web (Web 3.0), and the growing footprint for an individual over cyberspace has become more significant. There is a need for data mining, machine learning, and recommendation systems for a more intuitive web that can adapt to an individual's needs.

The current Web 2.0 provides a unified framework for the sharing and reuse of data across platforms, applications, and organizations alike for all users. Web 3.0, on the contrary, focuses on establishing relationships between data. One can therefore foresee the characterization of an individual over cyberspace by the data he possesses and the data he shares with others over the same space. This vision for a user-specific web is, however, plagued by challenges of data growth and uncertainty. Handling these data challenges is vital to establishing a well-connected semantic web. Despite these challenges the semantic web provides the avenue to revolutionizing the social structure in which individuals and communities have a stronger presence.

Cybernetics is an area of research that has its theoretical underpinnings in social network analysis and applications in online communities. These online communities include wikis, social network communities, collaborative book marking, and social tagging (to name a few). Cybernetics in this chapter is perceived in three key aspects: surveillance, event management, and threat analysis. Each of these aspects affects three prominent applications of cyber security, namely: data forensics, community intelligence, and incidence management.

Situational awarenes can be described as the effective recognition and realization of a system's (or an organization of users over the web) performance. The performance of a system is defined as the system's ability to achieve set deliverables that the system is expected to support. Effective SA relies on the prediction of events (threats) both internal and external to the system. In a social network, users are related to each other if they share similarities in interests referred to as topics of interest. These topics of interest are dynamic for a user and change with the user's interest and activity over the social network. This change in activity helps keep track of users' likes and dislikes that are vital in the realization of a situationally aware system.

Groups of users in a system that share similar interests are referred to as ad hoc communities. The tracking of ad hoc communities enables the detection of events, tracking of existing events, and summarizing all events. Furthermore, the tracking of communities establishes the complex interplay between users of communities in a cyber system.

Of the many challenges of SA, some are related to the problem of maturity. Here security should encompass a wide variety of cyber-security issues, from the collection of log information to the analysis of outliers and anomalies. Secondly, there is the challenge of handling the data deluge associated with social networks. Data over social networks are semi-structured. The development of techniques to manage and enable analysis of a diverse set of data types is challenging. Data management in a cyber system is vital as the confidence of predicting events is tied to the quality of data. These predicted events are succinct to the creation of a system capable of identifying zero day events (attacks), rather than a reactive system. The next challenge is the creation of scalable and reliable tools for SA. Currently, SA and its associated challenges focus on networks to answer two predominant issues: the determination of the current state of the system, and comparing the current conditions with normal conditions to identify potential inconsistencies that might indicate a threat. With large amounts of SN data unstructured and varied, creating newer tools to visualize this data is still an open challenge.

This chapter is targeted towards describing the significant role of cybernetics and SN in SA. We focus on the extraction, inference, and testing of higher-order feature spaces for the identification and characterization of ad hoc communities in a social network. We highlight techniques that exploit the behavior of individuals over an SN. We demonstrate a novel graph theoretic approach that exploits user-behavior patterns for effective ad hoc community detection. Leveraging the concepts of tag sense disambiguation, this approach effectively gauges the behavior of a user in a social tagging SN. Furthermore, we build on the concepts of inter-transaction mining over time to track the evolution of behavioral patterns within communities. In conclusion, we discuss future directions and potential applications of the proposed system.

4.2. User Interaction Pattern Analysis

As "social" is the *phrase du jour*, we foresee rampant growth of social web techniques. There has been an increased interest in social network analysis for identifying structure and semantics and providing better information

management and retrieval systems in this chaos of information overload. As we move toward a more socio-semantic web, identifying structure in web content will continue to provide semantics and automatic machine translation for users. This automatic machine translation provides netizens, citizens of the internet, with a curated and contextual web experience.

The most important criterion being similarity in interests, we employ a relevancy metric to link similar contributors and filter dissimilar ones. The significance of the research presented in this work is focused on identifying ad hoc user communities.

To allow us to link and filter users, we must first determine and extract the best features that capture the similarity among contributors. Although shared interests are connected implicitly in the interest domain, they are not connected explicitly in the connectivity domain. Thus, we aim to identify different sets of contributors who are interconnected through a particular topic of interest, and form groups that we refer to as ad hoc communities.

With the proliferation of collaborative tagging platforms, there has been a recent surge in the study of textual keywords (tags) applied by users to annotate web content (Lindstaedt *et al.*, 2009; seok Min *et al.*, 2009). Tags are examined to generate ways of identifying similarities between contributors. In social settings, contextualized tags facilitate online collaboration among contributors with similar interests, which provide avenues for personalized information exploration. However, this requires that first an ad hoc community be identified based on the semantic context of user information need. This semantic context is captured from the user's tag usage profile.

This ad hoc community detection is a valuable tool required in grouping related items based on relevancy identified by similarity. Community detection is used in two ways in social network mining. First, it is used to identify a structure in the form of social interest topics in which a community is formed by a group of tags (tag clusters) which identify social interest topics in the folksonomy-based social network. Second, it is used to cluster users into groups (ad hoc communities) based on similarities. We focus on identifying these communities; thus, the use of the term community detection from this point refers to ad hoc communities.

By definition, community detection refers to the identification of clusters in a network, and clusters refer to groups of nodes. The nodes in this group have more connections among each other than with nodes in other clusters (Fortunato, 2010). The task of discovering clusters of nodes in

a network is usually referred to as the problem of discovering community structures within the networks (Newman, 2004). It has been suggested that this method identifies higher-order structures in the networks to unveil insights into the functional organization of communities (Gulbahce and Lehmann, 2008). For community detection, identifying interactions between nodes can help determine communities; thus, establishing links between users (nodes), by identifying implicit social interactions, forms the crux of ad hoc community detection.

It is believed that meaningful tag clusters (consisting of sets of related tags in the tag space) are found to revolve around a particular topic of interest. A tag concept hierarchy is a hierarchy of interest topics identified in the tag space. In this space, an inherent topic–subtopic relationship exists, forming a hierarchical structure. Thus, identifying meaningful groups of related tags from a tag concept hierarchy has been found to enhance a social search by contextualizing tags. Tag contextualization is supported by the formation of tag communities and subcommunities. Therefore, tag concept hierarchy generation, from which we can identify meaningful communities and subcommunities of tags, is crucial. In the process of identifying tag communities, the system reduces a user's options to a limited set of tags, thereby filtering irrelevant information content associated with other tags. Hence, related communities are identified from tag concept hierarchy and provide relevancy to the user's information need or interest. We propose a graph-based information extraction methodology to extract topical features that identify the tendency of users in an SN to associate with other users with similar interests. We hypothesize that an ad hoc community of users sharing similar interests can be identified by overlapping tag clusters in the tag concept hierarchy.

4.3. Motivation

In the participatory Web 2.0, collaborative tagging systems that employ folksonomy information, which is a collection of users, tags, and resources, have been identified as a valuable source of information that can be used to build efficient information retrieval and recommender systems.

In a folksonomy, it has been shown that tags are the most valuable source of information among its three elements. The tag has been shown to be a good tool for the proper indexing, managing, and organizing of web resources, as seen in tag-based information retrieval and recommender systems (Hotho *et al.*, 2006; Zhang *et al.*, 2006). Tags are used to identify

shared knowledge in collaborative tagging systems (Golder and Huberman, 2013) and to identify social interest topics (Li *et al.*, 2008).

More recent information retrieval systems do not rely solely on raw tags from the folksonomies, but require a higher-order relationship between tags. This higher-order relationship between tags forms a semantic or meaningful structure and adds context to the tags. As we move toward a socio-semantic web, ontology and emergent semantic web technologies aim at linking entities within the web. Ontologies are also important for information retrieval (Zhang *et al.*, 2006; Zhou *et al.*, 2008) and recommendation systems (Arazy *et al.*, 2009), because they aid in semantic searches made possible by the ontological knowledge base.

Emergent semantic structure in social networks also exists in the form of ontologies (Mika, 2007). Studies have shown that hierarchical relationships between concepts are formed by the tags in the folksonomy system. These studies have outlined folksonomy applications for better information retrieval systems (Zhou *et al.*, 2008) and a more complete semantic web (Zhang *et al.*, 2006).

The main purpose for grouping/clustering tags is to extract semantics in the web chaos. As individual tags do not have an inherent structure and have a lot of noise in the form of sparseness, ambiguity, and varying granularity levels, we need to find tag clusters (higher-order relationships between tags) to alleviate these problems (Plangprasopchok *et al.*, 2010).

Tag annotations can be leveraged to cluster the tags into groups. Thus, tag use provides more avenues of information exploration and navigation, automatic content annotation (Brooks and Montanez, 2006), user profiling (Gemmell *et al.*, 2008), content clustering (Giannakidou *et al.*, 2008), and tag sense disambiguation (Au Yeung *et al.*, 2009). This tag-use information is the cornerstone of the automatic curation of web content, making the task of machine translation easier and user-intervention free.

The challenge of using this system, however, lies in identifying and linking higher-order relationships between tags based on semantic similarity. Grouping tags by identifying related tags has been done using association rules (Schmitz *et al.*, 2006) as well as clustering techniques (Papadopoulos *et al.*, 2010).

Employing association rules is not always feasible for identifying and grouping related tags, as these techniques rely on user-defined parameters of support and confidence. The clustering of tags based on conventional clustering techniques (Giannakidou *et al.*, 2008) requires that the number of clusters be defined as input. Although shortcomings of conventional

clustering techniques can be addressed using community detection methods on tag graphs, these methods do not consider the overlap among the tag clusters and only consider a disjoint set of tags. The work by Papadopoulos *et al.* (2010) uses overlapping clusters to establish the relationship between tags. We believe that employing the overlapping clique percolation method is best for identifying overlapping tag clusters. This method, although not formally a parameter-free algorithm, is used here in a parameter-independent manner, the details of which are discussed in the following section. This algorithm also finds groups based on inherent properties of tag associations that take into consideration the semantic context of the tag used to group them.

Related research include profiling a user in a social network. Profiling refers to identifying user interests, which can be carried out by analyzing the online activity of a user left behind during his or her web use. Another way to profile users (to capture user interests) is to identify user opinions toward different entities in the web that could be captured from user-generated content. Capturing such user opinions is possible due to a budding research area called opinion mining. This area has been explored as an approach toward information retrieval by Missen *et al.* (2013), who have analyzed words appearing in documents as sources for identifying user opinions/interests in a network. The opinion-mining approach has also been extended to the folksonomy-based social network arena, in which the focus has been on using user-generated tags to identify user interests/opinions as can be seen in Liang *et al.* (2010). Opinion mining has also been explored as a means for providing quality recommendations (Anand and Bharadwaj, 2013). Anand and Bharadwaj carve out trust–distrust networks based on a user's historic preferences (interests). Interests/opinions/preferences have also been explored where users are grouped based on similar opinions (Hua and Haughton, 2012). Shared communities of interest have been discussed by Cha *et al.* (2012).

In a different but related research area, user communities are identified in social networks based on extracted profiles. In this framework, communities are formed implicitly by shared interests. User interests evolve over time and, as a result, user-community membership also changes over time. In related research, Kashoob and Caverlee (2012) have discussed temporal group (community) formation and their transient interests. Temporal group formation emphasizes the study and analysis of social bookmarking communities over time. Our method differs from theirs, however, because we restrict our analysis to finding permanent user communities that share

similar interests over a given time. Kas *et al.* (2012) also discuss the topology changes in social networks based on changing trends and temporal group evolution.

Multiple user interests can result in users having multiple community memberships. This phenomenon causes overlapping community structure in social networks. Because of the potential overlap between memberships and interests, many researchers have studied communities with an overlapping nature, for example, Rees and Gallagher (2012). Overlapping user communities based on co-clustering of tags and users have been studied by Wang *et al.* (2010). The proposed methodology framework for ad hoc community detection aims at identifying discrete social groups of shared interest topics, as opposed to social groups with an overlapping nature.

4.4. Proposed Framework

The proposed framework for ad hoc community detection is based on the popular knowledge discovery in databases (KDD) and is an extension to the framework described by Nair and Dua (2012). Figure 4.1 provides an illustration of the four-step process that includes data preprocessing, feature extraction, and higher-order mining. The fourth step is an extension of temporal analysis of ad hoc communities using mining of frequent patterns. This framework uses a graph-based information extraction approach, involving the steps of topic modeling, user profiling, and community detection.

In this section, some of the concepts required for a better understanding of the folksonomy system are introduced before we explain the steps in our methodology. We define the entities involved in a folksonomy system and the relationships formed between entities that could be leveraged for identifying ad hoc communities of users.

We start by defining a resource as a group of any type of information available on the web that includes a set of web pages, URLs, music, videos, books, academic papers, events, or bookmarks. Here, resources correspond to research articles, and the term is defined as follows.

Definition 4.1. Resources, R, is a finite set of articles $\{a_1, a_2, \ldots, a_n\}$ from the who-posted-what table in the given data set.

Similarly, tag is a textual keyword in the form of metadata, or can be simply stated as a "meta keyword," which is used to annotate or label a resource (based on its information content). Therefore, we define tags as below.

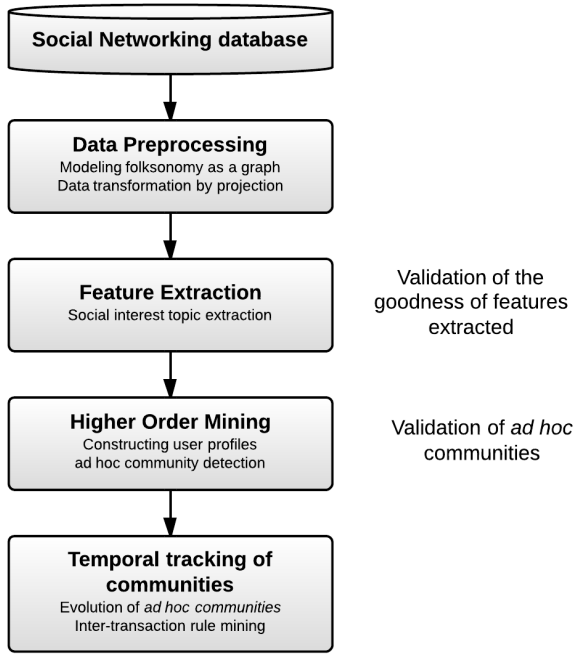


Fig. 4.1. The three-step framework consists of data preprocessing, feature extraction, and higher-order mining.

Definition 4.2. Tags, T, is a finite set $\{t_1, t_2, \ldots, t_m\}$ from the who-posted-what table in the given data set, which is used to label articles available in the data set.

A user is anyone who has a collection of resources and tags to which he or she is associated. This user uses the tags in his or her collection, which we call the user's tag vocabulary, to annotate different resources in the web the user is interested in, making that resource part of his or her library of resources.

Definition 4.3. Users, U, is a finite set $\{u_1, u_2, \ldots, u_p\}$ from the who-posted-what table in the given data set in which each user has different resources (articles) and tags in his or her library.

Folksonomy is the bottom-up classification of resources by users and is a result of using personal free tagging to form a data structure. The users rely on such structures for efficient recollection and retrieval of resources.

Folksonomy is an embodiment of the above-defined entities R, T, and U, and several primary and higher-order relationships exist between

the elements involved in the folksonomy. Primary relationships are formed between these entities when there is a direct relationship between the combinations of any two of these three entities. An example of a primary relationship is a resource-tag graph, where there is an edge between two entities of folksonomy, namely, resources and tags. Higher-order relationships occur in a folksonomy due to the relationship between any of these entities and any of the possible relationships (derivative relationships) between them. An example of a higher-order relationship is an interest graph in which there is an edge between users and social interest topics. Social interest topics, then, are derivations of the relationships between resources and tags. Therefore, we formalize the definition of folksonomy as follows.

Definition 4.4. A folksonomy, F, is a tuple $F = (U, T, R, Y)$, where U, T, and R are finite sets as defined above, and Y is a ternary relation between them, i.e., $Y \subseteq U \times T \times R$, called tag assignments/tag annotations (Schmitz *et al.*, 2006).

In a folksonomy, resources are connected to tags based on tag applications and users are connected to resources present in their library. Through the resources present in a user's library, the user becomes connected to a tag as well. This connection forms the ternary relationship between R, T, and U.

4.5. Data Preprocessing

Data preprocessing here corresponds to data transformation, which is carried out as a projection step. Data transformation starts with modeling the folksonomy's data as a graph. Then this graph is subjected to different projection operations to transform it from a higher dimension to a lower dimension.

Definition 4.5. A folksonomy graph $FG = (V, Y)$ is a tripartite graph, where the vertex set V is partitioned into three disjoint subsets of U, T, and R, such that $U \cap T = T \cap R = U \cap R = U \cap T \cap R = \emptyset$, and every edge $(u, v) \in Y$, connects a pair of vertices u and v where u and v are vertices that "do not" belong to the same disjoint set U, T, or R, i.e., $Y \subseteq U \times T \times R$.

The folksonomy graph FG, which is a tripartite graph, can be projected to its corresponding bipartite representations of user-resource, tag-resource, and user-tag bipartite graphs.

4.5.1. *Projection of tripartite to bipartite graph*

This tripartite to bipartite projection is carried out by considering any two sets of vertices indexed over a third set. This projection step is explained more clearly by using an example of folding a tripartite folksonomy graph to a bipartite tag-resource graph.

In order to capture interactions between resources and tags for individual users, we fold the tripartite folksonomy graph into resource and tag dimensions to get a resource base for every user u_i in FG.

Example 4.1. In order to study a single user, we extract all the tags used and resources tagged by this user. The bipartite tag-resource graph for a user u can be represented as $TR_u = (T \times R, E_{tr})$, where edge set $E_{tr} = \{(t, r) \mid (u, t, r) \in Y\}$.

Definition 4.6. A resource base $RB = (T(u_i) \times R(u_i), E, u_i)$ is a bipartite graph, for each user u_i, such that $u_i \in U$. The vertex set is $T(u_i) \subseteq T$ and $R(u_i) \subseteq R$ with the constraint that $T(u_i) \cap R(u_i) = \emptyset$. The edge set is given by the following equation, $E = \{(t(u_i), r(u_i)) \mid (u_i, t(u_i), r(u_i)) \in Y\}$, where vertex $t(u_i) \in T(u_i)$ and vertex $r(u_i) \in R(u_i)$.

The resource base is a bipartite graph that is a subgraph of the folksonomy graph FG, and includes the elements $R(u_i)$ and $T(u_i)$ indexed for a user u_i. This graph represents mapping of resources to tags in a user's library based on whether a particular tag is used to label a resource.

Definition 4.7. A global resource library (GRL) is defined as $GRL = (T \times R, E_{TR})$ that is a bipartite graph, where the vertex set $T \times R$ is partitioned into two disjoint subsets of T and R, such that $T \cap R = \emptyset$ and the edge set is $E_{TR} = \{(t, r) \mid (t, r) \in Y\}$, and GRL is a set of all the resource bases (RB), of all users U that form a large bipartite graph which can be thought of as a library indexed over resource bases of all users.

The GRL represents connections between different resources R and tags T. This connection is formed based on whether a particular tag is used to label a particular resource where the tags and resources come from the resource bases of all the users.

The GRL is represented using an incidence matrix B, where incidence matrix is used to represent two sets of vertices T and R in a bipartite graph, defined as $B_{m \times n} = \{b_{ij}\}$, where $i = 1 \ldots m$, $j = 1 \ldots n$, $m =$ number of resources R, $n =$ number of tags T, and $m \neq n$.

If the indices $i = j$, then $b_{ij} = 0$;

otherwise, $b_{ij} = \begin{cases} 1, & \text{if there is an edge between } t_i \text{ and } r_j \\ 0, & \text{if there is no edge between } t_i \text{ and } r_j \end{cases}$.

Definition 4.8. A tag graph $TG = (T, E_{TT})$ is a one-mode graph, where vertex set T represents the set of tags and E_{TT} is an edge set, such that $E_{TT} \subseteq \{(t_1, t_2) \mid (t_1, t_2) \in T\}$, and t_1 and t_2 are vertices that belong to the same vertex set T.

4.5.2. *Projection of bipartite to one-mode graph*

The projection of bipartite to one-mode graphs is carried out using matrix multiplication. In this step, the transpose of incidence matrix B is multiplied with B, which represents GRL to generate TG. The tag co-occurrence matrix (TCM), which is the matrix representation of TG, can thus be calculated as $TCM = B^T \times B$.

This matrix multiplication step automatically identifies co-occurrence weights (edge weight between two tags in TG) between two tags as values in TCM. The co-occurrence weight between two tags t_1 and t_2 in TG is aggregated on all the resources they have co-occurred (used/applied together).

This aggregation of weights is performed automatically using adjacency matrix multiplication during the projection step. However, in our methodology, we do not consider edge weights above 1; hence, any weight corresponding to connectivity is represented by a binary weight 1, and 0 represents no connectivity.

The TG is represented by using an adjacency matrix TCM, where adjacency matrix is used to represent a single set of vertices in a one-mode graph and is defined as $TCM_{n?n} = \{tcm_{ij}\}$, where $i = 1, \ldots, n$, $j = 1, \ldots, n$, and $n =$ number of tags T.

If the indices $i = j$, then $tcm_{ij} = 0$;

otherwise, $tcm_{ij} = \begin{cases} 1, & \text{if there is an edge between } t_i \text{ and } t_j \\ 0, & \text{if there is no edge between } t_i \text{ and } t_j \end{cases}$.

The interactions between all the tags represented by TCM correspond to a tag space.

To study and capture the tag interactions occurring in tag space TCM, which are made possible by the resources (i.e., to capture the links formed between all sets of tag pairs because of the common resources used by a pair of tags), we again fold/project the resource-tag bipartite graph GRL

into a one-mode graph of tags. We call the resultant graph a tag graph (TG) and we refer to the TG as the tag concept hierarchy (TCH).

4.6. Feature Extraction

Feature extraction is an information-extraction step in which we search for topic features (set of tags), which can be used to build user profiles. For feature extraction, we follow a graph-based topic modeling technique to extract abstract topics in the form of tag cliques from the tag graph. Hence, we also refer to the feature-extraction step as topic extraction. This process is illustrated schematically in Figure 4.2.

Definition 4.9. Tag clique, TC, in a tag graph $TG = (T, E_{TT})$ is a tag vertex set $\subseteq T$, such that for every two vertices in TS, there exists an edge, which is equivalent to saying that a subgraph (clique) induced by TS is complete.

