ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2015-ENST-0053

École doctorale Informatique,
Télécommunications et Électronique
(Paris)

THESIS

submitted to

TELECOM ParisTech

in partial fullfillment of the requirements for the degree of

Doctor of Philosophy

Speciality: Networking and Security

defended by

Hadrien Hours

on September 16, 2015

A causal approach to the study of telecommunication networks

Advisor: **Prof. Dr. Ernst Biersack**
Co-advisor: **Prof. Dr. Patrick Loiseau**

Committee in charge

M. Guillaume Urvoy-Keller, Prof.	Université de Nice Sophia Antipolis, France	Reviewer
M. Paulo Gonçalves, Prof.	ENS Lyon, Lyon, France	Reviewer
M. Kun Zhang, Prof.	Max Planck Institute, Tübingen, Germany	Examiner
Mrs. Michèle Sebag, Prof.	Université Paris Sud, Gif Sur Yvette, France	Examiner

THÈSE

TELECOM ParisTech
member college of Institut Mines-Télécom, and of ParisTech

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2015-ENST-0053

École doctorale Informatique,
Télécommunications et Électronique
(Paris)

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « réseaux et sécurité »

présentée et soutenue publiquement par

Hadrien Hours

le 16 septembre 2015

Une approche causale de l'étude des réseaux de télécommunications

Directeur de thèse : **Prof. Ernst Biersack**
Co-directeur de thèse : **Prof. Patrick Loiseau**

Jury

M. Guillaume Urvoy-Keller, Prof.	Université de Nice Sophia Antipolis, France	Rapporteur
M. Paulo Gonçalves, Prof.	ENS Lyon, Lyon, France	Rapporteur
M. Kun Zhang, Prof	Max Planck Institute, Tübingen, Allemagne	Examinateur
Mme Michèle Sebag, Prof.	Université Paris Sud, Gif Sur Yvette, France	Examinateur

THÈSE

TELECOM ParisTech
école de l'Institut Mines-Télécom – membre de ParisTech

A causal approach to the study of telecommunication networks

Thesis

Hadrien Hours
hadrien.hours@eurecom.fr

École Doctorale Informatique, Télécommunication et Électronique, Paris
ED 130

September 16, 2015

Advisor:
Prof. Dr. Ernst Biersack
EURECOM, Sophia-Antipolis

Co-Advisor:
Assistant Prof. Dr. Patrick Loiseau
EURECOM, Sophia-Antipolis

Reviewers:
Prof. Dr. Gonçalves Paulo,
ENS Lyon, Lyon

Prof. Dr. Urvoy-Keller Guillaume,
Université de Nice Sophia Antipolis,
Sophia-Antipolis

Examiners:
Prof. Dr. Sebag Michèle,
Université Paris Sud, Gif Sur Yvette

Prof. Dr. Zhang Kun,
Max Planck Institute, Tübingen.

Acknowledgements

“Though the road’s been rocky it sure feels good to me.”, Bob Marley

It has been a rocky road where doubts were often present and the destination often unsure. But I had the great opportunity to work under the supervision of Pr. Ernst Biersack and Pr. Patrick Loiseau, who I want to thank for their constant support and understanding whether professionally or humanely. Different in their personal and professional way of seeing and handling things, each of them has given me all the necessary advice, support and belief that led to the results presented in our work.

“To travel is to take a journey into yourself.”, T. S. Eliot

And I would add that the people you meet are usually the best part of the travel.

I would never thank enough all the amazing friends I shared these years with, in the good and hard moments. Jonas Zaddach, who started his Ph.D. the same day as mine and shared the whole journey with me, and Captain Martina Cardone, who never left the boat, who shared so many moments that will stay for ever in my mind as life-time memories. They have been here in all the good and hard moments of this Ph.D. and it is in them that I found the strength to keep moving forward in the hard times. A special thank to Quentin Jacquemart, who always had his door and mind open to share problems, solutions and great evenings. Paul and Claudia that have always brought this warm and particular ordered disorder that makes things always look good whatever the issue you face. Finally, a special thank to Miriam, who really believed in me and supported me in the hardest times of my Ph.D. when it would have been easy to get lost if not for a light that guided you.

“Our family is a circle of strength of love, with every birth and every union the circle grows.”

And I would add that even when one has to leave the circle then the bounds get stronger so the one who left is always present. I want to thank my family who has always managed to give me the strength and will to pursue the direction I chose and the possibility to follow the path of research and this Ph.D.

“Il n’est de vent favorable pour celui qui ne sait où il va.”, Seneca

My last thank should go to Pr. Kun Zhang who showed a lot of support for our work and, while we were somehow outsiders of the world of *Causality*, believed in the potential of our initiative.

Abstract

Telecommunication networks are complex systems that are taking more and more importance in our daily life. Every day sees a new service being created or a new technology being developed. More and more actors participate in the extensive use of telecommunication networks in many aspects of our life (Heath, Business, Social Networks, Gaming, Education, Security, Military). Telecommunication networks are complex and heterogeneous systems. The usage of telecommunication networks and the different technologies on which they rely are in constant evolution.

Due to this evolution, an important number of parameters need to be taken into account to study the mechanisms and performance of telecommunication networks. Classical approaches base their understanding of a given system on the observation of simultaneous or correlated events. Such approaches are then limited by the way they tackle the problem. To understand a system in perpetual evolution one has no choice but to try to infer the underlying mechanisms responsible of what can only be observed, the structural dependencies between the components forming the system being studied. The inference of the structural mechanisms responsible for the behavior and performance of a system and the study and use of these mechanisms to understand observable systems is a well known problem. One possible way to tackle this issue relies on the concepts and theory of *Causality*.

Many studies have been made to understand and develop the concepts of causality. From Aristotle to Judea Pearl many important names have participated in the development and understanding of causality. However, recent advances made in the formalization, definition and use of the causal theory to model and to understand scientific problems has given a new opportunity to structure the knowledge and the understanding of any system that one is given to observe. In particular, the adoption of graphical causal models to represent the structural dependencies of systems, along with graphical criteria to study the causal effects of its parameters, greatly simplifies the causal studies of systems as complex as telecommunication networks.

In the special case of telecommunication networks, the system we want to study presents many constraints that make the application of the causal theory challenging. Exhibiting these challenges and solving them is at the hear of our work.

Telecommunication networks, because of their complexity and the facility to obtain observational data, represent a very good candidate to perform a causal study of their performance. The causal approach that we use in our studies of telecommunication networks performance allows us to predict its performance if we were to modify the parameters that influences its behavior. Due to the cost, risk and difficulty to lead control experiments in the domain of telecommunication networks, our work is based on the passive observations of a given system exclusively. Our causal approach is able to predict the effect of altering the mechanisms responsible of the behavior of the system being observed and design strategies of intervention to improve its performance.

Most of the methods that exist to perform the causal study of a given system make strong assumptions regarding the nature of the data obtained from the observations of this system. Unfortunately, in the domain of telecommunication networks, the nature of the underlying mechanisms do not follow the assumptions of linearity, normality or discreteness made by these methods. By identifying the different constraints that make the causal study of telecommunication networks different from other domains where this approach is usually adopted, we are able to adapt the existing methods and to fully exploit the benefits of a causal approach. We show the benefits of choosing a causal approach and we compare our approach to previous empirical studies or to studies based on correlation only.

An important part of our work focuses on the different challenges that arise when performing a causal study of real case scenarios. We insist on the methodology to adopt both in the implementation of existing methods and on the adaptation of these existing implementations to the constraints of telecommunication networks. Three different studies are presented and, at the end of each of these studies, we draw the lessons that have been learned facing the different challenges met in these studies.

In Chapter 1 we present telecommunication networks. We provide the necessary background and expose the different mechanisms that will support the work presented in this thesis. In the same chapter, we present the inference of the causal model of a system and the use of such model to predict interventions based on passive observations exclusively. Finally, we will conclude Chapter 1 presenting the motivations behind the use of a causal approach to study telecommunication networks.

In Chapter 2 we present works related to the one that will be presented in this thesis. We will first focus on works in the domain of the causal theory that supported or could extend our work. We then present existing studies of telecommunication networks and of telecommunication network based application. These studies will often provide the necessary background to design and interpret the studies presented in later chapters.