Thus, TC is a pair (TS, E), where TS is a set of vertices corresponding to tags and E is the edge set, such that $E \subseteq \{(t_1, t_2) \mid t_1, t_2 \in TS\}$, where t_1 and t_2 are vertices that belong to the same vertex set TS and $TS \subseteq T$.

There exists several such tag cliques in the tag graph; hence, we define TC_n as a finite set $\{TC_1, TC_2, \ldots, TC_n\}, \forall TC_i \subseteq T$.

As shown in the equation above, tag cliques are subsets of related tags identified in the tag graph that correspond to tag clusters in such a way that each tag in the cluster is connected to every other tag of the community, forming a cohesive dense subgroup of tags (a clique). All the tags in such a cluster are supposed to belong to a topic area of interest in the real world. We refer to these clusters as social interest topics or interest topics.

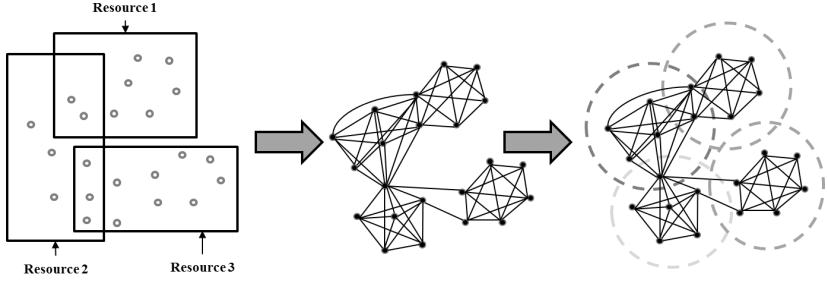


Fig. 4.2. Steps of topic extraction: (a) resource-tag graph (GRL); (b) tag concept hierarchy; and (c) topic extraction (TC).

The identified tag cliques correspond to interest topics. Furthermore, the sub-cliques within the tag clique correspond to subtopics of interest.

4.6.1. *Tag sense disambiguation*

Folksonomy systems have an inherent problem in which many tags are ambiguous and unfamiliar to users. This unfamiliarity is due to a lack of knowledge of the context in which a tag was used, making them fail to understand the meaning of the tag. The methods used to disambiguate the context of each tag's use are referred to as tag sense disambiguations (TSDs).

In our analysis of tag graphs for identifying tag cliques, we need a technique to identify tags that have been used in different contexts to provide for a better TSD. This requirement indicates that we should not just find tag cliques in the tag graph, but that we should also find tag cliques that overlap. Finding tag cliques that overlap makes them effective for disambiguating the context of tag use. In order to reach this objective, we have used the clique percolation method (CPM) (Palla *et al.*, 2005), which is an effective graph theoretic method; we define the details and workings of CPM in the next section.

4.6.2. *Clique percolation method*

The clique percolation method is a graph theoretic algorithm used to analyze a network and identify overlapping community structures of networks (Palla *et al.*, 2005). Formally, this algorithm is not parameter free. However in our work, we used it in a parameter-free way as follows. The implementation of the CPM algorithm finds overlapping communities from graphs based on first locating the maximal cliques that are also overlapping. This algorithm uses these maximal cliques to further build k-clique communities that are not necessarily maximal. Finding these k-clique communities is thus dependent on a parameter k. In our work, we only used the maximal complete subgraphs that were identified by the CPM algorithm and is thus independent of any user-defined parameters. Hence we used the CPM algorithm in a parameter-free manner.

4.6.3. *Tag concept hierarchy*

Due to the above-mentioned required criterion of TSD, our methodology used the clique percolation method (Palla *et al.*, 2005) to find the overlapping tag cliques in the tag graph.

Definition 4.10. A tag concept hierarchy $TCH = (T, E, \delta)$ is a one-mode graph, where T is a set of vertices corresponding to tags, E is a set of edges set $E \subseteq \{(u, v) \mid u, v \in T\}$, and δ represents the hierarchical relationship formed by the tag cliques which can be defined as follows.

Definition 4.11. We define a tag k-clique TC in TCH (where $k \geq 4$) with vertex set TS, $\exists$ vertex set of tags $r \subset TS$, such that for every two vertices in r, an edge exists connecting the two.

The constant δ represents subsumption hierarchical relationships in the form of topic–subtopic relationship, where the general topic subsumes the subtopic forming a hierarchy.

The definition of TCH is an extended version of the definition of TG appending the hierarchy relationships involved in it, and we call this tag graph a tag concept hierarchy from this point forward.

4.6.4. *Effective tag sense disambiguation using tag concept hierarchy*

TCH is a byproduct of the folding of the global resource library into one dimension of tags. Thus, TCH is a tag–tag one-mode graph that represents connectivity between tags (represented by a tag co-occurrence matrix) in the tag space TCM. In such a graph, the constant δ represents the subsumption hierarchy formed between different topics formed by a set of tags. The inherent hierarchical characteristics of TCH can thus be thought of as a semantic structure formed by tags. The social topics (tag cliques) embedded in TCH and corresponding to various concepts formed by tags can be thus thought of as a concept hierarchy formed by the tags in the tag space TCM. Hence, we can also call the tag graph a tag concept hierarchy. When combined, the steps of generating GRL to extracting social interest topics TC form the topic extraction step.

4.7. Higher-order Mining

Higher-order mining is the mining of knowledge based on higher-order features, which are the relationships between the raw features with which we worked in previous steps. In this work, the raw features correspond to tags, and higher-order features correspond to social interest topics formed by tags (tag cliques). In this step, higher-order features, in the form of tag cliques (interest topics), obtained from the previous step are mined.

For the higher-order mining step, we use the following sub-steps: (1) building a user profile; (2) establishing links between users based on interest similarity; and (3) detecting ad hoc communities.

4.7.1. *User profile*

We extract interest graphs by modeling user profiles using a method that includes mapping users to interests (social interest topics) identified in the form of tag cliques in the previous step. This mapping yields a vector for each user, and features correspond to social interest topics. We define an interest graph as follows.

Definition 4.12. An interest graph $IG = (U \times TC, E_{UTC})$ is a bipartite graph, where the vertex set is partitioned into two disjoint sets of vertices, users U and tag cliques TC, such that $U \cap TC = \emptyset$ and edge set, $E_{UTC} = \{(u, tc) \mid (u, tc) \in Y\}$, where u and tc are vertices and do not belong to the same disjoint set U or TC.

Since this bipartite graph captures the mappings from users U to their interest topics TC, we refer to the interest graph a user profile in this work. While building a user profile, we consider the user's degree of interest toward a social interest topic and use this degree of interest to assign weights to the edge that maps the user U to an interest topic TC in the IG. There are three types of user profiles (interest graphs) used in this work: binary, weighted, and term frequency-inverse document frequency (*TF-IDF*)-weighted user profiles. The differences between binary, weighted, and *TF-IDF*-weighted user profiles are based on different weighing schemes.

4.7.2. *Types of user profiles*

The three interest graphs corresponding to binary, weighted, and *TF-IDF*-weighted weighing schemes are represented using three incidence matrices BUP, WUP, and TUP, respectively, in which an incidence matrix is used to represent a bipartite graph with two sets of vertices. We define each of these incidence matrices as follows.

Definition 4.13. A binary user profile $BUP_{m \times n} = \{bup_{ij}\}$ is a matrix, where $i = 1, \ldots, m$, $j = 1, \ldots, n$, and $m \neq n$.

If the indices $i = j$,
then $bup_{ij} = 0$;
otherwise,

$$bup_{ij} = \begin{cases} 1, & \text{if there is an edge between } u_i \text{ and } tc_j \\ 0, & \text{if there is no edge between } u_i \text{ and } tc_j \end{cases}$$

where there is an edge between u_i and tc_j only if $|t(u_i) \cap tc_j| \geq 1$.

Definition 4.14. A weighted user profile $WUP_{m \times n} = \{wup_{ij}\}$ is a matrix, where $i = 1, \ldots, m$, $j = 1, \ldots, n$, and $m \neq n$.

If the indices $i = j$,
then $wup_{ij} = 0$;
otherwise,

$$wup_{ij} = \begin{cases} |t(u_i) \cap tc_j|, & \text{if there is an edge between } u_i \text{ and } tc_j \\ 0, & \text{if there is no edge between } u_i \text{ and } tc_j \end{cases}$$

where there is an edge between u_i and tc_j only if $|t(u_i) \cap tc_j| \geq 1$.

Definition 4.15. A *TF-IDF*-weighted user profile $TUP_{m \times n} = \{tup_{ij}\}$ is a matrix, where $i = 1, \ldots, m$, $j = 1, \ldots, n$, and $m \neq n$.

If the indices $i = j$,
then $tup_{ij} = 0$;
otherwise,

$$wup_{ij} = \begin{cases} Norm\,|t(u_i) \cap tc_j|, & \text{if there is an edge between } u_i \text{ and } tc_j \\ 0, & \text{if there is no edge between } u_i \text{ and } tc_j \end{cases}$$

where there is an edge between u_i and tc_j only if $|t(u_i) \cap tc_j| \geq 1$ and $Norm\,|t(u_i) \cap tc_j| = |t(u_i) \cap tc_j| \times log\frac{n}{|tc_j|}$.

The *TF-IDF* weighting is used to penalize large, common communities (generic topic communities) and boost the significance of small communities (communities very focused on a specific topic).

4.7.3. *Linking users based on similarity*

In any user (social) graph, there is a sense of commonality (similarity) between users. This step in our methodology follows the same notion for linking similar users when similarity is identified by the alignment of user interests identified in the form of social interest topics. Similarity breeds connection (McPherson *et al.*, 2001). Thus, we linked people who shared similar interests.

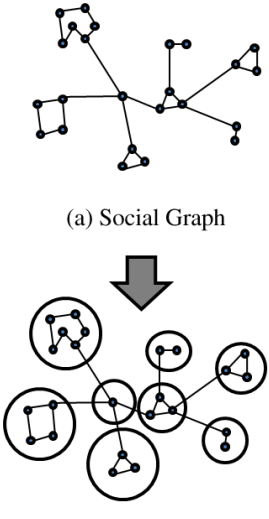
Definition 4.16. A social graph $SG = (U, E_{UU})$ is a one-mode graph, where U represents the vertex set of users and E_{UU} is an edge set, such that $E_{UU} \subseteq \{(u_1, u_2) \mid (u_1, u_2) \in U\}$, where u_1 and u_2 are vertices belonging to

vertex set U and $\exists E_{u_1 u_2}$, if $sim(u_1, u_2) \geq 1$, where $sim(u_1, u_2) = \frac{(u_1.u_2)}{|u_1||u_2|}$ determines the similarity between a pair of users, u_1 and u_2.

In this step, social links were established between users based on their user-interest-profile similarity. In a user profile (interest graph), similarity between two users who are identified by an interest topic vector is calculated by the dot product between these vectors. The interest graph, which is represented by an incidence matrix, is a collection of all user vectors. The dot product operation between every pair of user vectors to calculate the pairwise user similarity happens automatically during the projection step which involves matrix multiplication. Thus, the projection operation extracts social graph SG from the interest graph IG, where the social graph is a graph representing connectivity between users as shown in Figure 4.3(a). This social graph is subjected to the Louvain community detection algorithm to uncover ad hoc communities as shown in Figure 4.3(b).

4.7.3.1. *Projection of bipartite to one-mode graph*

This projection is carried out by matrix multiplication, where the incidence matrices BUP, WUP, and TUP, representing binary, weighted, and



(a) Social Graph

(b) Ad Hoc Communities

Fig. 4.3. Ad hoc community detection. (a) Social graph depicts the connectivity between users in the topical space; (b) Ad hoc communities (implicit communities) are identified in this step from the social graph modeled, which identifies users' affinity toward each other based on topical contexts.

TF-IDF-weighted interest graphs IG are multiplied with the transpose of BUP, WUP, and TUP, respectively, to generate three representations of social graph SG based on different weighing schemes. The user connectivity matrices, binary weighted and *TF-IDF*-weighted are denoted by:

$UCM_B = BUP \times BUP^T$,
$UCM_W = WUP \times WUP^T$, and
$UCM_T = TUP \times TUP^T$.

Therefore, an SG is a byproduct of graph projection operation on IG and connects users. This graph captures the connectivity between users based on the interest graph modeled, where the links between users are identified based on commonalities in user interests that were captured by interest graph.

4.7.4. *Communities in a graph*

In any given set of items, there exists a notion of similarity between a certain subset of items that helps to distinguish that subset from the remaining elements in the set. The same notion extends to a graph, wherein for a given set of vertices, there exists high similarity between a certain subset of vertices. This means that the vertices in the subset are highly similar to each other compared to other vertices in the graph providing a means to separate them from other vertices in the form of communities. This phenomenon is referred to as community structure in graphs.

4.7.5. *Identifying user communities in social graph*

Definition 4.17. An ad hoc community AC in a social graph $SG = (U, E_{UU})$ is a subgraph that contains a vertex set of users $US \subseteq U$, such that connections exist based on modularity maximization between (u_1, u_2) pairs that belong to US.

Thus, AC is a pair (US, E), where US is a set of vertices corresponding to users, and E is the edge set, such that $E \subseteq \{(u_1, u_2) \mid u_1, u_2 \in US\}$, where u_1 and u_2 are vertices that belong to the same vertex set US and $US \subseteq U$.

Several such ad hoc communities exist in the social graph; hence, we define AC_n, as a finite set of ad hoc communities $\{AC_1, AC_2, \ldots, AC_n\}$, $\forall AC_i \subseteq U$.

4.7.5.1. *Detecting ad hoc communities*

Thus, ad hoc communities are a subset of related users (based on modularity maximization) identified in a social graph that correspond to user clusters in such a way that users in a community are more densely connected to each other than to other users in the social graph and form a cohesive dense subgroup (a cluster). All users in such a cluster are supposed to have aligned topic area(s) of interest in the real world. We call these users sharing similar interests ad hoc communities.

Following the definition of community structures in networks, *AC* can be defined as a community. The difference between *AC* and the normal definition of community structure is that, in *AC*, the modules, clusters, or cohesive subgroups are formed revolving around one or more topical areas of interest shared by the users in the community in such a way that the interests of users in a community align more with the users within the community than with users outside. The ad hoc community detection that involves extracting communities from the social graphs was achieved by employing the Louvain community detection algorithm (Blondel *et al.*, 2008). This algorithm was primarily chosen due to it being a modularity maximization-based community detection algorithm, and also because it is parameter free. Among different community detection algorithms available, the Louvain community detection algorithm has the best characteristics. On employment of this algorithm on the social graph, we get ad hoc communities. This step accurately clustered users in the social graph into inherent groups that were not explicitly identified in the connectivity domain, but were captured based on the extracted interest topics. We called these communities ad hoc communities, as shown in Figure 4.3(b), meaning that these users accidentally collaborate for a specific case, based on their actions in the web. In our case, the actions corresponded to their tagging (tag usage) patterns.

4.7.5.2. *Louvain community detection*

The Louvain community detection method (Blondel *et al.*, 2008) is a greedy optimization method that optimizes the "modularity" of a partition of the graph where modularity measures the strength of the division of the graph into modules, groups, clusters, or communities.

4.7.6. *Temporal analysis of communities*

The evolution of social network ad hoc communities is gaining significant interest. The evolution of ad hoc communities over social networks can

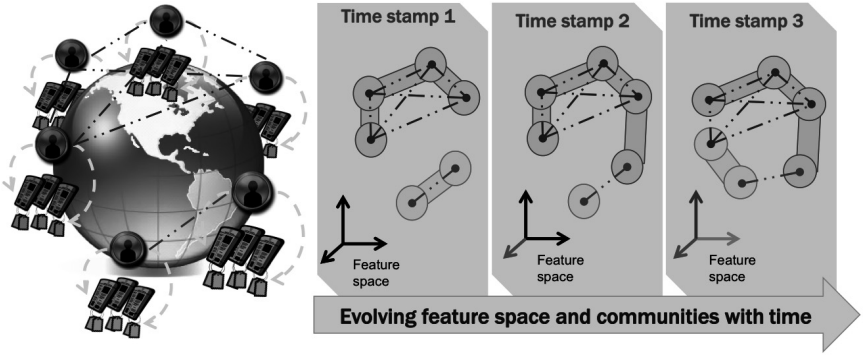


Fig. 4.4. Schematic representation of detecting ad hoc communities and capturing their evolutionary behavior in time.

provide insights into the influence of users and topics over time. Figure 4.4 provides a schematic representation of tracking ad hoc communities. However, this analysis has some computational challenges. Firstly, capturing the evolution of communities. This entails the capturing of the merging, splitting, appearance, or disappearance of communities. Secondly, the evolution of concepts. Here, it is challenging to establish the relevance of topics with time. We objectify that these challenges can be addressed using the concepts of inter and intra-transaction rule mining.

Classical association rule-mining algorithms focus on finding relationships among items, which occur together in a database of transactions (intra-transaction). Recent research has witnessed algorithms directed toward mining rules among items separated by a spatial component, including time (inter-transaction rules). Most inter-transaction association rule-mining algorithms focus on converting an existing transaction database into a mega transaction database using sliding-window segmentation, thereby abstracting the inter-transaction rule-mining problem to intra-transaction rule mining.

Researchers have concentrated on developing inter-transaction association rule-mining algorithms to find relationships among items that are separated by a spatial component and, hence, occur in different transactions. The application of such algorithms is far reaching, and can be used to analyze ad hoc communities with time.

4.8. Conclusion

Based on results obtained we are confident that the proposed data-mining framework can uncover implicitly formed communities of interest. These

communities are not easily conceived in the connectivity domain, but can be identified taking into consideration convergence among user interests. The presented framework takes into consideration that shared topics of interest identified as overlapping tag clusters from a tag concept hierarchy modeled on folksonomy data can be used to model links between users. Using these topics of interest we uncover the social connectivity graphs which are valuable for identifying user communities of shared interests embedded in the folksonomy system. We believe that these ad hoc communities can be tracked over time that can be captured using inter- and intra-transaction mining. This research finds its applicability in the arenas of recommender systems, web search and information retrieval, and other numerous application domains. In cyber-security, this can be used to detect ad hoc groups or communities who have malicious intent on social networking sites such as Facebook and Twitter.

References

Anand, D. and Bharadwaj, K. K. (2013). Pruning trust-distrust network via reliability and risk estimates for quality recommendations, *Social Netw. Analys. Mining* **3**, 1, pp. 65–84.

Arazy, O., Kumar, N. and Shapira, B. (2009). Improving social recommender systems, *IT Professional* **11**, 4, pp. 38–44, doi:10.1109/MITP.2009.76. Available at: http://dx.doi.org/10.1109/MITP.2009.76.

Au Yeung, C.-m., Gibbins, N. and Shadbolt, N. (2009). Contextualising tags in collaborative tagging systems, in *Proceedings of the 20th ACM Conference on Hypertext and hypermedia*, HT '09 (ACM, New York, NY), pp. 251–260, doi:10.1145/1557914.1557958. Available at: http://doi.acm.org/10.1145/1557914.1557958.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R. and Lefebvre, E. (2008). Fast unfolding of communities in large networks.

Brooks, C. H. and Montanez, N. (2006). Improved annotation of the blogosphere via autotagging and hierarchical clustering, in *Proceedings of the 15th International Conference on World Wide Web*, WWW '06 (ACM, New York, NY), pp. 625–632, doi:10.1145/1135777.1135869. Available at: http://doi.acm.org/10.1145/1135777.1135869.

Cha, M., Pérez, J. A. N. and Haddadi, H. (2012). The spread of media content through blogs, *Social Netw. Analys. Mining* **2**, 3, pp. 249–264.

Fortunato, S. (2010). Community detection in graphs, *Phys. Rep.* **486**, 35, pp. 75–174, doi:10.1016/j.physrep.2009.11.002. Available at: http://www.sciencedirect.com/science/article/pii/S0370157309002841.

Gemmell, J., Shepitsen, A., Mobasher, B. and Burke, R. (2008). Personalizing navigation in folksonomies using hierarchical tag clustering, in *Proceedings*

of the 10th International Conference on Data Warehousing and Knowledge Discovery, DaWaK '08 (Springer-Verlag, Berlin, Heidelberg), pp. 196–205, doi:10.1007/978-3-540-85836-2_19. Available at: http://dx.doi.org/10.1007/978-3-540-85836-2_19.

Giannakidou, E., Koutsonikola, V., Vakali, A. and Kompatsiaris, I. (2008). Co-clustering tags and social data sources, in *The Ninth International Conference on Web-Age Information Management, 2008. WAIM '08*, pp. 317–324, doi:10.1109/WAIM.2008.61.

Golder, S. and Huberman, B. A. (2013). The structure of collaborative tagging systems. Available at: http://arxiv.org/abs/cs/0508082.

Gulbahce, N. and Lehmann, S. (2008). The art of community detection, *BioEssays* **30**, 10, pp. 934–938, doi:10.1002/bies.20820.

Hotho, A., Jäschke, R., Schmitz, C. and Stumme, G. (2006). Information retrieval in folksonomies: Search and ranking, in *Proceedings of the 3rd European Conference on The Semantic Web: Research and Applications*, ESWC'06 (Springer-Verlag, Berlin, Heidelberg), pp. 411–426, doi:10.1007/11762256_31. Available at: http://dx.doi.org/10.1007/11762256_31.

Hua, G. and Haughton, D. (2012). A network analysis of an online expertise sharing community, *Social Netw. Analys. Mining* **2**, 4, pp. 291–303.

Kas, M., Carley, K. M. and Carley, L. R. (2012). Trends in science networks: Understanding structures and statistics of scientific networks, *Social Netw. Analys. Mining* **2**, 2, pp. 169–187.

Kashoob, S. and Caverlee, J. (2012). Temporal dynamics of communities in social bookmarking systems, *Social Netw. Analys. Mining* **2**, 4, pp. 387–404.

Li, X., Guo, L. and Zhao, Y. E. (2008). Tag-based social interest discovery, in *Proceedings of the 17th International Conference on World Wide Web*, WWW '08 (ACM, New York, NY), pp. 675–684, doi:10.1145/1367497.1367589.

Liang, H., Xu, Y. and Li, Y. (2010). Mining users' opinions based on item folksonomy and taxonomy for personalized recommender systems, in *2010 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 1128–1135, doi:10.1109/ICDMW.2010.163.

Lindstaedt, S., Mörzinger, R., Sorschag, R., Pammer, V. and Thallinger, G. (2009). Automatic image annotation using visual content and folksonomies, *Multimedia Tools Appl.* **42**, 1, pp. 97–113, doi:10.1007/s11042-008-0247-7. Available at: http://dx.doi.org/10.1007/s11042-008-0247-7.