In Chapter 3, we highlight the constraints, challenges and solutions to adapt methods that already exist to the causal study of telecommunication networks.

Chapter 4 exposes an early study that made of the Dropbox protocol performance. In this chapter, the main focus is set on the importance of the causal study design and on the risks that we exposed ourselves to when trying to use existing implementations to study systems that do not obey the rules for which these implementations were defined.

In Chapter 5, we first parameterize and validate the methods presented in Chapter 3 with the study of an emulated network. We then present the study of an Internet application, FTP, that shows the benefits of our approach and the methods that were validated in the study of the emulated network.

In Chapter 6, we extend our methods and present the study of a more complex scenario, the impact of the DNS service on the CDN user throughput.

Chapter 7 presents our conclusions regarding the adoption of a causal study to understand telecommunication networks performance. We summarize the different steps of our work and the lessons that have been learned.

In a last chapter, Chapter 8, we provide some insights on future works, some of which are currently being made and others that we see as interesting extensions of the work presented in this thesis.

Many studies, that participate to the development of the methods presented in our work, are referred to in the main body of our thesis and can be found in the Appendices.

Contents

Abstract **iii**

1 Introduction and background **1**

1.1 Introduction 1

1.2 Telecommunication networks 3

1.2.1 Protocols 3

1.2.2 Presentation 3

1.2.3 Internet 5

1.2.4 TCP/IP 5

1.2.5 TCP 7

1.2.6 Specificity of the Internet 13

1.3 Causality 16

1.3.1 Correlation and causality 16

1.3.2 Interventions and counterfactuals 17

1.3.3 Definition of causal operators 19

1.3.4 Bayesian networks and causal models 20

1.3.5 Works, studies and software 31

1.3.6 Summary 32

1.4 Causality in telecommunication networks 33

1.4.1 Motivation 33

1.4.2 Benefits 34

2 Related work **35**

2.1 Telecommunication networks 35

2.1.1 TCP performance study 36

2.1.2 Application performance studies 39

2.2 Causality 41

2.2.1 Theory 41

2.2.2 Toolboxes 44

2.3 Causal study of telecommunication network performance 45

3 Coping with telecommunication network constraints **51**

3.1 Problem statement 51

3.2 Independence tests 51

3.2.1 Constraints of telecommunication network parameters . . 52

3.2.2 Kernel based independence tests 54

3.3 KCI+bootstrap parameterization 60

3.3.1 Completion time 60

3.3.2 Accuracy of the KCI and bootstrap tests 63

3.4 Probability density function 70

3.4.1 Telecommunication network parameter constraints 70

3.4.2 Kernels and copulae 70

3.5 Studying telecommunication applications 73

3.5.1 Adopting a new approach to study a known system 73

3.5.2 The difficulty of choosing the granularity of our models . . 74

3.5.3 Adopting a progressive study approach 75

4 Dropbox protocol study **77**

4.1 Introduction 77

4.2 Dropbox presentation 78

4.3 Dropbox and Intrabase 80

4.3.1 Intrabase 80

4.3.2 System observation 80

4.3.3 Bulk transfer statistics 81

4.3.4 Concluding remarks 83
4.4 The study of Dropbox at the packet level 85
4.4.1 Presentation 85
4.4.2 Observations of the Dropbox traffic 85
4.4.3 Results 86
4.4.4 Concluding remarks 88
4.5 Lessons learned 90
4.5.1 Causality is not always the good approach 90
4.5.2 Degree of abstraction 91
4.5.3 Complexity and completeness 91
4.5.4 Design of the experiments 92
4.5.5 Preprocessing of the data 93

5 Study of the File Transfer Protocol 95
5.1 Emulated network 95
5.1.1 Mininet network emulator 95
5.1.2 Experimental setup 96
5.1.3 Causal model 98
5.1.4 Predictions 107
5.1.5 Concluding remarks 109
5.2 FTP traffic 109
5.2.1 Experimental set up 109
5.2.2 Causal model 109
5.2.3 Prediction 116
5.2.4 Concluding remarks 120
5.3 Lessons learned 121
5.3.1 Artificial dataset 121
5.3.2 Internet dataset 122
5.3.3 Methodology 122

6 Study of the DNS service 125

6.1 Problem statement . . . 125

6.2 Counterfactuals . . . 127

6.3 Experimental set up . . . 128

6.3.1 Experiment design . . . 128

6.3.2 Model parameters . . . 129

6.3.3 Observation summary . . . 129

6.4 Causal study of DNS impact on the throughput . . . 131

6.4.1 Modeling causal relationships . . . 131

6.4.2 Total causal effects study . . . 134

6.5 Concluding remarks . . . 142

6.6 Lessons learned . . . 143

6.6.1 Artificial dataset . . . 144

6.6.2 Data acquisition and pre processing . . . 144

6.6.3 Validating the method and then the data . . . 145

7 Conclusions 147

7.1 Objectives and achievements . . . 147

7.1.1 Progress . . . 147

7.1.2 Contributions . . . 150

7.2 Lessons learned . . . 153

8 Future work 155

8.1 Method . . . 155

8.1.1 Extension and improvement . . . 155

8.1.2 Usage . . . 157

8.2 Domain of application . . . 158

A Additional results and studies related to the independence tests 163

A.1 Comparison of the Z-Fisher criterion and KCI test . . . 163

A.2 KCI and categorical data . . . 167

B Technical details related to the DNS Study 173

B.1 Problematic . . . 173

B.2 Implementation . . . 175

B.2.1 Overview . . . 175

B.3 Practical issues . . . 175

B.3.1 Variable bin width histograms . . . 177

B.3.2 Fixed bin width . . . 178

B.3.3 Fixed bin width and high pass filter . . . 178

B.3.4 Open questions . . . 179

B.4 Artificial dataset . . . 179

B.4.1 Simulated dependencies . . . 179

B.4.2 Intervention prediction . . . 181

B.5 Ideal scenario . . . 181

B.5.1 Scenario A . . . 182

B.5.2 Scenario B . . . 184

B.5.3 Concluding remarks . . . 185

B.6 Removing samples in the tail of the distribution . . . 185

B.6.1 Concluding remarks . . . 187

B.7 Removing samples in zones away from the tail of the distribution 188

B.7.1 Concluding remarks . . . 189

B.8 Conclusion . . . 189

B.8.1 Results for the ideal scenario . . . 190

B.8.2 Removing from distribution tail . . . 191

B.8.3 Removing out of distribution tail . . . 191

C Parametric model of distributions 193

C.1 Rebmix . . . 193

C.1.1 Preprocessing stage . . . 194

C.1.2 Optimization stage . . . 194

C.2 Testing Rebmix package . . . 195

C.2.1 First test . . . 195

C.2.2 Second test 196
C.2.3 Appraisal 197
C.3 Parametric models and prediction 199
C.3.1 Modeling the throughput of the CDN users 199
C.4 Inferring the CDN throughput distribution with missing data 199
C.5 Concluding remarks 203

D Copulae **207**
D.1 Introduction 207
D.1.1 Probability integral transform 207
D.1.2 From modeling marginal dependencies to modeling multivariate distribution208
D.2 Copulae 208
D.2.1 Definition 208
D.2.2 Sklar theorem 209
D.2.3 Copula families 209

E Independence tests **213**
E.1 Existing works 213
E.1.1 Estimation of distributions 213
E.1.2 Discretization 213
E.1.3 Find functions separating X and Y 213
E.2 HSIC 214
E.2.1 Description 214
E.2.2 Test 215
E.3 KCI 217
E.3.1 Notations 217
E.3.2 Characterisation of independence 217
E.3.3 Kernel based independence test 218
E.3.4 Unconditional independence testing 220
E.3.5 Conditional independence testing 220
E.3.6 Approximating the null distribution with a Gamma distribution222
E.4 Differences 223
E.5 Conclusions 223

F Résumé **225**

F.1 Introduction 225

F.2 Une approche causale 226

F.2.1 Motivation 226

F.2.2 Inférence d'un modèle causal 227

F.2.3 Prédire l'effet d'une intervention 228

F.3 Réseaux de télécommunications 230

F.3.1 Le réseaux Internet 230

F.4 Adaptation d'une approche causale aux contraintes inhérentes à l'étude des performances d