McPherson, M., Smith-Lovin, L. and Cook, J. M. (2001). Birds of a feather: Homophily in social networks, *Annu. Rev. Sociol.* **27**, 1, pp. 415–444, doi:10.1146/annurev.soc.27.1.415. Available at: http://www.annualreviews.org/doi/abs/10.1146/annurev.soc.27.1.415.

Mika, P. (2007). Ontologies are us: A unified model of social networks and semantics, *Web Semant.* **5**, 1, pp. 5–15, doi:10.1016/j.websem.2006.11.002. Available at: http://dx.doi.org/10.1016/j.websem.2006.11.002.

Missen, M. M. S., Boughanem, M. and Cabanac, G. (2013). Opinion mining: Reviewed from word to document level, *SNAM* **3**, 1, pp. 107–125.

Nair, V. and Dua, S. (2012). Folksonomy-based ad hoc community detection in online social networks, *SNAM* **2**, 4, pp. 305–328.

Newman, M. E. J. (2004). Fast algorithm for detecting community structure in networks, *Phys. Rev. E* **69**, p. 066133, doi:10.1103/PhysRevE.69.066133. Available at: http://link.aps.org/doi/10.1103/PhysRevE.69.066133.

Palla, G., Derényi, I., Farkas, I. and Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society, *Nature* **435**, 7043, pp. 814–818, doi:10.1038/nature03607. Available at: http://dx.doi.org/10.1038/nature03607.

Papadopoulos, S., Kompatsiaris, Y. and Vakali, A. (2010). A graph-based clustering scheme for identifying related tags in folksonomies, in *Proceedings of the 12th International Conference on Data Warehousing and Knowledge Discovery*, DaWaK'10 (Springer-Verlag, Berlin, Heidelberg), pp. 65–76. Available at: http://dl.acm.org/citation.cfm?id=1881923.1881931.

Plangprasopchok, A., Lerman, K. and Getoor, L. (2010). Growing a tree in the forest: Constructing folksonomies by integrating structured metadata, in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10 (ACM, New York, NY), pp. 949–958, doi:10.1145/1835804.1835924. Available at: http://doi.acm.org/ 10.1145/1835804.1835924.

Rees, B. S. and Gallagher, K. B. (2012). Overlapping community detection using a community optimized graph swarm, *SNAM* **2**, 4, pp. 405–417.

Schmitz, C., Schmitz, C., Hotho, A., Jäschke, R. and Stumme, G. (2006). Mining association rules in folksonomies, *Data Science and Classification: Proc. of the 10th IFCS Conf., Studies in Classification, Data Analysis, and Knowledge Organization*, pp. 261–270, doi:10.1.1.93.9741.

seok Min, H., Choi, J., De Neve, W., Ro, Y.-M. and Plataniotis, K. (2009). Semantic annotation of personal video content using an image folksonomy, in *2009 16th IEEE International Conference on Image Processing (ICIP)* pp. 257–260, doi:10.1109/ICIP.2009.5413429.

Wang, X., Tang, L., Gao, H. and Liu, H. (2010). Discovering overlapping groups in social media, in *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10 (IEEE Computer Society, Washington, DC, USA), pp. 569–578, doi:10.1109/ICDM.2010.48. Available at: http://dx.doi.org/10.1109/ICDM.2010.48.

Zhang, L., Wu, X. and Yu, Y. (2006). Emergent Semantics from Folksonomies: A quantitative study, in S. Spaccapietra, K. Aberer and P. Cudré-Mauroux (eds), *Journal on Data Semantics VI*, (Springer-Verlag, Berlin, Heiderberg), pp. 168–186. Available at: http://dl.acm.org/citation.cfm?id=2167832.2167841.

Zhou, D., Bian, J., Zheng, S., Zha, H. and Giles, C. L. (2008). Exploring social annotations for information retrieval, in *Proceedings of the 17th International Conference on World Wide Web*, WWW '08 (ACM, New York, NY), pp. 715–724, doi:10.1145/1367497.1367594. Available at: http://doi.acm.org/10.1145/1367497.1367594.

Chapter 5

Several Approaches for Detecting Anomalies in Network Traffic Data

Céline Lévy-Leduc

AgroParisTech/INRA MIA UMR 518
16, Rue Claude Bernard, 75005 Paris, France
celine.levy-leduc@agaroparistech.fr

In this chapter we propose centralised and decentralised approaches for detecting changepoints in high-dimensional network traffic data. These methods consist of a data-reduction stage followed by a nonparametric changepoint detection test based on rank statistics and adapted to deal with the presence of censored data. The advantage of such a nonparametric approach is that it does not require any prior information on the distribution of the observations. Both types of methodologies can address massive data and perform network anomaly detection on the fly. The performance of these methods are investigated on real Internet traffic data provided by a major French Internet service provider.

5.1. Introduction

Since the recent attacks against major web services providers, which led to a disruption of services to users, network anomaly detection has become a major concern to the network security community. In this chapter, we shall focus on denial of service (DoS) attacks and distributed denial of service (DDoS) attacks which are typical examples of network anomalies.

Several methods for dealing with DoS and DDoS attacks have been proposed. They can be split into two categories: signature-based approaches and statistical methods. The former operate by comparing the observed patterns of network traffic with known attack templates: Roesch (1999) and Paxson (1999) propose two examples of such network anomalies detection systems. Thus, this type of methodology can only be applied to the detection of anomalies that have already been observed. The second type of methodology relies on statistical approaches and can thus potentially detect any type of network anomalies which do not have to belong to a prescribed

database. The statistical approaches consist of modelling network anomalies by abrupt changes in some network characteristics occurring at unknown time instants. Hence, the network anomaly detection issue can be seen as a changepoint detection problem which is a familiar topic in statistics, see, for instance, Basseville and Nikiforov (1993); Brodsky and Darkhovsky (1993); Csörgő and Horváth (1997).

In the changepoint detection field, two different approaches are usually distinguished: the detection can be retrospective and hence with a fixed delay (batch approach) or online, with a minimal average delay (sequential approach). A widely used changepoint detection technique in the network security field is the cumulated sum (CUSUM) algorithm which was first proposed by Page (1954) and which is a sequential approach. It has, for instance, been used by Wang *et al.* (2002) and by Siris and Papagalou (2006) for detecting DoS attacks of the TCP (Transmission Control Protocol) SYN (synchronization) flooding type. This attack consists of exploiting the TCP three-way hand-shake mechanism and its limitation in maintaining half-open connections. More precisely, when a server receives a SYN packet, it returns a SYN/ACK (synchronization acknowledged) packet to the client. Until the SYN/ACK packet is acknowledged by the client, the connection remains half-opened for a period of at most the TCP connection timeout. A backlog queue is built up in the system memory of the server to maintain all half-open connections, thus leading to a saturation of the server. In Siris and Papagalou (2006), the authors use the CUSUM algorithm to detect a changepoint in the time series corresponding to the aggregation of the SYN packets received by all the requested destination Internet protocol (IP) addresses. With such an approach, it is only possible to set off an alarm when a massive change occurs in the aggregated series. However, it is impossible to identify the attacked IP addresses.

In order to identify the attacked IP addresses, it is possible to apply a changepoint detection test to each time series corresponding to the number of TCP/SYN packets received by each destination IP address. This idea is used in Tartakovsky *et al.* (2006a) and Tartakovsky *et al.* (2006b) where a multichannel detection procedure is proposed: it makes it possible to detect changes which occur in a channel and which could be obscured by the normal traffic in the other channels if global statistics were used.

When analysing wide-area-network traffic, however, it is no longer possible to consider individually all the possible destination IP addresses for computational reasons since the quantity of data is too massive. For instance, the real data used for the evaluation of the proposed methods in

Sections 5.3 and 5.4 contain several thousands of different IP addresses in each one-minute observation window. In order to detect anomalies in such massive data within a reasonable time span, dimension reduction techniques have to be used. Several approaches have been proposed. The first one uses principal component analysis (PCA) techniques; see Lakhina *et al.* (2004). The second one uses random aggregation (or sketches); see Krishnamurthy *et al.* (2003) and Li *et al.* (2006). The identification of the involved attacked IP addresses is possible with the second approach but not with the first one.

In the approaches mentioned above, all the data are sent to a central analysis site, called the collector in the sequel, in which a decision is made concerning the presence of an anomaly. These methods are called centralised approaches. In this chapter, we present a method called *TopRank* for detecting changepoints in a multivariate time series under computational constraints which makes it possible to process the data on the fly. It is based on record filtering which is another dimension-reduction technique. With this method, network anomalies of the following type can be identified: TCP/SYN flooding, UDP (User Data Protocol) flooding, PortScan and NetScan. UDP flooding is an attack similar to TCP/SYN flooding which aims at saturating the memory of a destination IP address by sending a lot of UDP packets. A PortScan consists of sending TCP packets to each port of a machine to know which ones are open. In a NetScan attack, a source IP address sends packets to a large number of IP addresses in order to detect the machines which are connected to the network. The *TopRank* methodology is described in Section 5.2.1.1. It consists of a data-reduction stage based on record filtering followed by a nonparametric changepoint test based on rank statistics and adapted to the presence of censored data.

A limitation of these centralised approaches is that they are not adapted to large networks with massive data since, in this case, the communication overhead within the network becomes significant. In this situation, decentralised approaches are often preferred to centralised ones. In this chapter, we present some decentralised approaches which consist of processing the data within the network (in local monitors) in order to send only the most relevant data to the collector. In Huang *et al.* (2007), a method to decentralise the approach of Lakhina *et al.* (2004) is considered but, as previously explained, localising the network anomaly is impossible with such a method. In Section 5.2.2.1, we present an efficient way of decentralising the *TopRank* algorithm described in Section 5.2.1.1 and called *DTopRank* (Distributed *TopRank*) in the sequel. It consists of applying the *TopRank* algorithm locally in each monitor and sending only the most relevant data to the

collector. The data sent by the different local monitors are then aggregated in a specific way and a nonparametric rank test for doubly censored data is performed within the collector. The *DTopRank* algorithms make it possible to achieve a performance comparable with the fully centralised *TopRank* algorithm while minimising the quantity of data that needs to be sent to the collector. In Section 5.2.2.3, we propose another decentralised approach called *MultiRank* using a novel nonparametric rank-based changepoint detection test for multivariate data.

The chapter is organised as follows. In Section 5.2, we describe the centralised and decentralised approaches. The performance of the proposed algorithms are then assessed on real traffic data provided by a major French Internet service provider (ISP) in Sections 5.3 and 5.4.

5.2. Description of the Methods

In the sequel, the raw data consists of Netflow-type data collected at several points of the Internet network. The data at our disposal includes the source and destination IP addresses, the source and destination ports, the start time and the end time of the flow as well as the protocol and the number of exchanged packets.

Depending on the type of attack that one wants to detect, some time-indexed traffic characteristics are of particular interest and have to be processed for detection purposes. For instance, in the case of the TCP/SYN flooding, the quantity of interest is the number of TCP/SYN packets received by each destination IP address per unit of time.

We denote by Δ the smallest time unit used for building time series from our raw Netflow type data. The time series are built as follows: in the case of TCP/SYN flooding, we shall denote by $N_i^{\Delta}(t)$ the number of TCP/SYN packets received by the destination IP address i in the sub-interval indexed by t of size Δ seconds. The corresponding time series of the destination IP address i will thus be denoted by $(N_i^{\Delta}(t))_{t\geq 1}$. In the case of UDP flooding, $N_i^{\Delta}(t)$ will correspond to the number of UDP packets received by the destination IP address i in the tth sub-interval of size Δ seconds. For a PortScan, $N_i^{\Delta}(t)$ will be the number of different requested destination ports of the destination IP address i in the tth sub-interval of size Δ seconds and for a NetScan, it will be the number of different requested destination IP addresses by the source IP address i.

Our goal is now to provide algorithms for detecting changepoints in the time series $(N_i^{\Delta}(t))_{t\geq 1}$ for each $i \in \{1, \ldots, D\}$ under the following

computational constraint: being able to process the data on the fly, even for a high dimension D. For instance, in the case of TCP/SYN flooding, D is the number of destination IP addresses appearing in the raw data and can be huge, up to several thousand addresses within a minute and several million within a few days.

In the following we will drop the superscript Δ in the notation $N_i^{\Delta}(t)$ for notational simplicity.

5.2.1. *Centralised approaches*

In this chapter, we shall focus only on batch methods which means that the traffic is analysed in successive observation windows, each having a duration of $P \times \Delta$ seconds, for some fixed integer P. A decision concerning the presence of potentially attacked IP addresses is made at the end of each observation window and we also identify the involved IP addresses. The value of D then corresponds to the number of different i encountered in the observation window of time length $P \times \Delta$ seconds.

A crude solution for detecting changepoints in the time series $(N_i(t))_{1 \leq t \leq P}$ would be to apply a changepoint detection test to each time series $(N_i(t))_{1 \leq t \leq P}$ for all $i \in \{1, \ldots, D\}$. Since D can be huge even in a given observation window, we are faced in practice with massive data streams leading to the construction and the analysis of several thousands of time series even for short observation periods (around one minute). To overcome this difficulty, a data-reduction stage must precede the changepoint detection step.

Two different data-reduction mechanisms can be considered: record filtering and random aggregation (sketches). The record filtering consists of selecting the heavy-hitters among the flows involved and processing them while the random aggregation will construct and process several (randomly chosen) linear combinations of all the flows. Random aggregation is currently the dimension-reduction mechanism which is the most used in the network security community. Nevertheless, we believe that for the purpose of changepoint detection, in particular if the changepoints involve a massive increase, record filtering would be a more efficient alternative. At first glance, random aggregation has the advantage of processing all the data. However, heavy-hitters are mixed with many other flows, which may smooth the changepoints and result in poor detection. On the other hand, heavy-hitters are selected with high probability in record filtering and their changepoints are more likely to be detected. In the sequel, we shall focus

on the record-filtering approach. For a further comparison of these two approaches we refer the reader to Lévy-Leduc and Roueff (2009).

As for the changepoint detection step, we propose using nonparametric tests based on ranks which do not require any prior information concerning the distribution of the time series constructed after the dimension-reduction step. Such an approach is of particular interest for dealing with Internet traffic data, for which there is a lack of commonly accepted parametric models.

In the following, we shall refer to record filtering followed by a nonparametric changepoint detection test as the *TopRank* algorithm.

5.2.1.1. *TopRank*

The *TopRank* algorithm can be split into three steps described hereafter. Note that the following processing is performed in each observation window of length $P \times \Delta$ seconds and that all the stored data is erased at the end of each observation window.

1. Record filtering: For each time index t in $\{1, \ldots, P\}$, the indices of the M largest counts $N_i(t)$ are recorded and labelled as $i_1(t), \ldots, i_M(t)$ to ensure that $N_{i_1(t)}(t) \geq N_{i_2(t)}(t) \geq \cdots \geq N_{i_M(t)}(t)$. In the sequel, $\mathcal{T}_M(t)$ denotes the set $\{i_1(t), \ldots, i_M(t)\}$. We stress that, in order to perform the following steps, we only need to store the variables $\{N_i(t),\, i \in \mathcal{T}_M(t),\, t = 1, \ldots, P\}$.

2. Creation of censored time series: For each index i selected in the previous step ($i \in \bigcup_{t=1}^{P} \mathcal{T}_M(t)$), the censored time series is built. This time series is censored since i does not necessarily belong to the set $\mathcal{T}_M(t)$ for all indices t in the observation window, in which case, its value $N_i(t)$ is not available and is censored using the upper bound $N_{i_M(t)}(t) = \min_{i \in \mathcal{T}_M(t)} N_i(t)$. More formally, the censored time series $(X_i(t), \delta_i(t))_{1 \leq t \leq P}$ are defined, for each $t \in \{1, \ldots, P\}$, by

$$(X_i(t), \delta_i(t)) = \begin{cases} (N_i(t), 1), & \text{if } i \in \mathcal{T}_M(t) \\ \left(\min\limits_{j \in \mathcal{T}_M(t)} N_j(t), 0 \right), & \text{otherwise.} \end{cases}$$

The value of $\delta_i(t)$ indicates whether the corresponding value $X_i(t)$ has been censored or not. Observe that, by definition, $\delta_i(t) = 1$ implies that $X_i(t) = N_i(t)$ and $\delta_i(t) = 0$ implies that $X_i(t) \geq N_i(t)$. We also define the

upper and lower bounds of $X_i(t)$ by $\overline{X}_i(t) = X_i(t)$ and $\underline{X}_i(t) = X_i(t)\delta_i(t)$, respectively.

In practice the censored time series are only built for indices i selected in the first step, i.e. $i \in \bigcup_{t=1}^{P} \mathcal{T}_M(t)$. However, many such time series will be highly censored. Hence we propose an additional dimension reduction. At this stage, two strategies for dimension reduction can be considered. The first one consists of considering only the indices

$$i \in \bigcup_{t=1}^{P} \mathcal{T}_{M'}(t),$$

where $M' \in \{1, \ldots, M\}$ is a chosen parameter. The second one consists of processing a fixed number S of time series instead of all those in $\bigcup_{t=1}^{P} \mathcal{T}_M(t)$ (at most $M \times P$) and to build only the time series corresponding to the index i in the list $i_1(1), \ldots, i_1(P), i_2(1), \ldots, i_2(P), i_3(1), \ldots$ where the indices $i_k(t)$ are defined in the previous step. The main difference between these two strategies is that for the second one the number of time series to analyse is fixed and equal to S.

In the first two steps we have significantly reduced the amount of data to be processed. In the next step, a changepoint detection test is applied to the new time series corresponding to the selected IP addresses.

3. Changepoint detection test: The nonparametric changepoint test, described hereafter, is applied to each time series created in the previous stage and the corresponding p-value is computed, a small value suggesting a potential anomaly.

Let us now further describe the statistical test that we perform. This procedure aims at testing from the observations previously built $(\underline{X}_i(t), \overline{X}_i(t))_{1 \le t \le P}$ if a change occurred in this time series for a given i. More precisely, if we drop the subscript i for notational simplicity in the description of the test, the tested hypotheses are:

(H_0): "$(\underline{X}(t), \overline{X}(t))_{1 \le t \le P}$ are i.i.d. random vectors."

against

(H_1): "There exists some r such that $((\underline{X}(1), \overline{X}(1)), \ldots, (\underline{X}(r), \overline{X}(r)))$ and $((\underline{X}(r+1), \overline{X}(r+1)), \ldots, (\underline{X}(P), \overline{X}(P)))$ have a different distribution."

To define the proposed test statistic, we define, for each s, t in $\{1, \ldots, P\}$,

$$h(s,t) = \mathbb{1}(\underline{X}(s) > \overline{X}(t)) - \mathbb{1}(\overline{X}(s) < \underline{X}(t)),$$

where $\mathbb{1}(E) = 1$ if the event E is true and 0 if it is not, and

$$Y_s = \frac{U_s}{\sqrt{\sum_{t=1}^{P} U_t^2}}, \quad \text{with} \quad U_s = \sum_{v=1}^{P} h(s,v). \tag{5.1}$$

The test statistic is then given by

$$W_P = \max_{1 \le t \le P} \left| \sum_{s=1}^{t} Y_s \right|. \tag{5.2}$$

The following theorem, which is proved in Lung-Yut-Fong *et al.* (2012a), provides, under mild assumptions, the limiting distribution of W_P, as P tends to infinity, under the null hypothesis and thus provides a way of computing the p-values of the test.

Theorem 5.1. *Let $(\underline{X}, \overline{X})$ be a $\mathbb{R}^2$-valued random vector such that*

$$\mathbb{P}(F(\underline{X}^-) + G(\overline{X}) = 1) < 1, \tag{5.3}$$

where F is the cumulative distribution function (c.d.f.) of $\overline{X}$, G the c.d.f. of $\underline{X}$ and $F(x^-)$ denotes the left limit of F at point x. Let $(\underline{X}(t), \overline{X}(t))_{1 \le t \le P}$ be i.i.d. random vectors having the same distribution as $(\underline{X}, \overline{X})$, then, as P tends to infinity,

$$\sup_{0<u<1} \left| \sum_{s=1}^{\lfloor Pu \rfloor} Y_s \right| \xrightarrow{d} B^\star := \sup_{0<u<1} |B(u)|, \tag{5.4}$$

where $\{B(u), 0 < u < 1\}$ denotes the Brownian Bridge and $\xrightarrow{d}$ denotes the convergence in distribution.

Theorem 5.1 of this chapter thus extends Theorem 1 of Gombay and Liu (2000), where only one-sided censoring was considered and continuity of the random variables was assumed.

Remark 5.1. Theorem 5.1 provides a way of controlling the asymptotic false alarm rate, for large enough P. The only requirement of Theorem 5.1 is (5.3). In particular, if the random variables $\underline{X}$ and $\overline{X}$ both have a continuous c.d.f., (5.3) holds whenever $\mathbb{P}(\underline{X} = \overline{X}) > 0$, that is, when the probability of not being censored is positive. Thus, the p-values deduced from Theorem 5.1 are reliable whenever P is large enough and $\mathbb{P}(\underline{X} = \overline{X}) > 0$ that is if some non-censored values have been observed.

Based on (5.4), we take for the changepoint detection test the following p-value: $Pval(W_P)$, where for all positive b (see, for instance, Billingsley, 1968, p. 85),

$$Pval(b) = \mathbb{P}(B^\star > b) = 2\sum_{j=1}^{\infty}(-1)^{j-1}e^{-2j^2b^2}, \tag{5.5}$$

where $B^\star$ is defined in (5.4).

5.2.2. *Decentralised approaches*

The centralised *TopRank* algorithm analyses the global packet counts. In the case of decentralised approaches, we have at our disposal a monitoring system with a set of local monitors $M_1, \ldots, M_K$, which collect and analyse the locally observed time series. As a consequence of decentralised processing, the packets sent to a given destination IP address are not observed by all monitors, although some overlap may exist, depending on the routing matrix and the location of the monitors.

For TCP/SYN flooding detection purposes, we shall focus on the number of TCP/SYN packets and thus denote by $N_i^k(t)$ the number of TCP/SYN packets transiting to the destination IP address i in the subinterval indexed by t, observed by the kth monitor. In the proposed batch approach, detection is performed from the data observed during an observation window of duration $P \times \Delta$ seconds. The goal is to detect changepoints in the aggregated time series $(N_i(t))_{t\geq 1}$ corresponding to the total number of packets received by the IP address i using only the local time series $(N_i^k(t))_{t\geq 1}$ for each $k \in \{1, \ldots, K\}$ and a quantity of data transmitted to the collector that is as small as possible.

5.2.2.1. *DTopRank*

The *DTopRank* algorithm operates at two distinct levels: the local processing step within the local monitors $M_1, \ldots, M_K$, and the aggregation and global changepoint detection step within the collector.

1. Local processing: The local processing of *DTopRank* consists of four steps which are applied in each of the K monitors. The first three steps are similar to the *TopRank* algorithm applied to the local series of counts $(N_i^k(t))_{1\leq t\leq P}$ for each k in $\{1, \ldots, K\}$.

The fourth step consists of selecting the data to be transmitted to the collector. We select in each monitor M_k the d censored time series having the smallest p-values and send them to the collector. Thus, the collector receives at most $d \times K$ censored time series, instead of $\sum_{k=1}^{K} D_k$, where D_k is the number of destination IP addresses seen by the kth monitor, if a centralised approach was used.

2. Aggregation and changepoint detection test in the collector: Within the collector, the lower and upper bounds of the aggregated time series $(\underline{Z}_i(t), \overline{Z}_i(t))_{1\leq t\leq P}$ associated to the IP address i are then built as follows:

$$\underline{Z}_i(t) = \sum_{k\in\mathcal{K}} \underline{X}_i^{(k)}(t) \quad \text{and} \quad \overline{Z}_i(t) = \sum_{k\in\mathcal{K}} \overline{X}_i^{(k)}(t), \tag{5.6}$$

where $(\underline{X}_i^{(k)}(t),\ t = 1,\dots,P)$ and $(\overline{X}_i^{(k)}(t),\ t = 1,\dots,P)$ are the time series associated to the IP address i computed by the monitor M_k, and $\mathcal{K}$ is the set of monitors which have transmitted series associated to the IP address i. Then, the changepoint detection test described in step 3 of *TopRank* algorithm is applied to the time series $(\underline{Z}_i(t),\ t = 1,\dots,P)$ and $(\overline{Z}_i(t),\ t = 1,\dots,P)$. An IP address i is thus claimed to be attacked at a given false alarm rate $\alpha \in (0,1)$, if $Pval(W_P) < \alpha$.

As noted in Remark 5.1, Theorem 5.1 can be applied to the aggregated time series $(\underline{Z}_i(t), \overline{Z}_i(t))$ as long as they are not fully censored. By definition, the aggregated series is more censored than the individual series $(\underline{X}_i^{(k)}(t), \overline{X}_i^{(k)}(t))$ detected at monitor level. On the other hand, for a given address i, only the series that are among the d series having the smallest p-values at the monitor level are aggregated at the collector level. Hence, the collector usually aggregates strictly less than K series and only aggregates potentially significant series.

5.2.2.2. *BTopRank*

In the sequel, the *DTopRank* algorithm is compared with a simpler approach using, instead of the aggregation step, a simple Bonferroni correction of the p-values determined in each monitor. More precisely, in *BTopRank* an IP address is claimed to be attacked at the level α in $(0,1)$ within the collector if at least one local monitor has computed a p-value smaller than α/K, namely if $K(\inf_{1\leq k\leq K} \mathrm{Pval}_k) < \alpha$, Pval_k being the p-value computed in the monitor k.

5.2.2.3. *MultiRank*

Like the *DTopRank* approach, the *MultiRank* procedure also operates at two levels: the local processing step within the local monitors $M_1, \ldots, M_K$ and the global changepoint detection step within the collector.

Within each local monitor M_k and for each IP address i, the time series corresponding to the number of received packets $(N_i^k(t))_{1 \leq t \leq P}$ is built. For each IP address i, a changepoint detection test in the multivariate time series $(N_i^k(t))_{1 \leq t \leq P, 1 \leq k \leq K}$ has to be performed within the collector in order to make a decision concerning a network anomaly.

For dealing with this issue, we shall consider the following more general problem. Let P multivariate K-dimensional observations $(\mathbf{X}_1, \ldots, \mathbf{X}_P)$ and denote by $X_{i,k}$ the kth coordinate of X_i, such that $\mathbf{X}_i = (X_{i,1}, \ldots, X_{i,K})'$, where the prime is used to denote transposition. We aim at testing:

(H_0): "$(\mathbf{X}_1, \ldots, \mathbf{X}_P)$ are i.i.d random vectors."

against

(H_1): "There exists n_1 such that $(\mathbf{X}_1, \ldots, \mathbf{X}_{n_1})$ and $(\mathbf{X}_{n_1+1}, \ldots, \mathbf{X}_P)$ have different distributions."

For this, let $\mathbf{V}_P(n_1) = (V_{P,1}(n_1), \ldots, V_{P,K}(n_1))'$ denote the vector such that

$$V_{P,k}(n_1) = \frac{1}{P^{3/2}} \sum_{i=1}^{n_1} \sum_{j=n_1+1}^{P} \{\mathbb{1}(X_{i,k} \leq X_{j,k}) - \mathbb{1}(X_{j,k} \leq X_{i,k})\}, \quad k = 1, \ldots, K, \quad (5.7)$$

and define

$$\tilde{S}_P(n_1) = \mathbf{V}_P(n_1)' \hat{\Sigma}_P^{-1} \mathbf{V}_P(n_1), \quad (5.8)$$

where $\hat{\Sigma}_P$ denote the K-dimensional empirical covariance matrix defined by

$$\hat{\Sigma}_{P,kk'} = \frac{4}{P} \sum_{i=1}^{P} \{\hat{F}_{P,k}(X_{i,k}) - 1/2\}\{\hat{F}_{P,k'}(X_{i,k'}) - 1/2\}, \quad 1 \leq k, k' \leq K.$$

We now consider the statistic

$$\tilde{W}_P = \max_{1 \leq n_1 \leq P-1} \tilde{S}_P(n_1). \quad (5.9)$$

The following theorem, proved in Lung-Yut-Fong *et al.* (2012b), gives the asymptotic p-values of $\tilde{W}_P$ under the null hypothesis (H_0).

Theorem 5.2. *Assume that* $(\mathbf{X}_i)_{1\leq i\leq P}$ *are* $\mathbb{R}^K$*-valued i.i.d. random vectors such that, for all* k*, the c.d.f.* F_k *of* $X_{1,k}$ *is a continuous function. Further assume that the* $K\times K$ *matrix* Σ *defined by*

$$\Sigma_{kk'} = 4\,\mathrm{Cov}(F_k(X_{1,k}); F_{k'}(X_{1,k'})), \quad 1\leq k,k'\leq K,$$

is invertible. Then,

$$\tilde{W}_P \xrightarrow{d} \sup_{0<t<1}\left(\sum_{k=1}^{K} B_k^2(t)\right), \quad \text{as } P\to\infty, \tag{5.10}$$

where d *denotes convergence in distribution and* $\{B_k(t),\ t\in(0,1)\}_{1\leq k\leq K}$ *are independent Brownian bridges.*

To determine the p-value $P_{\mathrm{val}}(W_P)$ associated to (5.10), one can use the following result due to Kiefer (1959):

$$\begin{aligned} P_{\mathrm{val}}(b) &= \mathbb{P}\left(\sup_{0<t<1}\left(\sum_{k=1}^{K} B_k^2(t)\right) > b\right) \\ &= 1 - \frac{4}{\Gamma\left(\frac{K}{2}\right) 2^{\frac{K}{2}} b^{\frac{K}{2}}} \sum_{m=1}^{\infty} \frac{(\gamma_{(K-2)/2,m})^{K-2}\exp[-(\gamma_{(K-2)/2,m})^2]/2b}{[J_{K/2}(\gamma_{(K-2)/2,m})]^2}, \end{aligned} \tag{5.11}$$

where J_ν is the Bessel function of the first kind, $\gamma_{\nu,m}$ is the mth non-negative zero of J_ν and Γ is the Gamma function. In practice, only a few terms of the series have to be computed. For values of K of 40 or less computing the p-values from the 30 terms of the series was sufficient.

In order to reduce the quantity of information that is transmitted to the collector, a possibility would be to filter the information within the local monitors by using the *TopRank* as it is done in the *DTopRank* approach. The quantities $V_{P,k}$ and $\hat{\Sigma}_P$ should then be changed in order to take into account the presence of censored observations. This will be the subject of future work.

5.3. Application of the Centralised Approaches to Real Data

In this section, we give the results of the *TopRank* algorithm when it is applied to some real Internet traffic provided by France-Télécom within the framework of the ANR-RNRT OSCAR project and we give some hints about the choice of the different parameters involved.

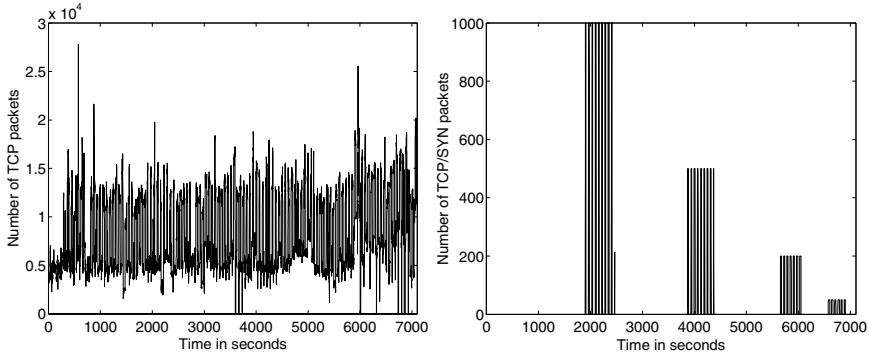


Fig. 5.1. Number of TCP packets exchanged and number of TCP/SYN packets received by the four attacked IP addresses.

This data corresponds to a recording of 118 minutes of ADSL (asymmetric digital subscriber line) and peer-to-peer (P2P) traffic to which some TCP/SYN flooding-type attacks have been added. Figure 5.1 (left) displays the total number of TCP packets received each second by the different requested IP addresses. The number of TCP/SYN packets received by the four attacked destination IP addresses are displayed on the right in Figure 5.1. As we can see in this figure, the first attack occurs at around 2000 seconds, the second at around 4000 seconds, the third at around 6000 seconds and the last one at around 6500 seconds.

From Figure 5.1, we can see that we are faced with massive data and that the attacks are completely hidden in TCP traffic and thus very difficult to detect.

5.3.1. *Choice of parameters*

As described above, Step 1 (the record filtering step) and Step 2 (the creation of the censored time series) of the *TopRank* algorithm rely on several parameters, (P, Δ, M for Step 1 and S or M' for Step 2). As for Step 3 (the changepoint detection test), it does not require the tuning of any parameters since it is a nonparametric approach. The main objective of the first two steps is to cope with high dimensionality and the requirements of real-time implementation. The choice of the parameters must satisfy these requirements as well as a relevant selection of the time series to be processed in the subsequent detection step. In the following, we give some guidance regarding parameter selection in the context of network anomaly detection.

Since a decision concerning the presence of attacks is made at the end of each observation window, the maximal detection delay is given by the time length of the observation window, that is: $P \times \Delta$ seconds. Once the time length of the observation window has been chosen, one should choose P as large as possible under the constraints of real-time data processing. Indeed, a large P ensures a better statistical consistency. On the other hand, the test statistic W_P in (5.2) has a $O(P^2)$ computational complexity. Given the computational limits of a standard computer, we chose $P = 60$ in order to allow up to 10^3 time series to be processed within a one-minute observation window.

Let us now explain the choice of M, which sets the censorship level. Indeed, the number of TCP/SYN packets received by the Mth most-requested machine corresponds to a threshold above which an IP address may appear as potentially attacked. From the data, we remark that 99% of the observed values of this threshold are at most 10 when $M = 10$ and at most 5 for $M = 20$. In the applications to real data, we chose $M = 10$ to allow us to capture flows with significantly high traffic rates (10 packets per second) while ensuring a low cost in terms of memory storage. Recall that a $M \times P$ data table $(\underline{X}_i(t), \overline{X}_i(t))_{1 \le t \le P, 1 \le i \le M}$ has to be stored during the first and second step.

We now comment on the choice of the parameter M' in $\{1, \ldots, M\}$. Taking $M' = 1$ means that we only analyse the IP addresses i having, at least once in an observation window, the largest $N_i(t)$. Taking $M' = M$ means that the detection step is applied to the censored time series of all IP addresses i which have been selected in the filtering step. Hence increasing M' increases the number of analysed censored time series. The right part of Figure 5.2 displays the number of time series which are actually built in Step 2 of *TopRank* after the filtering stage of Step 1 for different values of M': $M' = 1$, $M' = 5$ and $M' = M = 10$ when we are looking for TCP/SYN flooding-type attacks. The left part of Figure 5.2 displays the number of different destination IP addresses every minute of the traffic trace. Comparing the left part with the right part of Figure 5.2, we see that Step 1 and Step 2 of the *TopRank* algorithm appear to be necessary to provide an implementation feasible on the fly. Indeed, applying Step 3 to each IP address every minute would produce an excessive computational load. As for the statistical performance of the method with respect to the parameter M', we shall see in the following section that the parameter M' does not significantly change the results in terms of false alarm and detection rates.

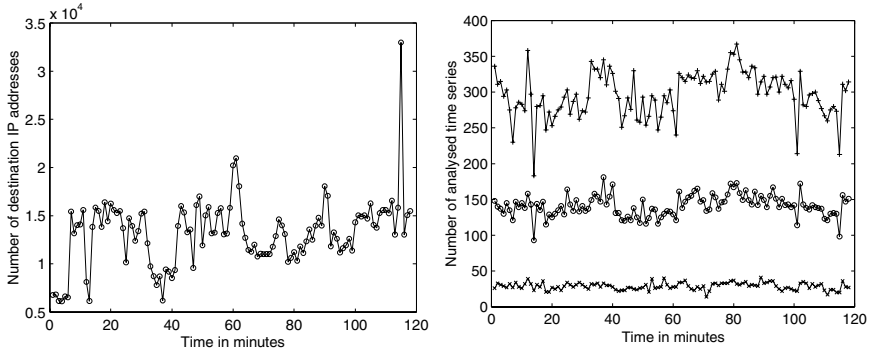


Fig. 5.2. Left: Number of destination IP addresses every minute. Right: Number of analysed time series every minute when $M' = 1$ ("x"), $M' = 5$ ("o") and $M' = M = 10$ ("+").

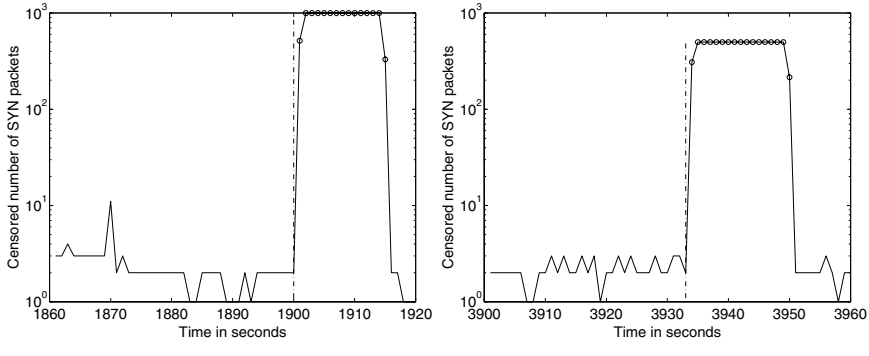


Fig. 5.3. Censored time series of the four attacked IP addresses where the vertical lines correspond to the detected changepoint instants and the uncensored values are displayed with ("o").

5.3.2. *Performance of the method*

We now investigate the performance of the *TopRank* algorithm with the following parameters: $P = 60$, $\Delta = 1$s, $M = 10$ and $M' = 1$.

5.3.2.1. *Statistical performance*

First, note that with the previous choice of parameters the attacked IP addresses have been identified when the upper bound of the p-value α introduced in Step 3 of *TopRank* is such that $\alpha \geq 2 \times 10^{-6}$.

Figure 5.3 displays the censored time series (Step 2 of *TopRank*) of two of the four attacked IP addresses. These censored time series are displayed

Table 5.1. Statistical performance for detecting the four successive SYN flooding attacks displayed on the right-hand side of Figure 5.1. SP is the method devised by Siris and Papagalou (2006), the h row gives the smallest threshold values that ensure the detection of each attack. In the last row, the corresponding number of false alarms is displayed.

	Number of SYN packets	1000	500	200	50
	h	5	6.5	9.7	16.34
SP	Number of false alarms	69	65	62	22

in the first observation window in which the algorithm detects the anomaly. We also display with a vertical line the instant where the change is detected. Remember that the uncensored time series of the four attacked IP addresses are displayed in Figure 5.1 (right). We observe that we recover the real number of received TCP/SYN packets received per second: 1000 for the first IP address and 500 for the second one.

The *TopRank* part of Table 5.1 gives the smallest p-value above which the corresponding attack is detected as well as the number of false alarms. The number of false alarms corresponds to the number of IP addresses for which an alarm is triggered but which are different from the attacked IP addresses. For instance, the first attack is detected if $\alpha \geq 10^{-8}$, see (5.5), and the associated number of false alarms is equal to 3. If $M' = 5$ or $M' = M = 10$, the results remain unchanged except for the third attack for which the number of false alarms equals 10 instead of 9.

In Table 5.1, we also give the results obtained from the same data with a method proposed by Siris and Papagalou (2006). This algorithm uses the CUSUM algorithm to look for a changepoint in the time series corresponding to the sum of received SYN packets by all the destination IP addresses which have been requested. For each observation window of 60 seconds, an alarm is set off when the statistic g_n defined in equation (6) of Siris and Papagalou (2006) is greater than a threshold h at least once in the window. This quantity g_n depends on two parameters α and β. We use the same values as Siris and Papagalou (2006), namely $\alpha = 0.5$ and $\beta = 0.98$, to obtain the results displayed in Table 5.1. We observe that the *TopRank* algorithm allows us not only to retrieve the attacked destination IP addresses but also seems to perform better in terms of false alarm rate. This suggests that aggregating traffic flows results in a poor detection of malicious flows, especially when the normal traffic is high.

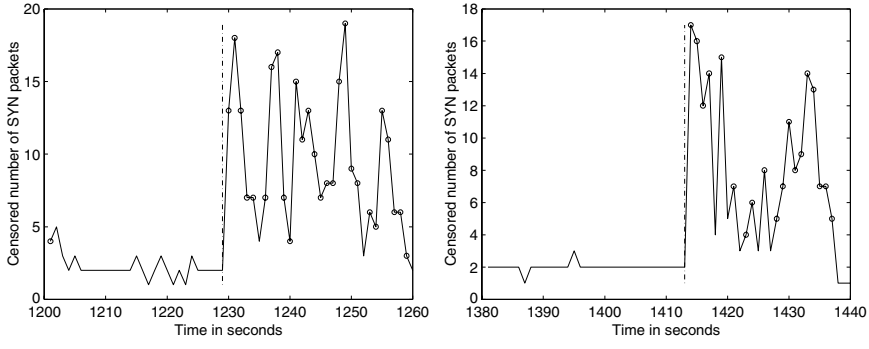


Fig. 5.4. Censored time series for IP addresses considered as false alarms where the vertical lines correspond to the detected changepoint instants and the uncensored values are displayed with ("o").

Indeed, a close look shows that the normal traffic is particularly high during the first attack, which explains why the corresponding threshold value is the lowest (and the false alarm rate the highest) although this attack is the most intense.

To compute the number of false alarms, we have considered that the attacked IP addresses were only those for which an attack was generated but it is possible that the underlying ADSL and P2P traffic contains some other attacks. Figure 5.4 displays the censored time series of some IP addresses which were considered to be false alarms in the *TopRank* algorithm as well as the time instant where a change was detected (vertical line). However, if we refer to their time series, these IP addresses could be considered as being attacked. Thus, the results have been computed in the most unfavourable way for our methodology.

5.3.2.2. *Numerical performance*

As we have seen, the *TopRank* algorithm seems to give satisfactory results from a statistical point of view. Moreover, with $M = 10$, $M' = 1$ and $P = 60$, applying the *TopRank* algorithm takes only one minute and 19 seconds to process the whole traffic trace of 118 minutes, when looking for TCP/SYN flooding type-attacks with a computer having the following configuration: RAM 1GB, CPU 3GHz. For an application of the *TopRank* approach to a data set containing anomalies of UDP flooding, PortScan and NetScan, we refer the reader to Lévy-Leduc and Roueff (2009).

5.4. Application of the Decentralised Approaches

We consider the same data as those used in Section 5.3. As this data set does not contain full routing information, it has been artificially distributed over a set of virtual monitors as follows: the data is shared among $K = 15$ monitors by assigning each source destination pair (source IP address, destination IP address) to a randomly chosen monitor; a single monitor thus records all the flows between two particular IP addresses. The experiments reported below are based on 50 independent replications of this process. Finally, the existing anomalies have been down-sampled (by randomly dropping packets involved in the attacks) to 12.5 and 25 packets/s, respectively, to explore more difficult detection scenarios.

Figures 5.5 (a), (b) display the number of TCP/SYN packets globally exchanged within two different monitors whereas (c), (d) focus on the traffic received by the attacked IP addresses within these two monitors.

The attacked IP addresses (bottom part of Figure 5.5) are completely hidden in the global TCP/SYN traffic (top part of Figure 5.5) and thus very difficult to detect. Note also that 1,006,000 destination IP addresses are present in this data set, with an average of 15,000 destination IP addresses in each of the 118 one-minute observation windows. Hence, real-time processing of the data would not be possible, even at the monitor level, without a dimension-reduction step such as record filtering.

5.4.1. *Performance of the methods*

In what follows, the *DTopRank* algorithm is used with the same parameters as those adopted in Section 5.3 for the *TopRank* algorithm, with one-minute windows divided in $P = 60$ sub-intervals of $\Delta = 1\,\mathrm{s}$, with $M = 10$ and $S = 60$. The parameter d was set to $d = 1$, due to the limited number of attacks expected in each one-minute window. In setting P and Δ the main concern is the overall observation duration $\Delta \times P$ which should be sufficient to allow for meaningful statistical decisions while ensuring an acceptable detection delay and that the extracted series can still be considered as stationary in the absence of change.