F.4.1 Test d'indépendance 233

F.4.2 Estimation des distributions 234

F.5 Validation de notre approche à l'aide d'un émulateur 235

F.5.1 Scénario 235

F.5.2 Dataset 235

F.5.3 Modèle causal 236

F.5.4 Prédiction d'intervention 237

F.6 Études causales de systèmes réels 238

F.6.1 FTP 238

F.6.2 DNS et CDN 242

F.7 Conclusions et directions futures 249

F.7.1 Conclusions 249

F.7.2 Directions futures 251

G Glossary **253**

G.1 Acronyms and Abbreviations 253

List of Figures **258**

List of Tables **260**

List of Publications **261**

Bibliography **263**

Chapter 1

Introduction and background

1.1 Introduction

Our work focuses on the study of telecommunication networks. Its novelty comes from the approach used that is used, based on causality. Hence, this chapter presents the two domains that are the basis of our work: telecommunication networks and causality.

Telecommunication networks are complex and dynamic systems. Several constraints specific to the domain of telecommunication network make its study a difficult problem. Because of the complexity of telecommunication networks, many parameters need to be considered in a model that captures their performance. As we increase the number of parameters, we also increase the number of inter dependencies between the explanatory variables. Classical approaches rely on domain knowledge or on the detection of correlation between different events that, when observed together, could explain the observed performance. Unfortunately, in the domain of telecommunication networks the presence of spurious associations, where two parameters might appear correlated but in fact are not, is common. With the development of new technological solutions (for example LTE or Content Centric Networks) and of new usages of telecommunication networks (for example telemedicine), the domain of the telecommunication networks is changing very fast. The design of models, or approaches, that model the performance of a network performance based on the co occurrence of events becomes rapidly inaccurate and of little interest.

Telecommunication networks, because of the variety of their components (such as subnetworks, connected devices, services) cannot easily be emulated. The other approach that would consist in intervening on some components of an isolated subnetwork to observe the evolution of the subnetwork performance and draw conclusions regarding the adoption of these changes at a larger scale

is also complex. First, because it is hard to isolate a subpart of the Internet network that is representative of the Internet network as a whole. Second, because the clients of a telecommunication network expect (and often pay for) a quality of service that cannot generally be degraded for experimental purposes.

Because of these constraints, the study of telecommunication networks is complex and a special care is to be taken when tackling the problematic of modeling their performance. The solution proposed in our work is to design an automatic approach that exhibits the structural dependencies between the different components that form a telecommunication network and the parameters that influence its performance. Our approach relies on the principles of ***Causality*** and the representation of causal dependencies with graphical models, namely ***Bayesian networks***.

The approach we propose does not require active measurements and, based on passive observations exclusively, allows to predict the performance of a given application (such as HTTP), or service (such as DNS), if we modify the behavior of the elements of the network or the parameter of the application being studied. Coupling the formality of a causal approach and the representation of causal dependencies using Bayesian networks allows to greatly simplify the study of systems as complex as telecommunication networks.

However, the application of a causal approach to the domain of telecommunication networks is not necessarily straightforward. While the theory we use in our work relies on very few assumptions, its application to real case study requires to adapt to the constraints of telecommunication networks the different implementation of the different methods used in causal inference. It is therefore at the heart of our work to correctly assess the different assumptions, or absence of assumption, that the application of the causal theory to our problem requires and why telecommunication networks differ from the other domains where a causal approach is usually used, such as social sciences, biology or economical sciences.

In order to understand the specificity of telecommunication networks, the different constraints inherent to their study and the reason why a causal approach is of great value, we first give, in the following section of this chapter, general definitions and explanations of telecommunication networks. We present the different components that form a telecommunication network and how they interact together. We also describe most of the parameters that often define the performance of a given service running on the Internet network. These parameters will be used in the different studies presented in this thesis.

In a second part of this chapter, we present the theory of causality, focusing on the approach and definitions that will be used in our work. We also explain why a causal approach is not necessarily straightforward and why a special care needs to be taken when trying to apply an existing theory to a domain it was not designed for. This part will be the base of many of the methodological

choices we make in our work. Our approach to the study of telecommunication networks performance using a causal approach is presented in Chapter 3.