Figure 5.6 shows the benefits of the aggregation stage within the collector of the *DTopRank* algorithm with respect to the use of a local strategy or the simple Bonferroni correction in the *BTopRank* algorithm. Figures 5.6 (a), (b) and (c) display the time series $(\underline{X}(t),\ t = 1, \ldots, P)$ and $(\overline{X}(t),\ t = 1, \ldots, P)$ associated to an attacked IP address in three different monitors as well as the corresponding p-values. Figure 5.6 (d) displays the

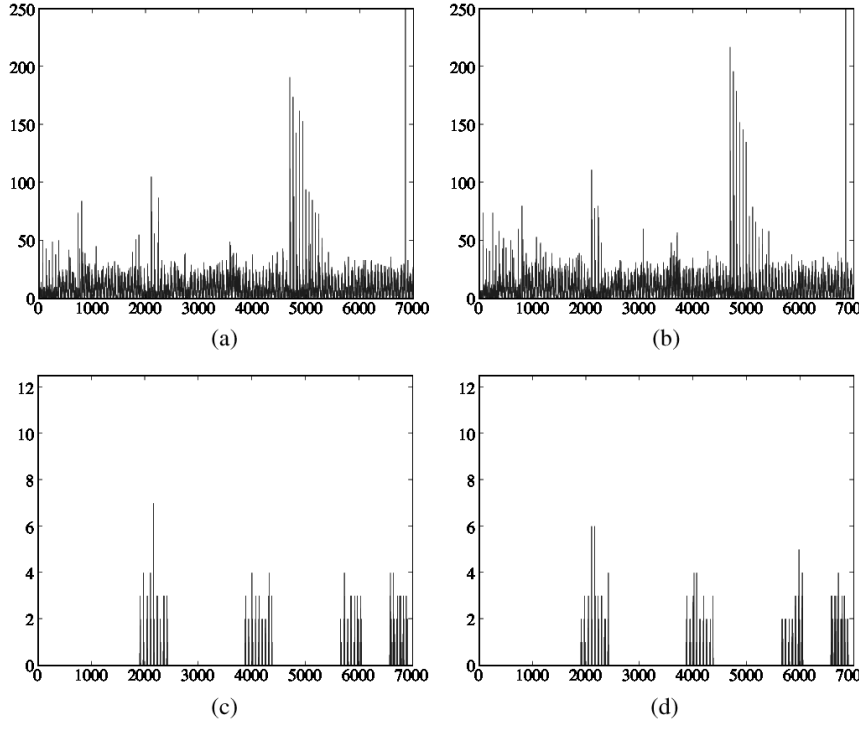


Fig. 5.5. Number of TCP/SYN packets globally exchanged within two particular monitors (a), (b) and received by the four attacked IP addresses within these two particular monitors (c), (d). (a) and (c) correspond to the traffic in monitor 1. (b) and (d) correspond to the traffic in monitor 2.

aggregated time series $(\underline{Z}(t),\ t = 1, \ldots, P)$ and $(\overline{Z}(t),\ t = 1, \ldots, P)$, as defined in (5.6), as well as the associated p-value. Note that the aggregated time series corresponds to the aggregation of 11 time series created by 11 different monitors where the attacked IP address has been detected. The p-value of the aggregated time series is equal to 5.65×10^{-6} and is thus much smaller than the ones determined at the local monitors. If the *BTopRank* algorithm had been used, the corresponding p-value would have been 5.4×10^{-3}.

The *DTopRank*, *BTopRank* and *TopRank* algorithms have also been further compared through 50 Monte Carlo replications in the cases of attacks having an intensity of 12.5 SYN/s and 25 SYN/s. We observed that *DTopRank* performs very similarly to the centralised algorithm, especially in the range of interest where the false alarm rate is about 1e-4 (recall that there

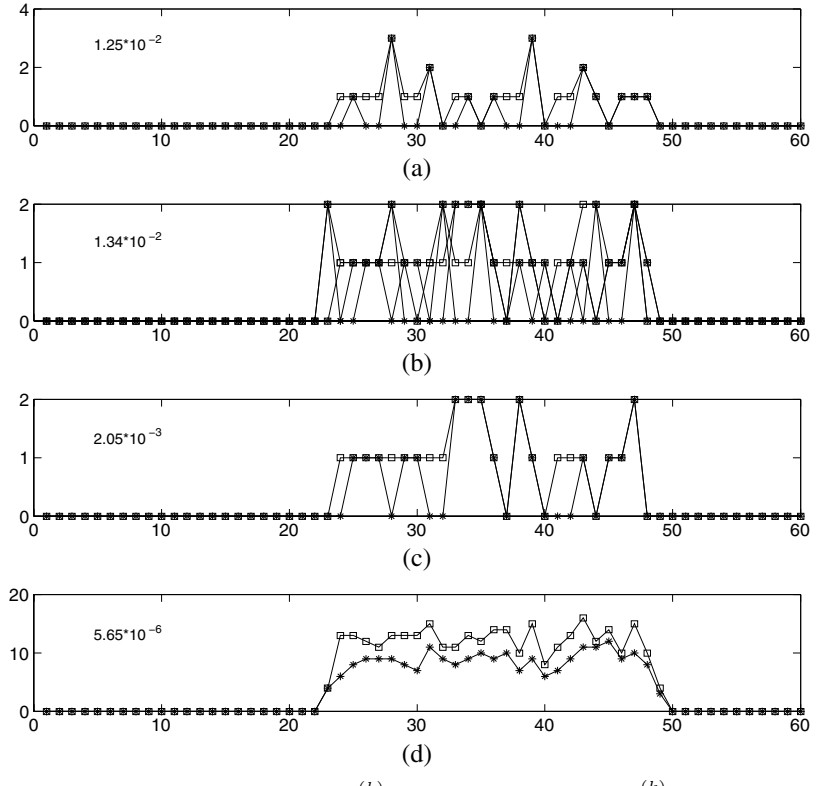


Fig. 5.6. (a), (b), (c): times series $(\underline{X}_i^{(k)}(t),\ t = 1, \ldots, 60)$ and $(\overline{X}_i^{(k)}(t),\ t = 1, \ldots, 60)$ displayed with (*) and (□), respectively, for three different values of k, (d): $(\underline{Z}_i(t),\ t = 1, \ldots, 60)$ and $(\overline{Z}_i(t),\ t = 1, \ldots, 60)$ displayed with (*) and (□), respectively.

are about 15,000 different IP addresses in each one-minute window). However, the quantity of data exchanged within the network is much reduced as the centralised algorithm needs to obtain information about, on average, 34,000 flows per minute whereas the *DTopRank* algorithm only needs to transmit the d upper- and lower-censored time series from the monitors to the collector. For $d = 1$ and $K = 15$, this amounts to 1800 scalars that need to be transmitted to the collector, versus 34000×5 (start and end time stamps, source and destination IP, number of SYN packets for each flow) for the centralised algorithm, resulting in a reduction of almost two orders of magnitude of the data that needs to be transmitted over the network. For further comparisons of these three different approaches, we refer the reader to Lung-Yut-Fong *et al.* (2012a).

References

Basseville, M. and Nikiforov, I. V. (1993). *Detection of Abrupt Changes*: *Theory and Applications* (Prentice Hall, Upper Saddle River, NJ).

Billingsley, P. (1968). *Convergence of Probability Measures* (Wiley, New York).

Brodsky, B. E. and Darkhovsky, B. S. (1993). *Nonparametric Methods in Change-Point Problems* (Kluwer Academic Publisher, Berlin).

Csörgő, M. and Horváth, L. (1997). *Limit Theorems in Change-point Analysis* (Wiley, New York).

Gombay, E. and Liu, S. (2000). A nonparametric test for change in randomly censored data, *Can. J. Stat.* **28**, 1, pp. 113–121.

Huang, L., Nguyen, X., Garofalakis, M., Jordan, M. I., Joseph, A. and Taft, N. (2007). In-network PCA and anomaly detection, in B. Schölkopf, J. Platt and T. Hoffman (eds), *Advances in Neural Information Processing Systems 19* (MIT Press, Cambridge, MA), pp. 617–624.

Kiefer, J. (1959). K-sample analogues of the Kolmogorov–Smirnov and Cramér–V. Mises tests, *Ann. Math. Statist.* **30**, pp. 420–447.

Krishnamurthy, B., Sen, S., Zhang, Y. and Chen, Y. (2003). Sketch-based change detection: Methods, evaluation, and applications, in *IMC '03*: *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement* (ACM, New York, NY), pp. 234–247.

Lakhina, A., Crovella, M. and Diot, C. (2004). Diagnosing network-wide traffic anomalies, in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (ACM, New York, NY), pp. 219–230.

Lévy-Leduc, C. and Roueff, F. (2009). Detection and localization of change-points in high-dimensional network traffic data, *Ann. Appl. Stat.* **3**, 2, pp. 637–662.

Li, X., Bian, F., Crovella, M., Diot, C., Govindan, R., Iannaccone, G. and Lakhina, A. (2006). Detection and identification of network anomalies using sketch subspaces, in *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement* (IMC 2006, Rio de Janeiro, Brazil) (ACM, New York, NY), pp. 147–152.

Lung-Yut-Fong, A., Lévy-Leduc, C. and Cappé, O. (2012a). Distributed detection/localization of change-points in high-dimensional network traffic data, *Statist. Comput.* **22**, 12, pp. 485–496.

Lung-Yut-Fong, A., Lévy-Leduc, C. and Cappé, O. (2012b). Homogeneity and change-point detection tests for multivariate data using rank statistics, Tech. rep., arXiv:1107.1971.

Page, E. S. (1954). Continuous inspection schemes, *Biometrika* **41**, pp. 100–115.

Paxson, V. (1999). Bro: A system for detecting network intruders in real-time, *Comput. Netw.* **31**, 23–24, pp. 2435–2463.

Roesch, M. (1999). Snort: Lightweight intrusion detection for networks, in *Proceedings of the 13th USENIX Conference on System Administration* (LISA 1999, Seattle, USA) (USENIX Association, Berkeley, CA), pp. 229–238.

Siris, V. A. and Papagalou, F. (2006). Application of anomaly detection algorithms for detecting syn flooding attacks, *Comput. Commun.* **29**, 9, pp. 1433–1442.

Tartakovsky, A., Rozovskii, B., Blazek, R. and Kim, H. (2006a). Detection of intrusion in information systems by sequential change-point methods, *Statistical Methodology* **3**, 3, pp. 252–340.

Tartakovsky, A., Rozovskii, B., Blazek, R. and Kim, H. (2006b). A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods, *IEEE T. Sign. Proces.* **54**, 9, pp. 3372–3382.

Wang, H., Zhang, D. and Shin, K. G. (2002). Detecting SYN flooding attacks, *Proceedings of the Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)* **3**, pp. 1530–1539.

Chapter 6

Monitoring a Device in a Communication Network

Nick A. Heard and Melissa J. Turcotte

Department of Mathematics, Imperial College London
London, SW7 2AZ, United Kingdom
n.heard@imperial.ac.uk

Anomalous connectivity levels in a communication graph can be indicative of prohibited or malicious behaviour. Detecting anomalies in large graphs, such as telecommunication networks or corporate computer networks, requires techniques which are computationally fast and ideally parallelisable, and this puts a limit on the level of sophistication which can be used in modelling the entire graph. Here, methods are presented for detecting locally anomalous substructures based on simple node and edge-based statistical models. This can be viewed as an initial screening stage for identifying candidate anomalies, which could then be investigated with more sophisticated tools. The focus is on monitoring diverse features of the same data stream emanating from a single communicating device within the network, using conditionally independent probability models. Whilst all of the models considered are purposefully very simple, their practical implementation touches on a diverse range of topics, including conjugate Bayesian inference, reversible jump Markov chain Monte Carlo, sequential Monte Carlo, Markov jump processes, Markov chains, density estimation, changepoint analysis, discrete p-values and control charts.

6.1. Introduction

Monitoring a large interaction network over time for anomalous behaviour is computationally very challenging. Typically each node will exhibit their own unique modes of behaviour, which will depend on the identity of their neighbours in the network with which they communicate, their own availability to communicate over time, the availability of their neighbours, and other seasonal factors which will affect their propensity to communicate.

The aim of this work was to build probability *models* of normal behaviour in an interaction network, either at the node level or at the edge level, and then use these probability models to *monitor* the network and detect anomalous departures from this normal behaviour. Departures from

normal behaviour are interesting, either from a commercial perspective of understanding changes in consumer requirements, or from a security perspective of detecting fraudulent use of an account.

To aid computational tractability, modelling of nodes and edges will rely upon simple independence assumptions. If the nodes are computing devices, such as PCs or mobile phones, such models lend themselves to analytics where the analysis can potentially be run on the device itself. In this scenario, very simple analytics are required which potentially use only a minimal proportion of the computational power of the device.

For each device, there is an associated multivariate stream of data with diverse characteristics and of differing dimension for different devices. The aim is to process this multivariate stream into a single time-varying score of surprise. Probability models are constructed for the full multivariate data stream using conditional independence assumptions, and surprising recent behaviours against this learnt model of normality are then sought.

A fundamental choice in this problem is whether to work in continuous or discrete time. This choice applies both to modelling and monitoring, and all four possibilities of continuous or discrete time modelling or monitoring are explored and finally compared.

As a concrete example, attention here will be focused on mobile phone communications; however, the framework being presented should be regarded as more general with applicability to computer networks and other security settings, requiring only domain-specific alterations to the proposed probability models.

The next section introduces a synthetic mobile phone data set which will be used for comparing the different approaches which are proposed. Sections 6.3 and 6.4, respectively, present continuous and discrete time models for the data streams. Sections 6.5 and 6.6 present monitoring tools in continuous and discrete time, respectively. Section 6.7 shows some results of applying different combinations of the modelling and monitoring methods to the synthetic mobile phone data, and some concluding remarks are made in Section 6.8.

6.2. VAST 2008 Challenge Data

A publicly available, unclassified, but synthetic, data set was provided by the VAST 2008 Challenge (http://www.cs.umd.edu/hcil/VAST-challenge08). The data are the records of mobile phone calls made within a small, fictional population of 400 nodes over a ten-day period.

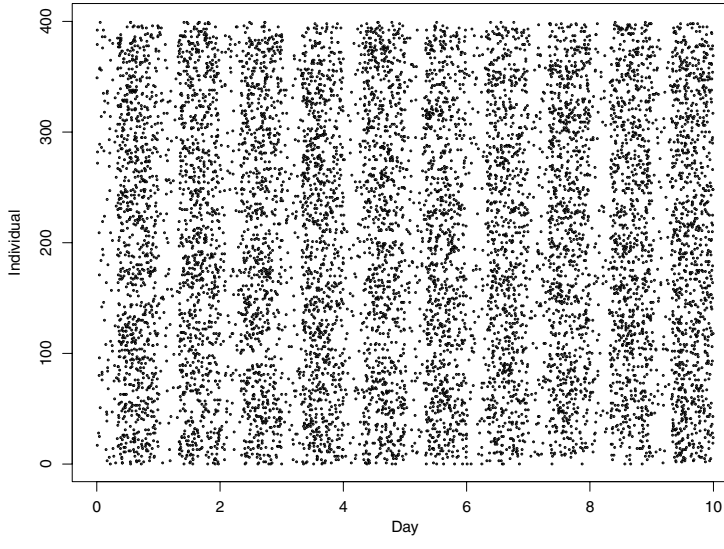


Fig. 6.1. VAST data by event time and source node.

Each call event is logged with full details of the metadata of that call: which node called which other node, what time the call started, what time the call ended, and the cell tower which the source node connected to when making the call, to provide an imprecise geolocation. The event times in the data set are plotted in Figure 6.1, and the diurnal seasonal patterns in the data are apparent from the stripey appearance of the data.

The records should provide critical information about an important social network structure. The aim of the challenge was to detect some anomalous activity from a small subset of the individuals sometime within the ten-day period. From the results of an award-winning published work on this challenge by Ye *et al.* (2008), which used a combination of the *PageRank* algorithm (Brin and Page, 1998) and visual analytic methods, there is good reason to suspect that the major anomalous activity occurs on the eighth day and involves a list of at least 11 individuals. For illustration purposes only, the subnetwork of those 11 individuals, made up of any nodes with which they communicated over the ten-day period, is plotted in Figure 6.2; the 11 anomalous nodes are shaded. In particular it is worth noting that individuals 200 and 300 are fairly central in the network of anomalous actors, but have many fewer communicating neighbours than the other anomalous nodes.

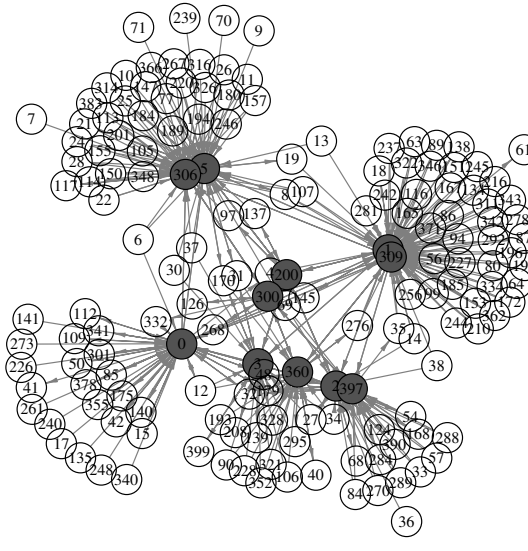


Fig. 6.2. Subgraph of VAST data from malicious actors' contacts.

6.3. Continuous Time Behavioural Modelling

6.3.1. *Markov jump processes*

Markov jump processes provide a natural framework for continuous time modelling of the transitions between different behaviours of a device in a communication network. There are many different ways to represent the status of node i in an interaction network using these processes and the following sections demonstrate some possibilities, ranging from a binary "currently connected" status indicator for a node through to full modelling of each individual edge. Other constructions are possible; for example, if the nodes have been clustered in some way, one could instead consider the frequencies of connections to particular node types. It should also be noted that this flexible framework readily extends to cases where measurable, possibly time-varying covariates are available for nodes or edges, as these simply present further terms in the equations for the transition intensities which now follow.

The following sections consider a communication network containing N nodes, labelled $\{1, 2, \ldots, N\}$. Denote the ordered event times at which the connections of node i begin as $0 < t_1^i < t_2^i < \cdots$. Let $d_1^i, d_2^i, \ldots$ be the durations of those connections; for $j = 1, 2, \ldots$, let $e_j^i \in \{1, 2, \ldots, N\}$ be the

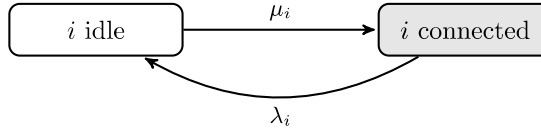


Fig. 6.3. Two-state "idle-connected" undirected Markov model.

identity of the node which i connected to at time t_j^i; finally, let $\delta_j^i \in \{0,1\}$ denote whether the jth connection to node i was outgoing ($\delta_j^i = 1$) and so initiated by node i, or incoming ($\delta_j^i = 0$) and so initiated by node e_j^i. Note that every connection appears twice in this notation, once as an outgoing connection and once as an incoming connection.

For a telephone communication network, it will be assumed that node i can be connected to at most one node at any one time. This corresponds to a requirement that $\forall j \geq 1, t_{j+1}^i > t_j^i + d_j^i$.

6.3.1.1. *Idle and connected states, undirected edges*

The two-state model in Figure 6.3 shows the simplest representation of the behaviour of node i.

The labelled transition intensity $\mu_i(t)$ is the rate at which node i makes connections at time t,

$$\mu_i(t) = \lim_{dt \downarrow 0} \frac{\mathbb{P}(i \text{ will become connected within } [t, t+dt) \,|\, i \text{ idle at time } t)}{dt}.$$

Once in the connected state, the transition intensity $\lambda_i(t)$ is the rate at which node i would terminate a connection at time t,

$$\lambda_i(t) = \lim_{dt \downarrow 0} \frac{\mathbb{P}(i \text{ will become idle within } [t, t+dt) \,|\, i \text{ connected at time } t)}{dt}.$$

6.3.1.2. *Idle and connected states, directed edges*

Figure 6.4 shows a three-state representation of the behaviour of node i, with a distinction now made between making a connection and receiving a connection; communications are now assumed to have a direction.

Here there are two possible transitions from the idle state. $\mu_{i\cdot}(t)$ is the intensity with which node i will initiate connection to any recipient,

$$\mu_{i\cdot}(t) = \lim_{dt \downarrow 0} \frac{\mathbb{P}(i \text{ will make a connection within } [t, t+dt) \,|\, i \text{ idle at time } t)}{dt},$$

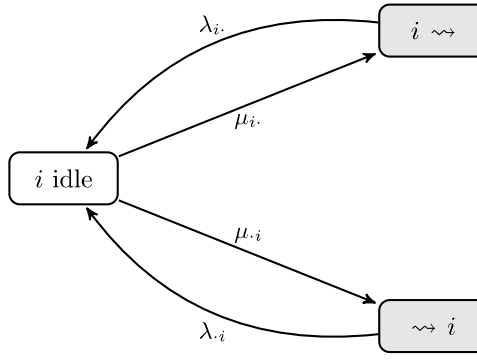


Fig. 6.4. Three-state "idle-connected" directed Markov model. $i \rightsquigarrow$ denotes i initiating a connection, whilst $\rightsquigarrow i$ denotes i receiving a connection.

whilst $\mu_{\cdot i}(t)$ is the intensity with which node i will receive a connection from any source,

$$\mu_{\cdot i}(t) = \lim_{dt \downarrow 0} \frac{\mathbb{P}(i \text{ will receive a connection within } [t, t+dt) \,|\, i \text{ idle at time } t)}{dt}.$$

To relate this model to the two-state model of Section 6.3.1.1, note that

$$\mu_i(t) = \mu_{i\cdot}(t) + \mu_{\cdot i}(t).$$

Similarly, $\lambda_{i\cdot}(t)$ and $\lambda_{\cdot i}(t)$ are the intensities with which node i will terminate an outgoing or incoming connection, respectively. If these two intensities were assumed to be equal, then they would relate to the previous model parameters through

$$\lambda_i(t) = \lambda_{i\cdot}(t) = \lambda_{\cdot i}(t),$$

but in general this may not be true.

6.3.1.3. *Embedded Markov chain for connection identities*

In the above models for the timing of connections, no consideration is given to the identities of the nodes with which node i is connecting. Further understanding of the normal behaviour of a node can be gained by learning the nodes with which it typically makes connections.

The next section will consider an extended Markov model to incorporate this level of detail. However, an alternative formulation can be specified by regarding the sequence of identities of the communicants as a separate, conditionally independent, time-stamped data stream.

Recall that the identities of the communication events of node i formed a sequence $e_1^i, e_2^i, \ldots$, where each $e_j^i \in \{1, 2, \ldots, N\}$. The simplest model for this sequence would assume each outcome in the sequence is an independent draw from a multinomial distribution with parameters

$$\theta_k^i = \mathbb{P}(e_j^i = k), \quad k = 1, 2, \ldots, N$$

where $\sum_{k=1}^N \theta_k^i = 1$. This model allows node i to have preferred nodes for connections, which can be learnt over time.

The conjugate prior for the vector of probabilities $(\theta_1^i, \theta_2^i, \ldots, \theta_N^i)$ is Dirichlet $(\alpha_1, \alpha_2, \ldots, \alpha_N)$ where by default each prior parameter $\alpha_k > 0$ might be set equal, or otherwise there is an option to *a priori* favour connectivity to *popular* nodes, using, for example,

$$\alpha_k \propto \text{indegree}(k).$$

A richer model, which does not assume independence of the identities, is to treat $e_1^i, e_2^i, \ldots$ as a Markov chain on $\{1, 2, \ldots, N\}$, with transition probabilities

$$\theta_{k,\ell}^i = \mathbb{P}(e_{j+1}^i = \ell \,|\, e_j^i = k), \quad k, \ell = 1, 2, \ldots, N$$

where $\sum_{\ell=1}^N \theta_{k,\ell}^i = 1$. This allows a first-order level of dependency in the connections of node i, where patterns of a connection with node k typically being followed by a connection with node ℓ are captured.

For fixed k, the vector of probabilities $(\theta_{k,1}^i, \theta_{k,2}^i, \ldots, \theta_{k,N}^i)$ are the conditional distribution of the identity of the next connection, given the most recent communication was with node k. Hence this vector of probabilities is the kth row of the transition probability matrix of the Markov chain for node i. Allowing each row of the transition probability matrix to be independent draws from the same Dirichlet prior as above completes the conjugate Bayesian specification of this model.

6.3.1.4. *Separate states for each connecting node, undirected edges*

Alternatively, as noted in Section 6.3.1.3, the state space of the continuous time Markov jump process can be extended to incorporate connection identities. An undirected specification, where no attention is paid to which node initiates the connection, is diagrammatically represented by Figure 6.5.

This model is much more heavily parameterised, with separate transition intensities (μ_{ij}, λ_{ij}) for all pairs of nodes in the network, but restricted here such that, say, $i < j$. Potentially this could lead to $O(N^2)$ level of

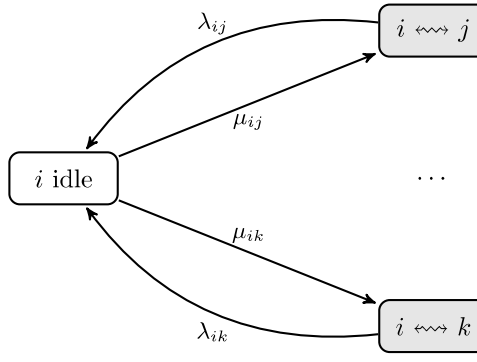


Fig. 6.5. Generalised undirected Markov model. $i \leftrightsquigarrow j$ denotes i and j connecting in either direction.

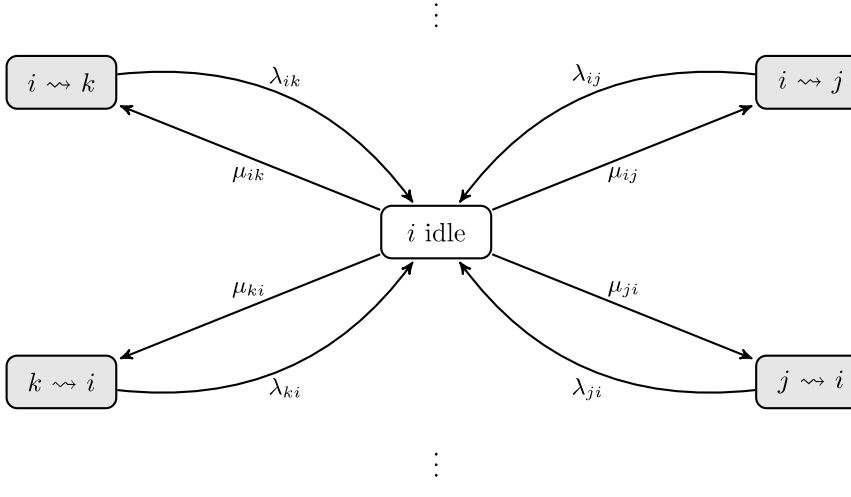


Fig. 6.6. Generalised directed Markov model. $i \rightsquigarrow j$ denotes i initiating a connection with node j.

computational effort, but in practice communication networks are sparse and the intensities for the majority of edges will be zero.

6.3.1.5. *Separate states for each connecting node, directed edges*

A further extension of Section 6.3.1.4 assumes the connections are also directed. This is represented by Figure 6.6.

Now there are separate transition intensities (μ_{ij}, λ_{ij}) for all pairs of nodes in the network, with no restriction. This is the richest possible representation of the communication status.

6.3.2. *Inference for Markov jump processes*

Even for the richest Markov jump process model (see Section 6.3.1.5), the data observed for each node can be divided up into separate (partially overlapping) data sets for each of the different transition intensities, such that learning each of the intensities can be seen as completely separate inference problems. Even from the idle state, an event time marking a connection to one node simply represents a censored observation period for waiting times for connections to all other nodes, which can be resumed once the node returns to the idle state.

For the task of anomaly detection, this provides another choice for the analyst. One may be interested in finding, say, a changepoint for a single intensity function μ_{ij} or λ_{ij}, to indicate that this specific relationship has changed; or taking a more global view of node i, one may wish to detect a changepoint which simultaneously affects, say, μ_{ij} for all j. (A third option, which is to look for changes in a subset of these intensities, is the subject of current work and beyond the scope of the work presented here.)

Let $Y_i(t) \in \{0, 1\}$ be an indicator function such that

$$Y_i(t) = 1 \iff i \text{ idle at time } t.$$

Taking the full model from Section 6.3.1.5 as an example, define $N_{ij}(t)$ to be the counting process of events $i \rightsquigarrow j$. $N_{ij}(t)$ can be analysed as an inhomogeneous Poisson process with intensity function

$$Y_i(t)Y_j(t)\mu_{ij}(t).$$

Note that the durations of the connections of node i (to any node, including j) and node j (to any node, including i) both act as a censoring mechanism for the process $N_{ij}(t)$.

Additionally, the connected durations also provide another data stream. These durations can be viewed simply as marks to the point process of connections; or in this extended Markov model, as observations from another state transition intensity, and this approach is followed here.

Supposing connection durations are also of inferential interest, let $\bar{N}_{ij}(t)$ be the counting process of $i \rightsquigarrow j$ connection terminations.

Also let $Y_{ij}(t) \in \{0, 1\}$ be an indicator function such that

$$Y_{ij}(t) = 1 \iff i \text{ connected to } j \text{ at time } t.$$

$\bar{N}_{ij}(t)$ can be analysed as an inhomogeneous Poisson process with intensity function

$$Y_{ij}(t)\lambda_{ij}(t).$$

Note that the pair of processes $(N_{ij}(t), \bar{N}_{ij}(t))$ are heavily dependent upon one another, since either $N_{ij}(t) = \bar{N}_{ij}(t)$ or $N_{ij}(t) = \bar{N}_{ij}(t) + 1$; additionally one always has zero intensity, since $Y_{ij}(t)$ cannot be nonzero when either $Y_i(t)$ or $Y_j(t)$ are zero.

However, their p-values (considered later) are asymptotically independent of one another as more and more observation time elapses.

6.3.3. *Seasonal changepoints*

Fast, tractable conjugate Bayesian inference for Markov jump processes is available if the state transition intensities $\lambda_{ij}(t)$ and $\mu_{ij}(t)$ are assumed constant and independently gamma distributed. But the inherent seasonality in the processes does not admit constant intensities for representing normal behaviour.

For example, looking at a daily level, it is likely there will be variability in connectivity between the night-time, daytime, evening, and so on. For example, see Figure 6.1, which illustrates this level of seasonality in the VAST data.

Let S be a seasonal period over which the processes are expected to show repetitive intensity patterns; for example, S might be the length of one day.

The process $N_{ij}(t)$ will be assumed here to be the censored counting process (implied by the state space diagrams) of events arising from an inhomogeneous Poisson process with intensity function

$$\mu_{ij}(t) = \mu^{ij}\ m_{ij}(t \bmod S)$$

where $m_{ij} : [0, S] \to \mathbb{R}^+$ is a probability density function for the time within the season at which a single connection would be made. A density function representation is helpful here, since the requirement of m_{ij} to have a fixed integral over $[0, S)$ (equal to 1) allows the multiplier μ^{ij} to be identifiable.

The intensity function of the process $N_{ij}(t)$ for normal behaviour therefore becomes

$$\mu^{ij} Y_i(t) Y_j(t)\ m_{ij}(t \bmod S). \tag{6.1}$$

Similarly, $\bar{N}_{ij}(t)$ will have intensity

$$\lambda^{ij} Y_{ij}(t) l_{ij}(t \bmod S) \tag{6.2}$$

where $l_{ij} : [0, S] \to \mathbb{R}^+$ is the probability density function of a single connection ending time for the same edge.

For (approximately) known density functions l_{ij}, m_{ij}, this construction then admits conjugate Bayesian inference when

$$\lambda^{ij}, \mu^{ij} \sim \Gamma(a, b).$$

The density functions l_{ij}, m_{ij} for capturing seasonality must be learnt from training data, and then periodically updated.

If some edges have sparse data, then it can be reasonable to assume that for $j \neq k$,

$$l_{ij} = l_{ik} = l_i,$$
$$m_{ij} = m_{ik} = m_i$$

and possibly even $l_i = m_i$. This also reduces the computational resource required.

6.3.3.1. *Bayesian changepoint density estimation*

To allow simple and tractable inference, the unknown density m_{ij} or l_{ij} is assumed to be piecewise constant on $[0, S)$, with k_S changepoints $\sigma_{1:k_S}$ ordered such that $0 = \sigma_0 < \sigma_1 < \cdots < \sigma_{k_S} < \sigma_{k_S+1} = S$.

Both the number of changepoints and their locations in $[0, S)$ are assumed to be unknown; they are assumed to follow a Poisson process with rate ν_S, and so have prior density

$$p(k_S, \sigma_{1:k_S}) = \nu_S^{k_S} e^{-\nu_S S}.$$

Conditional on $(k_S, \sigma_{1:k_S})$, let θ_j be the probability of an observation falling in the jth segment, $j = 1, 2, \ldots, k_S + 1$. Together the changepoints

and these probabilities specify a piecewise constant density, say

$$m(s) = \sum_{j=1}^{k+1} \mathbb{I}_{[\sigma_{j-1},\sigma_j)}(s) \frac{\theta_j}{\sigma_j - \sigma_{j-1}}, \quad s \in [0, S).$$

Straightforward conjugate Bayesian inference can be completed by specifying

$$[\theta_{1:k_S+1} \mid (k_S, \sigma_{1:k_S})] \sim \text{Dirichlet } (\alpha_1, \alpha_2, \ldots, \alpha_{k_S+1}),$$

where $\alpha_j = \alpha\{[\sigma_{j-1}, \sigma_j)\}$ and $\alpha(\cdot)$ is a base measure on $[0, S)$; the default choice being Lebesgue measure,

$$\alpha\{[\sigma_{j-1}, \sigma_j)\} = \alpha \frac{(\sigma_j - \sigma_{j-1})}{S}.$$

To make inference on the number and locations of good changepoints, let $\mathcal{D}$ be the data set of event times wrapped around into this season. That is, an event time at time $t \in [0, T]$ provides an observation at time $t \bmod S$ (the remainder of t after division by S). Let n_j be the number of such observations falling in the jth segment defined by the changepoints, and $n = \sum_{j=1}^{k_S+1} n_j$.

Care has to be taken to not double count the censorship induced by the alternating availability of the node to make a communication. Equations (6.1) and (6.2) already account for the fact that node i is not available to make a new connection whilst it is still engaged in an existing connection. So the density estimates should not reflect this censoring.

Strictly, this would require treating each seasonal period of data as a sample from a truncated version of $m_{ij}(s)$, but this breaks the conjugacy of the Dirichlet model. So instead an approximate solution is proposed.

Focusing on the connections made by node i, for $s \in [0, S]$, after observing n_S seasons let

$$\tilde{Y}_i(s) = \sum_{j=0}^{n_S} Y_i(s + jS)$$

be the number of seasons in which the node was being observed and available to make a connection at time s. Figure 6.7 shows a plot of this function for individual 1 from the VAST data. Notice that in the middle of the day, where most of the connections have been made, the connection durations have meant that the individual has been significantly less available to make or receive further connections during the ten-day observation period.

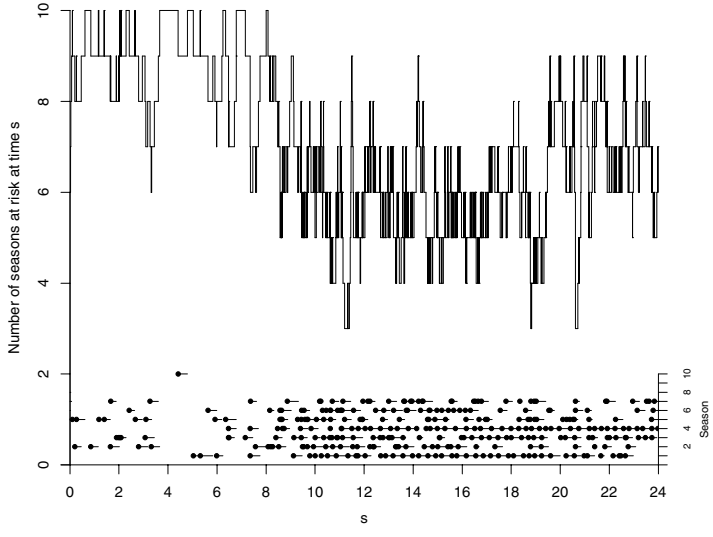


Fig. 6.7. $\tilde{Y_1}(s)$ for the VAST data, where s gives the hour of day. The points and lines at the bottom show the phone call event times and durations.

Define

$$a_j = \int_{s=\sigma_{j-1}}^{\sigma_j} \tilde{Y}_i(s)\alpha(ds),$$

$$s_j = \int_{s=\sigma_{j-1}}^{\sigma_j} \tilde{Y}_i(s)ds$$

to be, respectively, the base measure and Lebesgue measure of the total observation time of node i in the jth segment, and

$$a = \sum_{j=1}^{k_S+1} a_j$$

to be the base measure of the overall total observation time for node i.

Then approximate inference can proceed by defining

$$q_j = \frac{s_j}{n_S(\sigma_j - \sigma_{j-1})}$$

and then

$$\theta'_j = \frac{q_j\theta_j}{\sum_{i=1}^{k+1} q_i\theta_i}$$

as the realised probabilities of observing an event in the jth segment, and then performing conjugate inference assuming

$$[\theta'_{1:k_S+1} \,|\, (k_S, \sigma_{1:k_S})] \sim \text{Dirichlet } (a_1, a_2, \ldots, a_{k_S+1}).$$

Any estimate of θ' can be transformed back into true probabilities θ which are unaffected by censoring via

$$\theta_j = \frac{\theta'_j / q_j}{\sum_{i=1}^{k+1} \theta'_i / q_i}. \tag{6.3}$$

Under the conjugate Dirichlet model, the vector $\theta'_{1:k_S+1}$ can be integrated out, and reversible jump Markov chain Monte Carlo (RJMCMC) (Green, 1995) sampling of changepoints from the posterior distribution is trivial, with equilibrium density

$$p(k_S, \sigma_{1:k_S} \,|\, \mathcal{D}) \propto \nu_S^{k_S} \prod_{j=1}^{k_S+1} \frac{\Gamma(a_j + n_j)}{\Gamma(a_j) s_j^{n_j}}.$$

For the single estimate required here, the MAP number of changepoints is first obtained,

$$k_S^* = \underset{k_S}{\text{argmax}}\, p(k_S|\mathcal{D})$$

and then conditional on $k_S = k_S^*$, the MAP changepoints are obtained,

$$\sigma^*_{1:k_S} = \underset{\sigma_{1:k_S^*}}{\text{argmax}}\, p(k_S^*, \sigma_{1:k_S^*} \,|\, \mathcal{D}).$$

Using (6.3), the transformed posterior mean heights for the piecewise constant probability density function corresponding to these changepoints are given by

$$m_j^* \propto \frac{a_j + n_j}{(a+n)s_j}, \quad j = 1, 2 \ldots, k_S + 1.$$

Figure 6.8 shows the density estimate obtained for individual 3 from the VAST data when assuming the same seasonality for all communications with this node, using Lebesgue measure as the base measure.

6.4. Discrete Time Behavioural Modelling

An alternative range of modelling possibilities arise if the connection event data are first aggregated into counts on a discretisation of the time domain. For example, the data may be collected or otherwise processed into, say,

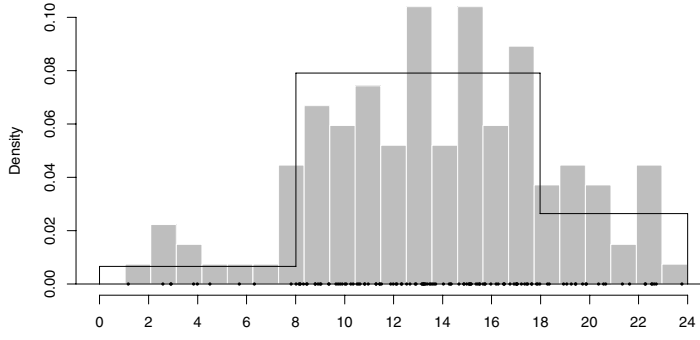


Fig. 6.8. Estimated $m_3(s)$ using the VAST data for individual 3, compared with a histogram of those data.

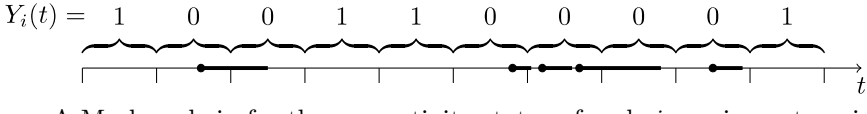
one-minute intervals, so that for each edge there would be a time series recording the number of connections observed each minute.

For consideration of whether a node or edge is idle or connected at each time point, the discrete time analogy of the Markov jump processes used in Section 6.3 are Markov chains. A simple Markov chain model is outlined here. In addition, since events have been aggregated there is potentially further information in the number of connections that have occurred in non-idle periods. Such models are not considered in depth here, but would be trivial to incorporate; an example implementation is given by Heard *et al.* (2010).

6.4.1. *Markov chain modelling of a node: idle and connected states*

Let $Z_i(t)$ be the number of connections for node i at the tth time point. For a binary representation, analogous to Section 6.3.2, let $Y_i(t) \in \{0, 1\}$ be the indicator variable for whether node i is idle at time t; so now in discrete time,

$$Y_i(t) = 0 \iff Z_i(t) > 0.$$



A Markov chain for the connectivity status of node i requires a transition probability matrix

$$\begin{pmatrix} \phi^i & 1-\phi^i \\ \psi^i & 1-\psi^i \end{pmatrix}$$

where

$$\phi^i = \mathbb{P}(Y_i(t) = 0 \,|\, Y_i(t-1) = 0)$$
$$\psi^i = \mathbb{P}(Y_i(t) = 0 \,|\, Y_i(t-1) = 1)$$

are the conditional probabilities of node i being connected at time t given that it was either connected or idle, respectively, at time $t-1$.

Assuming a constant transition probability matrix over time corresponds to an assumption that durations of connected periods follow a Geometric (ϕ^i) distribution rather than the exponential distribution from the continuous time model, although here no event distinction is made if a connection is terminated and a new one begins within the same discrete time period.

6.4.2. *Seasonal changepoints*

Seasonal variability in behaviour can be captured by dividing the seasonal period into segments and fitting separate Markov chains for transition behaviour within each seasonal segment. This provides a discrete time analogue of the changepoint model given in Section 6.3.3.

In discrete time, the vector of ordered changepoints $\sigma_{1:k_S}$ take values from a discrete set of points $\{1, 2, \ldots, S\}$. The presence of changepoints at each of these positions can be viewed as independent Bernoulli (ν_S) trials, leading to the discrete time analogue of the Poisson process prior of Section 6.3.3,

$$p(k_S, \sigma_{1:k_S}) = \nu_S^{k_S}(1-\nu_S)^{S-k_S}.$$

Conjugate beta distribution priors for the transition probability matrix parameters

$$\phi^i, \psi^i \overset{\text{iid}}{\sim} \text{Beta}(a_0, a_1)$$

allow the unknown transition probability matrices to be integrated out. And so similar to Section 6.3.3, RJMCMC sampling can be performed directly on the unknown number of unknown changepoints, with equilibrium distribution

$$p(k_S, \sigma_{1:k_S} \,|\, \mathcal{D}) \propto \nu_S^{k_S}(1-\nu_S)^{S-k_S} \times \prod_{j=1}^{k_S+1} \prod_{i=0}^{1} \frac{\Gamma(a_0+a_1)\Gamma(a_0+n_{i0}^j)\Gamma(a_1+n_{i1}^j)}{\Gamma(a_0)\Gamma(a_1)\Gamma(a_0+a_1+n_{i0}^j+n_{i1}^j)},$$

where (n^j_{i0}, n^j_{i1}) record the number of transitions from state i to states 0 and 1 respectively that have been observed in the jth seasonal period $\{\sigma_{j-1}, \sigma_{j-1}+1, \ldots, \sigma_j - 1\}$.

The seasonal changepoints can initially be learnt on a batch of training data, and then updated periodically.

6.4.3. *Node connection counts*

At a fine enough level of discretisation, often a binary view of activity will be sufficient. However, for those time points when node i is not idle, the magnitudes of the number of connections in them can be regarded as a further data stream.

A model for $[Z_i(t) \mid Z_i(t) > 0]$ can be assumed to be independent from the model for $Y_i(t)$. When the connection counts are of inferential interest, this two-level model is still advantageous as it provides a hurdle or zero-inflated component for capturing the over-expression of zeros common in connectivity data.

6.4.4. *Edge-level modelling*

The identities of the edges for each connection of node i can be viewed as a further data stream. Sequentially each identity can be modelled as independent samples, or as a dependent, irregularly time-spaced Markov chain, as proposed for continuous time in Section 6.3.1.3.

Alternatively, each edge could be treated completely separately as a separate modelling stage. Taking only those discrete time points at which the node is active, identical Markov chain modelling is performed at the edge level. In this setting, separate seasonal changepoints can be learnt for each edge. This might not be preferable if data on some edges is sparse; however, it may be necessary if seasonality variability is different for the different edges emanating from a node.

6.5. Continuous Time Behavioural Monitoring

The previous two sections have presented continuous and then discrete time approaches to modelling the arrival of events in a communication network, and the durations of these events.

Attention is now turned to the task of monitoring the network for anomaly detection. Like modelling, this can be done either in continuous or discrete time. This section considers continuous time behavioural

monitoring, which is only suitable when implemented in conjunction with a continuous time model for the data. In contrast, the next section will examine discrete time monitoring, which can be used with either an underlying continuous time or discrete time model for the event processes.

6.5.1. *Changepoint analysis*

Continuous time anomaly detection is most naturally framed as a continuous time changepoint problem, where changepoints mark changes in behaviour. Since a node may undergo several changes in behaviour over time, multiple changepoints are considered.

Recall (6.1) for the intensity function $\mu_{ij}(t)$ of the process $N_{ij}(t)$ under normal behaviour. Changes in the overall level of connectivity from normal behaviour will act as changepoints in the scalar μ^{ij}.

For each of a sequence of monitoring times $t_0 < t_1 < t_2 < \cdots$, inference will be made about the changepoints which have occurred in the process based on observation over $[t_0, t_n]$. The changepoints which have occurred by t_n will be denoted $\tau_{1:k_n} = (\tau_1, \ldots, \tau_{k_n})$, and are assumed to arrive as a homogeneous Poisson process with intensity ν, and so have prior density

$$p(k_n, \tau_{1:k_n}) = \nu^{k_n} e^{-\nu t_n}.$$

The scalar multipliers μ^{ij} for the intensities are assigned conjugate priors,

$$\mu^{ij}_{0:k_n} = (\mu^{ij}_0, \ldots, \mu^{ij}_{k_n}) \overset{\text{iid}}{\sim} \Gamma(\alpha, \beta),$$

and so these multipliers can be integrated out from the likelihood function without any estimation. It follows that the changepoint posterior distribution is given up to proportionality by

$$\pi_{[t_0,t_n]}(\tau_{1:k_n}, k_n) \propto \gamma_{[t_0,t_n]}(\tau_{1:k_n}, k_n) = \nu \mathrm{e}^{-\nu t} \prod_{k=0}^{k_n} \frac{\beta^\alpha}{\Gamma(\alpha)} \frac{\Gamma(\alpha + r_k)}{(\beta + y_k)^{r_k+\alpha}}$$

where $\tau_0 = t_0$, $\tau_{k+1} = t_n$ and r_k is the number of observed connections in the kth segment and

$$y_k = \int_{t=\tau_k}^{\tau_{k+1}} Y_i(t)\, Y_j(t)\; m_{ij}(t \bmod S) dt$$

is the seasonally rescaled total observation time spent in the kth segment.

After observing the network from t_0 until the first monitoring time point t_1, an approximate sample of changepoints $(\tau_{1:k_1}, k_1)^{(i)}$ from the

posterior $\pi_{[t_0,t_1]}(\tau_{1:k_1}, k_1)$ are easily obtained by reversible jump MCMC (Green, 1995). In most cases, there will be a low probability of there being any changepoints at this stage. Subsequently, an efficient sequential Monte Carlo (SMC) algorithm for updating changepoint samples (Turcotte and Heard, 2013) can exploit the similarity of $\pi_{[t_0,t_{n-1}]}$ and $\pi_{[t_0,t_n]}$ at successive monitoring times.

6.5.2. *Anomaly function*

Following Turcotte and Heard (2011), to construct a measure of anomaly at time t, let $g(t)$ be the time that has passed since the last changepoint,

$$g(t) = t - \tau_{i^*},$$
$$i^* = \max_i \{\tau_i \leq t\}.$$

Small values of $g(t)$ correspond to recent change and therefore anomalous behaviour.

Note that the prior expectation and variance of $g(t)$ are both increasing with t, and therefore it is preferable to work with a standardised alternative

$$h(t) = \frac{g(t) - \mathbb{E}[g(t)]}{\sqrt{\mathbb{V}\mathrm{ar}[g(t)]}}.$$

For monitoring, Turcotte and Heard (2011) consider two approaches. The first calculates the posterior expectation $\mathbb{E}[h(t)]$, and monitors how this evolves over time; this should take highly negative around the time of an anomaly.

The second approach calculates and monitors the posterior probability $\mathbb{P}(h(t) < 0) >$, since this should be high around the time of an anomaly. This approach has the advantage that it is more easily calibrated, and anomalies can be flagged whenever

$$\mathbb{P}(h(t) < 0) > \alpha$$

for some α, say 0.95.

Both monitoring tools are considered in Section 6.7.

6.6. Discrete Time Behavioural Monitoring

Finally, the case of monitoring using discrete time analytics is considered. The well-studied field of quality control charts provides the ideal framework here. These charts measure the level of surprise at each discrete time point,

and then aggregate these levels of surprise over time. An important article with examples close to the current context was published by Lambert and Liu (2006).

In Heard *et al.* (2010), each node or edge had a single data stream from which a predictive p-value was obtained at each discrete time point to measure current surprise. In the presence of multiple data streams, approximately conditionally independent p-values are calculated for each data stream at any given time point, and then combined using Fisher's method to obtain a single measure of surprise which will serve as the unit for a quality control chart.

6.6.1. *Quality control chart*

For a collection of independent p-values $p^1, \ldots, p^k$, Fisher's method combines these separate measures of surprise into a single score

$$X^2 = -2 \sum_{i=1}^{k} \log p^i. \tag{6.4}$$

When the null hypotheses which give rise to the p-values are correct and the p-values arise from continuous distributions, then each p-value is Uniform$[0, 1]$ and it follows that $X^2 \sim \chi^2_{2k}$.

At monitoring time point t_n, the aim is to combine independent uniform p-values obtained for the most recently observed data from each aspect of the multivariate data stream using Fisher's method, to give a single measure of surprise p_n.

Since each of these combined, time-indexed p-values $\{p_n\}$ should be independently uniformly distributed on $[0, 1]$ under the null hypothesis of normal behaviour, they can be transformed to real values $\{Z_n\}$ which independently follow a standard normal distribution under the same hypothesis, using Stouffer's Z-score method

$$Z_n = \Phi^{-1}(1 - p_n).$$

Now outlying (small p-value) behaviour at time t_n will correspond to a large value of Z_n.

Finally, to accumulate evidence of anomalous behaviour over time, following Lambert and Liu (2006) the proposed method runs an exponentially weighted moving average ($EWMA$) chart $\{S_n\}$ on the Z-scores $\{Z_n\}$,

$$S_n = (1 - w)S_{n-1} + wZ_n, \quad n \geq 1, \tag{6.5}$$

where $S_0 = 0$, with the dual benefits that more recent values carry highest weight but recent surprise over successive intervals can also be accumulated. The tunable parameter $w \in [0, 1]$ controls the level of significance placed on the most recent Z-score.

This model has well-understood boundaries under the null hypothesis (Lambert and Liu, 2006), of the form

$$L\sqrt{\frac{w}{2-w}[1-(1-w)^{2n}]}.$$

6.6.2. *Discrete p-values*

The discrete time monitoring analytic proposed in Section 6.6.1 relies on p-values which are truly Uniform$[0, 1]$ under the null hypothesis. However, even from continuous time models, the discrete time monitoring statistics which will be considered in this section will have discrete or mixed type distributions and hence the resulting p-values will not have this property.

For example, for the discrete time model which takes a binary view of activity status, for computational tractability p-values need to be calculated for each calculated binary observation of activity status and then combined using a method such as Fisher's. The crude discrete p-values from a Bernoulli distribution can only take two values, and so are very far from uniform.

6.6.2.1. *Basic theory of discrete p-values*

Let X be a discrete random variable on $\mathbb{N} = \{0, 1, 2, \ldots\}$. Denote the probability mass function (pmf), survivor function and cumulative distribution function, respectively, as

$$\begin{aligned} p_x &= \mathbb{P}(X = x) \\ s_x &= \mathbb{P}(X \geq x) = 1 - \sum_{j=0}^{x-1} p_j \\ f_x &= \mathbb{P}(X \leq x) = 1 - s_x + p_x. \end{aligned}$$

Note that $s_0 \equiv 1$ and $\lim_{x\to\infty} s_x = 0$. Without loss of generality, it can be assumed that $p_0 > 0$.

Then, for example, one-sided upper-tail p-values of draws from X are discrete random variables s_X with range $\{s_0 > s_1 > \cdots\}$ and pmf

$$\mathbb{P}(s_X = s_x) = p_x, \quad x = 0, 1, 2, \ldots.$$

Boolean outcomes such as idle/connected can be labelled as 0 or 1 in such a way that $p_0 \geq p_1$. Count random variables could be similarly relabelled if such p-values defined as

$$p\text{-value}(x) = \sum_j \mathbb{I}_{[0,p_x]}(p_j)p_j.$$

were preferred.

A problem in the context of this work is that p-values for discrete random variables are discrete and therefore not $\mathrm{U}[0,1]$ random variables. Instead they are only approximately uniform when a latent variable U is drawn conditional on the realised value of X, as now demonstrated.

Theorem 6.1. *Suppose*

$$X \sim p_x,$$
$$[U|X = x] \sim \mathrm{U}(s_{x+1}, s_x].$$

Then $U \sim \mathrm{U}[0,1]$.

Proof. For $u \in [0,1]$, U has density function

$$f(u) = \sum_x p_x f(u|x) = \sum_x (s_x - s_{x+1}) \frac{\mathbb{I}_{(s_{x+1},s_x]}(u)}{s_x - s_{x+1}}$$
$$= \sum_x \mathbb{I}_{(s_{x+1},s_x]}(u) = \mathbb{I}_{[0,1]}(u). \qquad \square$$

Noting the range of U in the above theorem, it follows immediately that discrete p-values, which deterministically take the value s_x, are stochastically larger than $\mathrm{U}[0,1]$ random variables.

Corollary 6.1. *If* $X \sim p_x$, *then* $\mathbb{E}(s_X) > \frac{1}{2}$.

Additionally, a second result is easily obtained:

Theorem 6.2. *If* $U \sim \mathrm{U}[0,1]$ *and* $X \sim p_x$, *then* $\mathbb{P}(U < s_X) > \frac{1}{2}$.

Proof.

$$\mathbb{P}(U < s_X) = \sum_x p_x \mathbb{P}(U < s_x) = \sum_x p_x s_x$$
$$= \sum_x p_x \sum_{j \geq x} p_j = \frac{1 + \sum_x p_x^2}{2} > \frac{1}{2}. \qquad \square$$

In conclusion, discrete p-values are too large if they are going to be used in Fisher's method, which assumes U$[0, 1]$ random variables. Note that the same results hold for lower-tail p-values.

To bridge this gap, discrete p-values should be regarded as an interval censored observation of a truly uniform p-value, as prescribed by Theorem 6.1. This is analogous to drawing a random variable X from a continuous distribution, but then interval censoring X such that it is only known that X lies between two integers x and $x + 1$. A discrete observation of $X = x$ then corresponds to the censored observation of a p-value

$$s \sim \mathrm{U}(s_{x+1}, s_x]. \tag{6.6}$$

The next sections present two alternative solutions to this problem: the first is a simple deterministic correction, the second is a numerical approximation of an expected true p-value obtained from Monte Carlo simulation.

6.6.2.2. *Deterministic adjustment*

A simple correction is to use an adjusted p-value

$$s_x^* = \frac{1}{2}(s_x + s_{x+1}) = s_x - \frac{p_x}{2},$$

which is the midpoint of the range of the uniform latent variable in (6.6).

Theorem 6.3. *If $X \sim p_x$ and $U \sim \mathrm{U}[0, 1]$, then $\mathbb{E}(s_X^*) = \frac{1}{2}$ and $\mathbb{P}(U < s_X^*) = \frac{1}{2}$.*

Proof. Since $p_x = s_x - s_{x+1}$,

$$\mathbb{E}(s_X^*) = \frac{\sum_x (s_x - s_{x+1})(s_x + s_{x+1})}{2} = \frac{\sum_x s_x^2 - s_{x+1}^2}{2} = \frac{s_0}{2} = \frac{1}{2}.$$

Also,

$$\mathbb{P}(U < s_X^*) = \sum_x p_x s_x^* = \frac{\sum_x (s_x - s_{x+1})(s_x + s_{x+1})}{2} = \frac{1}{2}. \qquad \square$$

Although s^*_X is of course still discrete, as an adjusted p-value it at least possesses these two important properties of U[0, 1].

The preceding arguments have all been concerned with upper-tail p-values s_x. For lower-tail p-values, there is an analogous correction to f_x, given by

$$f^*_x = \frac{1}{2}(f_x + f_{x-1}) = 1 - s^*_x.$$

Corollary 6.2. *If $X \sim p_x$ and $U \sim \mathrm{U}[0,1]$, then $\mathbb{E}(f^*_X) = \frac{1}{2}$ and $\mathbb{P}(U < f^*_X) = \frac{1}{2}$.*

Finally, for two-sided p-values, a little more care is required; this is the subject of the next section.

6.6.2.3. *Two-sided adjusted p-values*

For $x = 0, 1, 2, \ldots$ define

$$g_x = \min(s_x, f_x).$$

Let x_m be the median of X satisfying both $s_{x_m} \geq \frac{1}{2}$ and $f_{x_m} \geq \frac{1}{2}$.

Then since s_x and f_x are, respectively, non-increasing and non-decreasing in x, g_x is unimodal in x with maximum at x_m. An example of g_x from a Poisson random variable is plotted in Figure 6.9.

For $x = 0, 1, 2, \ldots$, to define p-values let

$$R_x = \{j : g_x \geq g_j\}.$$

If $x = x_m$ then $R_x = \mathbb{N}$. Otherwise, since g_x is unimodal then $\mathbb{N} \backslash R_x$ is a finite subinterval of $\mathbb{N}$, say $R_x = \mathbb{N}\backslash\{a_x, a_x+1, \ldots, b_x\}$ where $a_x \leq x_m \leq b_x$ and either $a_x = x - 1$ if $x < x_m$ or else $b_x = x + 1$ if $x > x_m$.

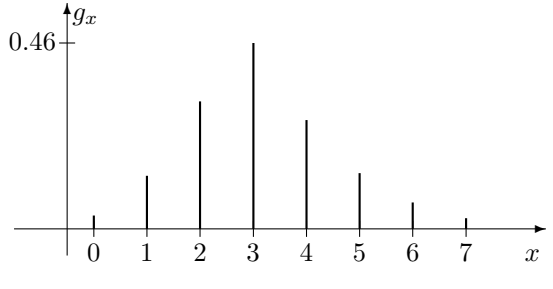


Fig. 6.9. g_x for $X \sim \mathrm{Poi}(3.5)$.

An unadjusted, two-sided p-value for an observed x is

$$h_x = \sum_{j \in R_x} p_j,$$

but this has the same uniformity issues as the one-sided p-values. For example, an observation of the median has unadjusted p-value $h_{x_m} = 1$.

By identical reasoning to the one-sided case, a suitable adjusted two-sided discrete p-value is

$$h_x^* = h_x - \frac{p_x}{2}.$$

Corollary 6.3. *If $X \sim p_x$ and $U \sim \mathrm{U}[0,1]$, then $\mathbb{E}(h_X^*) = \frac{1}{2}$ and $\mathbb{P}(U < h_X^*) = \frac{1}{2}$.*

6.6.2.4. *P-values for variables of mixed type*

Another important case which needs to be considered is the behaviour of p-values from variables of mixed type; that is, variables whose distribution has both a continuous part and a discrete part. Here, attention is restricted to the case where such a variable arises through right-censoring of a continuous time to event variable.

Suppose X is a random variable of mixed type with distribution

$$\mathbb{P}(X \le x) = \begin{cases} \int_0^x \zeta(u)du, & x < T \\ 1, & x \ge T, \end{cases}$$

as illustrated in Figure 6.10.

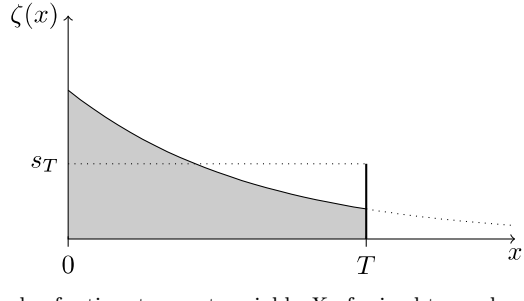


Fig. 6.10. Example of a time to event variable X of mixed type, due to right-censoring at T.

Let $f_T = \int_0^T \zeta(u)du$ be the probability of X being drawn from the continuous part (the shaded area of Figure 6.10), and $s_T = 1 - f_T$ be the probability that $X = T$ (the height of the spike at T in Figure 6.10).

An upper-tail p-value s_X has distribution

$$\mathbb{P}(s_X \leq s) = \begin{cases} 0, & 0 < s < s_T \\ s, & s_T \leq s \leq 1. \end{cases}$$

So p-values from observations of X are also of mixed type.

A lower-tail p-value f_X has distribution

$$\mathbb{P}(f_X \leq f) = \begin{cases} f, & 0 < f < f_T \\ 1, & f_T \leq f \leq 1. \end{cases}$$

Define a corrected upper-tail p-value,

$$s_x^* = \begin{cases} \dfrac{s_T}{2}, & s_x = s_T \\ s_x, & s_T < s_x \leq 1 \end{cases}$$

where the first case corresponds to the point mass at T and the second case corresponds to the continuous part. Note that when X has been sampled from the continuous part, no correction is made to the p-values. Otherwise if $X = T$, then the corrected p-value is the midpoint of $(0, s_T]$.

Similarly, define a corrected lower-tail p-value,

$$f_x^* = \begin{cases} f_x, & f_x = 0 \leq f_x < f_T \\ \dfrac{f_T + 1}{2}, & f_x = f_T. \end{cases}$$

In this case if $X = T$, then the corrected p-value is the midpoint of $[f_T, 1)$, and otherwise no correction is required.

Finally, the definition of a two-sided p-value depends upon whether the censoring value T is beyond the median of the continuous density function ζ, since this determines whether or not s_T is greater than one half. If $s_T > 1/2$, then the two-sided p-value is simply the lower-tail p-value f_x since f_X can never exceed s_X. On the other hand, if $s_T \leq 1/2$, then the two-sided p-values from a mixed type variable behave just like those from a continuous random variable, and therefore require no correction. Hence a

corrected two-sided p-value is

$$h_x^* = \begin{cases} 2g_x, & s_T \leq 0.5 \\ f_x^*, & s_T > 0.5. \end{cases}$$

Then $\{s_x^*\}$, $\{f_x^*\}$ and $\{h_x^*\}$ have the properties stated in the earlier corollaries for corrected discrete p-values.

6.6.2.5. *Expected adjustment*

Alternatively, rather than performing an immediate deterministic adjustment on p-values, each observed p-value can be preserved as a uniform random variable. For example, by equation (6.6) an observation of $X = x$ simply spawns an upper-tail p-value variable with a $\mathrm{U}(s_{x+1}, s_x]$ distribution.

When several independently observed p-values $p^1, p^2, \ldots, p^n$ are going to be combined, typically arising from different distributions, if the ith p-value p^i is uniform on $(a_i, b_i]$ then, for example, expectations of functions of interest such as Fisher's score (6.4) can be taken with respect to their joint distribution with density

$$\prod_{i=1}^{n} \frac{\mathbb{I}_{(a_i,b_i]}(p^i)}{b_i - a_i}.$$

Monte Carlo estimation of expectations arising from this joint distribution are trivial to perform, but carry an increased computational cost.

In the current work, both the deterministic and Monte Carlo corrections were observed to give very similar answers to one another.

The next two sections outline the p-values that will arise, respectively, from the continuous time and discrete time models of Sections 6.3 and 6.4, which are corrected using the methods from this section before being streamed to a control chart (see Section 6.6.1).

6.6.3. *Poisson process p-values*

Suppose an inhomogeneous Poisson process has recently been observed over $(t_{n-1}, t_n]$ (see Section 6.3.2), producing two pieces of summary data:

(1) a total number of events $N(t_n) - N(t_{n-1}) = k$;
(2) event times $t_{n-1} = t^{(0)} < t^{(1)} < \cdots < t^{(k)} < t_n$ and inter-arrival times $x_1 = t^{(1)} - t_{n-1}, x_2 = t^{(2)} - t^{(1)}, x_k = t^{(k)} - t^{(k-1)}$ and perhaps a right-censored arrival time $x_{k+1} = t_n - t^{(k)}$.

Respectively there are two natural p-values to consider (and corresponding lower-tail and two-sided analogues):

(1) $\mathbb{P}(N(t_n) - N(t_{n-1}) \geq k)$;
(2) $s_{x_1}, s_{x_2}, \ldots, s_{x_k}, s^*_{x_{k+1}}$ combined via Fisher's method.

Sverdrup's (unpublished) observations on transition intensities imply that the first would be more powerful when the intensity of connections is low, otherwise the second would have more power.

6.6.4. *Bernoulli process p-values*

Suppose a Bernoulli process has recently been observed over discrete time points $\{t_{n-1}, t_{n-1}+1, \ldots, t_n\}$ (see Section 6.4.1). Note that observing a two-state Markov chain for node i in Section 6.4.1 (rather than independent samples) equates to alternating between the observation of two independent Bernoulli processes with parameters ϕ^i and ψ^i.

For each t in $\{t_{n-1}, t_{n-1}+1, \ldots, t_n\}$, let $Y_i(t) \in \{0, 1\}$ be the activity status of node i at time t.

If t falls in the jth seasonal segment, then having previously observed (n^j_{i0}, n^j_{i1}) transitions from state i to states 0 and 1, respectively, in that segment up to time $t-1$, the conjugate beta distribution priors for the transition probabilities imply independent posterior distributions

$$\phi^i \sim \text{Beta}(a_0 + n^j_{00}, a_1 + n^j_{00}),$$
$$\psi^i \sim \text{Beta}(a_0 + n^j_{10}, a_1 + n^j_{11}).$$

So the predictive distribution for the next observation in the chain is

$$p_0 = \mathbb{P}(Y(t) = 0 \,|\, Y(t-1) = i) = \frac{a_0 + n^j_{i0}}{a_0 + a_1 + n^j_{i0} + n^j_{i1}},$$
$$p_1 = \mathbb{P}(Y(t) = 1 \,|\, Y(t-1) = i) = 1 - p_0.$$

If $p_0 = p_1$ then the next observation is uninformative as the crude discrete p-value will surely take value 1 and, when viewed as a partial observation, the unobserved p-value will simply be Uniform$[0, 1]$ under the null hypothesis.

Otherwise, without loss of generality suppose $p_1 > p_0$. Then if $Y(t) = 0$, the crude p-value will be p_0 and the unobserved part will be Uniform$[0, p_0]$; otherwise if $Y(t) = 1$, the crude p-value will be 1 and the unobserved part will be Uniform$(p_0, 1]$.

Using either of the corrections of Section 6.6.2, each of the independent p-values for $t \in \{t_{n-1}, t_{n-1}+1, \ldots, t_n\}$ are combined using Fisher's method to obtain a single measure of surprise for this monitoring window.

6.7. Results for the VAST Data

The plots in the following sections show control charts or changepoint curves for a selection of the different model and analytic combinations when applied to the VAST data of Section 6.2, using 100 update windows over the ten-day period. Continuous time modelling followed by either discrete or continuous monitoring is then followed by discrete time modelling and monitoring.

Both directed and undirected views of the call-time data streams of a node or edge can be considered in four different ways. In the first, the direction of the communications is ignored and all calls are treated as homogeneous events; in the second, just outgoing calls are counted; in the third, just incoming calls are counted; in the fourth, outgoing calls and incoming calls provide two separate counting processes suitable for a joint analysis. For node-level analyses, the effects of including the caller identities or the cell towers as additional data streams are also considered.

Each line in the plots represents one node from the network. If present, a dashed line indicates a possible threshold for decision making, where a node is marked as anomalous if its curve crosses the threshold. In some cases a near optimal choice of threshold for this data set is shown, to show the best performance that could have been achieved.

The darkest grey lines in the plots, along with identity numbers to the right in bold face, indicate malicious actors who have been detected; any other known malicious actors are coloured medium grey, and their identifiers are shown in italic face. The lightest grey lines potentially indicate false positives, as these actors are not known to be malicious.

6.7.1. *Continuous time Markov jump process model*

First, continuous time modelling (Section 6.3) is applied to the VAST data. Sections 6.7.1.1 and 6.7.1.2 present the resulting control charts from the discrete time analytic S_n; these two sections contrast the performance achieved from modelling the aggregated call behaviour of a node against splitting up the data streams for each of the different identities with which the node communicates. Section 6.7.1.3 then presents results from the continuous time analytics $\mathbb{E}[h(t)]$ or $\mathbb{P}(h(t) < 0)$.

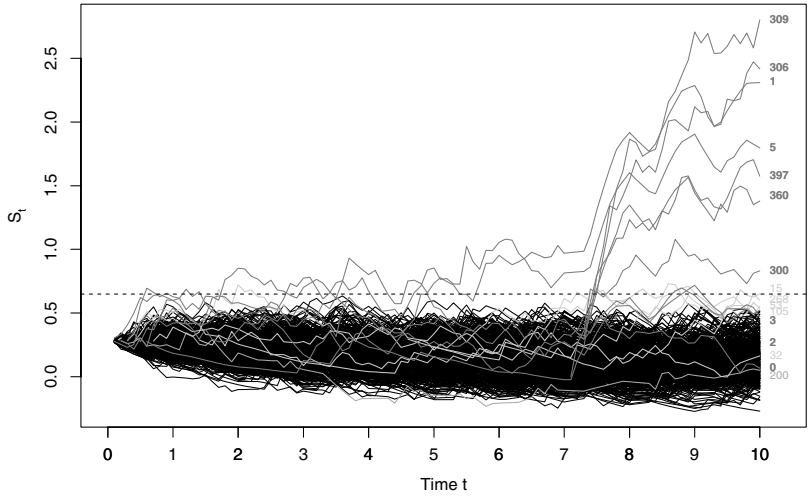


Fig. 6.11. VAST data control charts: continuous time model for undirected node event times.

6.7.1.1. *Control charts when monitoring node connection times*

Figure 6.11 shows the control chart S_n from (6.5) for each individual in the VAST data set when looking at all calls made or received by the individual as one series.

The dashed line shows the threshold which would give optimal performance in terms of maximising true positives and minimising false positives for these particular control charts; any individual whose control chart ever exceeds the threshold is marked as having been flagged as anomalous.

Note that individual 200, earlier seen to be at the centre of the malicious subnetwork in Figure 6.2, is not detected at the optimal threshold. In fact, individual 200 is not detected by any of the methods presented here, since his call behaviour does not noticeably change; it is more realistic to find this individual through "guilt by association", as he regularly communicated with malicious individuals 1, 2, 3 and 5, who are all detected here; 1 and 5 are detected with particularly strong significance. However, to achieve just one false negative, four false positive anomalies are being flagged (indicated in light grey).

Figure 6.12 shows the control charts obtained when only looking at the outgoing call behaviour of each individual. Surprisingly, this causes a huge reduction in detection power, with five malicious actors going undetected; reducing the threshold to detect more malicious individuals leads

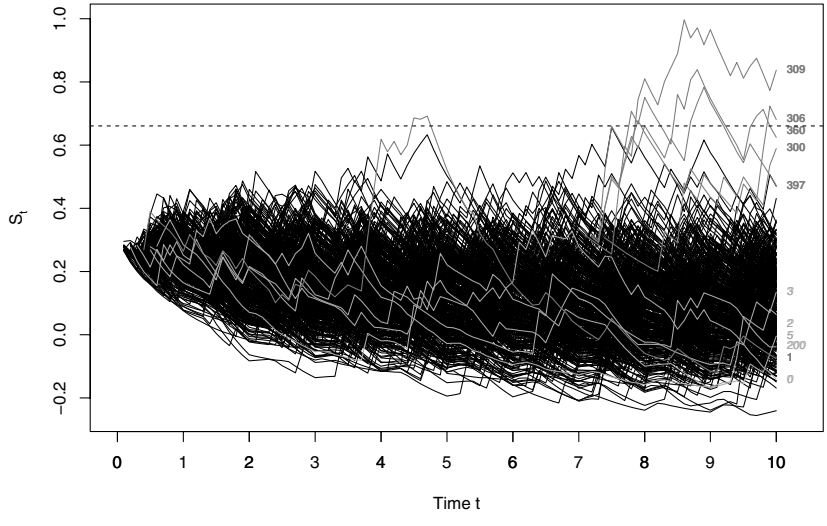


Fig. 6.12. VAST data control charts: continuous time model for outgoing event times for each node.

to an avalanche of false positives. Clearly, there is only a weak signal in the outgoing call behaviour here.

In contrast, Figure 6.13 shows the control charts obtained when only looking at the incoming calls received by each individual. Performance here exceeds that achieved when merging the incoming and outgoing calls; at the optimal threshold, all of the malicious actors except individual 200 is found, but this time with only two false positives.

A strong signal in the incoming call behaviour is interesting, perhaps suggestive of an organisation where the malicious actors are receiving instructions rather than relaying. To explore if stronger detection power could be obtained from the incoming call data, Figure 6.14 shows the results from adding a second data stream to the analysis of each node, using a Markov chain to model the identities of the sources of the incoming calls (see Section 6.3.1.3). However, this can be seen to be only injecting noise into the analysis, with the false positive rate increasing.

Very similar results are obtained (and so not shown here) if the originating cell tower for the incoming calls is added as a data stream in the analysis. A similar increase in false positives is observed. However, in both cases, it appears that some of the malicious actors would be detected sooner using these extra data streams.

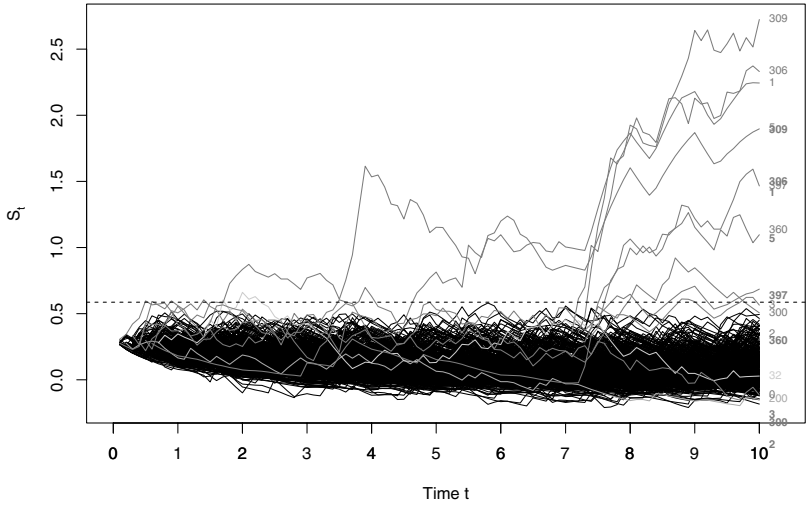


Fig. 6.13. VAST data control charts: continuous time model for incoming event times for each node.

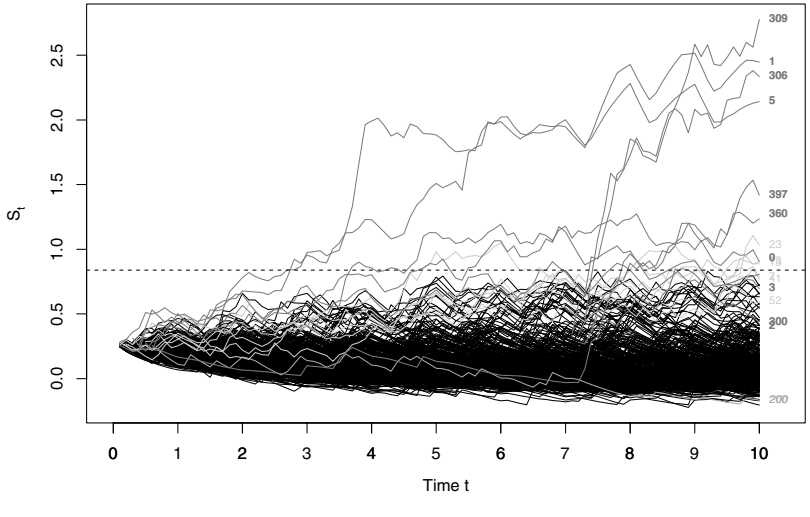


Fig. 6.14. VAST data control charts: continuous time model for incoming event times and Markov model for corresponding incoming caller for each node.

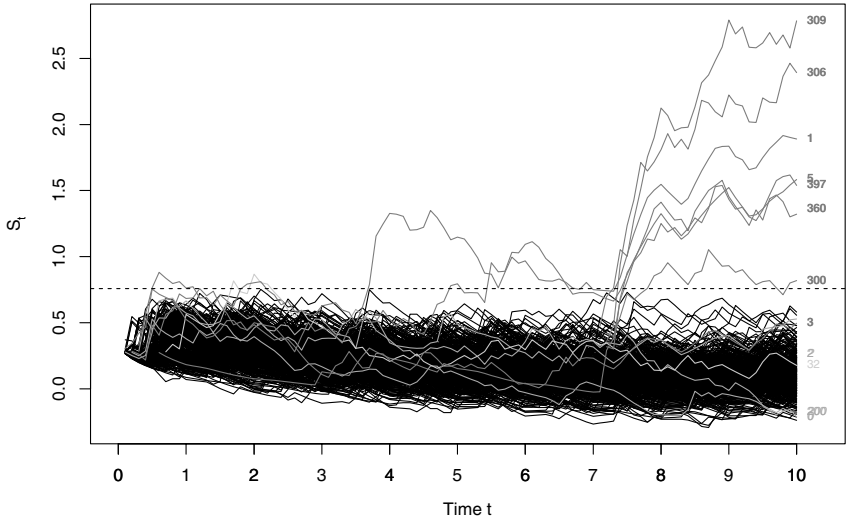


Fig. 6.15. VAST data control charts: continuous time models for incoming and outgoing event times for each node.

Finally for this approach, the incoming and outgoing calls of each node were treated as two separate data streams in the analysis. The resulting control charts are shown in Figure 6.15, and, as in Figure 6.11, the inclusion of outgoing call data is shown to reduce performance, even when introduced as a separate data stream.

6.7.1.2. *Control charts when monitoring edge connection times of each node*

In these analyses, separate incoming and outgoing call data streams were created for each relationship of a node. The p-values for the different relationships of a node were then combined into a single surprise score to make control charts. As in the previous section, four different representations of the data were considered: a merged incoming/outgoing call stream; outgoing calls; incoming calls; outgoing and incoming calls as two separate streams.

Interestingly the control charts behave quite differently from the analyses of Section 6.7.1.1, but the detection performance is very similar, if slightly inferior. Again, outgoing calls are found to carry the least information, and including both incoming calls and outgoing calls as two data

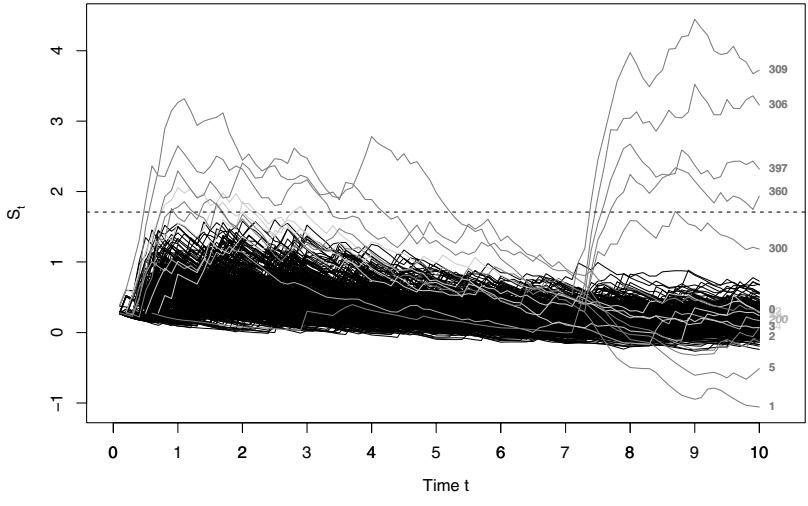


Fig. 6.16. VAST data control charts: continuous time models for incoming and outgoing event times for each edge of a node.

streams, now for each edge in the network graph, gives the best performance of all. The control chart for this analysis is shown in Figure 6.16.

6.7.1.3. *Changepoint analysis results*

In this section, results are presented for the same continuous time model and data streams as Section 6.7.1.2, but now using the continuous time anomaly detection analytics based on changepoint discovery (Section 6.5.1).

As with the discrete time analytic results in Section 6.7.1.1, the incoming calls were found to carry the strongest signal, and a joint analysis of the outgoing and incoming call streams outperforms analysis when merging the two streams together. Note that in this section, the thresholds for $\mathbb{P}(h(t) < 0)$ were fixed at 95% and so have not been chosen to be optimal; therefore direct comparisons both within this section and between sections are not intended. Figures 6.17 and 6.18 show the anomaly detection functions $\mathbb{E}[h(t)]$ and $\mathbb{P}(h(t) < 0)$ from Section 6.5.2 when modelling the incoming call data for each node.

In the plot of $\mathbb{E}[h(t)]$ (Figure 6.17) the known malicious actors are indicated by their identifiers, and it is unclear how a simple threshold could be applied here without absorbing a large number of false positive anomalies; in contrast, for the plot of $\mathbb{P}(h(t) < 0)$ (Figure 6.18) a threshold is simply

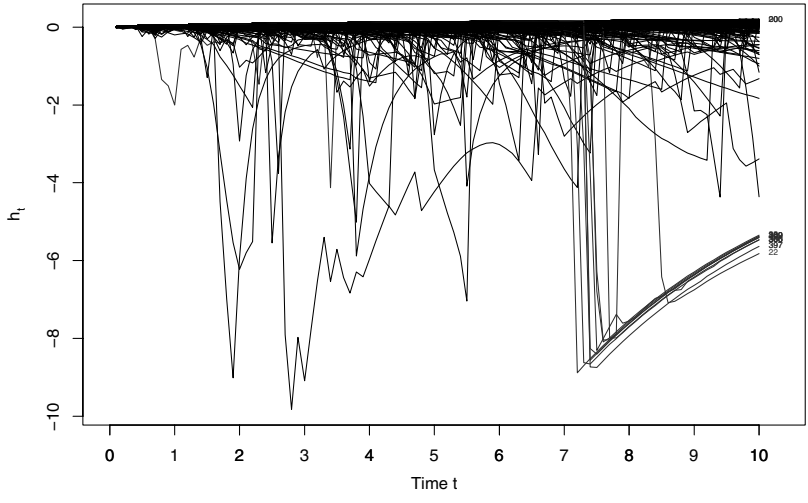


Fig. 6.17. VAST data $\mathbb{E}[h(t)]$: continuous time model for incoming event times for each node.

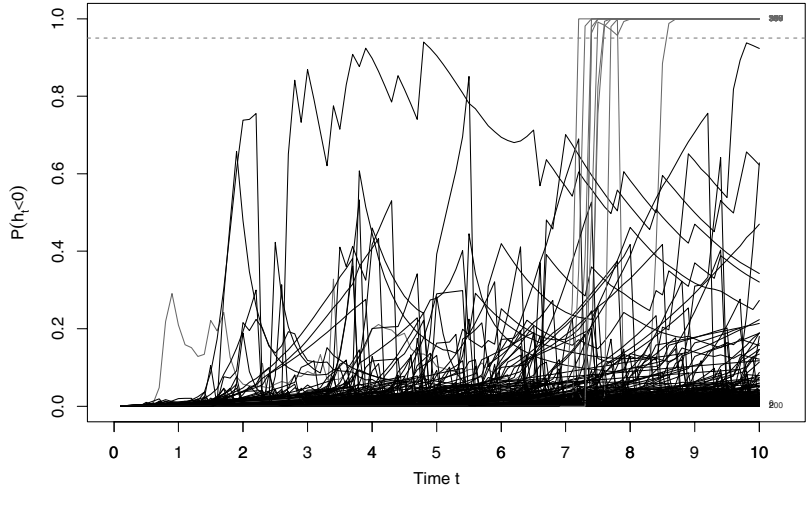


Fig. 6.18. VAST data $\mathbb{P}(h(t) < 0)$: continuous time model for incoming event times for each node.

set at 95% and for this merged data stream all but one of the malicious actors are detected, along with a handful of false positives.

6.7.2. *Discrete time Markov chain model*

In this section the discrete time model (Section 6.4) is applied to the VAST data, and output is shown from the discrete time control chart analytic. To form discrete time series from the VAST data, the ten days of data were discretised into binary activity status indicator variables for each minute, meaning each series was of length 14,400 and therefore each update window consisted of 144 observations of activity status.

As in previous sections, the following analyses were considered: a merged incoming/outgoing call stream; outgoing calls; incoming calls; outgoing and incoming calls as two separate streams. Again, incoming calls were found to carry much greater signal than outgoing calls under this model. However, in this case, similar performance is obtained whether analysing just incoming calls (displayed in Figure 6.19), or both incoming and outgoing calls either as one merged stream or as two separate streams. Overall though, there is much less separation between the control charts of malicious actors and those of the rest of the network. In particular, the other central node from the malicious subnetwork of Figure 6.2, individual 300,

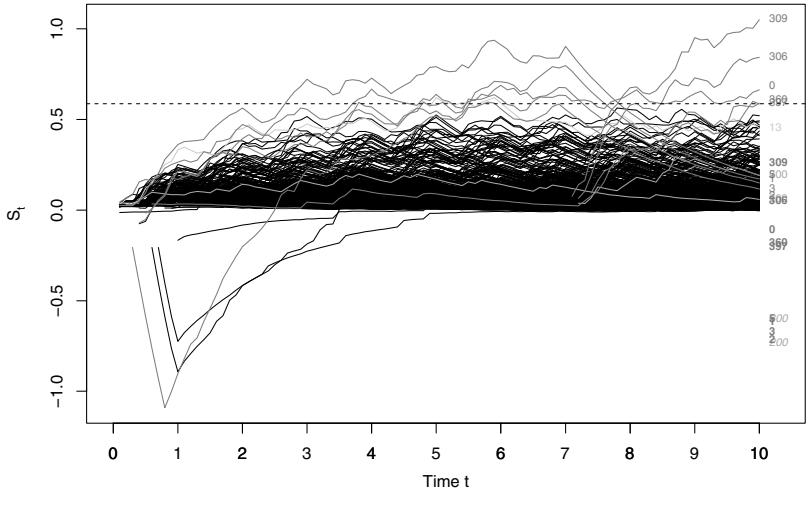


Fig. 6.19. VAST data control charts: discrete time model for incoming node event times for each node.

can no longer be detected without accepting an avalanche of false positives, and is always below the optimal threshold.

6.8. Conclusions

The control charts and changepoint plots of Section 6.7 yield the following tentative conclusions about the methods and the VAST data, based on this analysis.

6.8.1. *Summary of methods*

Continuous time modelling gave more accurate inference and is computationally faster than discrete time modelling, which also requires an arbitrary discretisation of the time domain. In contrast, continuous time monitoring and discrete time monitoring gave very similar inference, and it is discrete time monitoring that has the greater computational simplicity (with no requirement for costly MCMC or SMC simulation).

6.8.2. *Summary of VAST data results*

Although the analysis was only performed on synthetic data, it is interesting to be able to gain insight into how the anomalies in the data were generated. The majority of the signal for detecting the malicious actors in the VAST 2008 Challenge data was in the change in pattern of their incoming calls. The outgoing calls held almost no information. This finding was consistent across all of the different analytics.

Beyond this, there was limited further information to be extracted from the identity of the incoming callers, or the cell towers used by those callers; earlier detection of some malicious actors came with a price of a higher false positive rate. Splitting the data stream into separate process for each edge led to more noise rather than more signal. It seems the information in the VAST 2008 Challenge data was in the aggregated call patterns.

References

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine, *Comput. Networks ISDN* **30**, pp. 107–117.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination, *Biometrika* **82**, pp. 711–732.

Heard, N. A., Weston, D. J., Platanioti, K. and Hand, D. J. (2010). Bayesian anomaly detection methods for social networks, *Ann. Appl. Stat.* **4**, pp. 645–662.

Lambert, D. and Liu, C. (2006). Adaptive thresholds, *J. Am. Stat. Assoc.* **101**, 473, pp. 78–88.

Turcotte, M. J. M. and Heard, N. A. (2011). Continuous time changepoint detection with applications in dynamic networks, Tech. rep., Imperial College London. Available upon request from the authors.

Turcotte, M. J. M. and Heard, N. A. (2013). An efficient sequential Monte Carlo algorithm for sampling multiple changepoints in continuous time, Tech. rep., Imperial College London. Available upon request from the authors.

Ye, Q., Zhu, T., Hu, D., Wu, B., Du, N. and Wang, B. (2008). Cell phone mini challenge award: Social network accuracy – Exploring temporal communication in mobile call graphs, in *Proceedings of IEEE VAST Symposium* (IEEE, Piscataway, NJ), pp. 207–208.

Index