In the last part of this chapter, we come back on the different properties of telecommunication networks and causality and, with the different properties that are presented in the first two sections, we review the benefits of our approach and the different considerations to keep in mind when designing the causal study of telecommunication networks.

1.2 Telecommunication networks

In the following sections, we define telecommunication networks and we present some of their properties illustrated through different examples. We further focus on a special case of telecommunication networks that will be at the center of our work, namely the Internet network.

1.2.1 Protocols

Telecommunication networks operate according to pre defined rules called protocols. Protocols allow the different actors of a telecommunication network system to work together and to provide services to its users. There exists an important number of protocols supporting the Internet network, as there exists different services provided by the Internet network or supporting the Internet network. In our work, we focus on end user performance. We are mainly interested in Internet based applications such as the File Transfer Protocol (FTP), Hyper Text Transfer Protocol (HTTP) and so on. Such applications rely on the Internet protocol suite that is presented in Section 1.2.4. In this section, we also present one of the most important protocols of the Internet protocol suite, called the Transmission Control Protocol (TCP). TCP is the protocol that will be mainly studied in our work.

For brevity reasons, we only present the different concepts ruling the Internet and TCP protocol. We refer the readers to the book from Kurose and Ross [KR09] for deeper explanations of the different concepts presented or mentioned in this section.

1.2.2 Presentation

A telecommunication network is a system formed by a set of nodes and links that allows two entities using this system to communicate, see Figure 1.1. There exists an important variety of telecommunication networks, defined by the type of intermediate nodes (base station, router, no intermediate node,...),

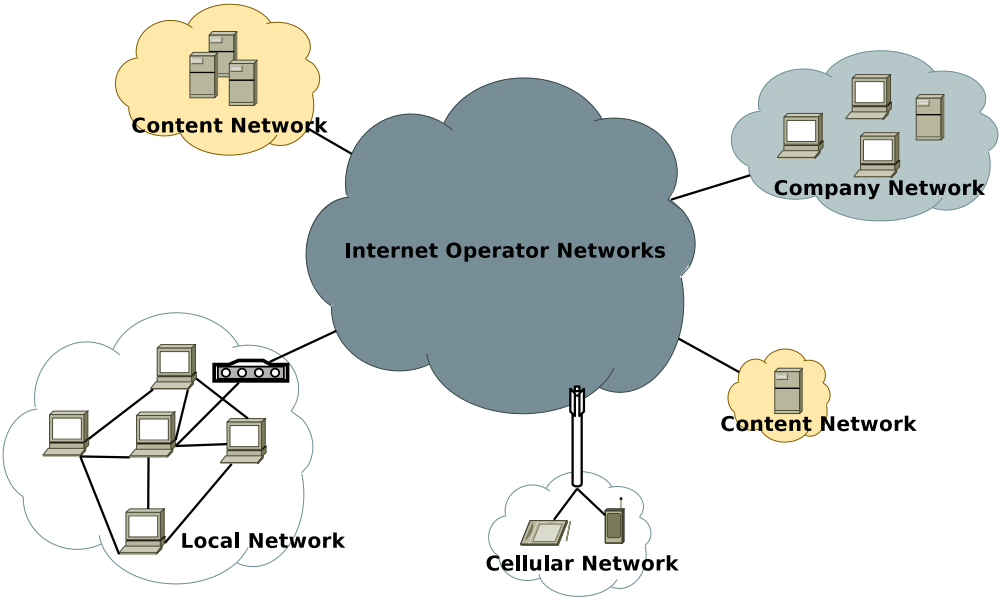


Figure 1.1: Example, from a high level point of view, of a communication network

the type of links (wired, wireless, hybrid), the technology (Ethernet, Bluetooth, WiFi) or the communication protocols (SONET/SDH, IEEE 802, Internet protocols) forming the network. In our work, we focus on the Internet network. The Internet network is formed by interconnected computer networks whose communications are based on the Internet protocol suite.

The Internet network is a Packet Switched Network (PSN). Therefore, when two Internet users want to communicate, the resources are not allocated when the communication channel is created and freed at the end of the communication. Instead, the Internet offers a Datagram Packet Switching service, where each packet is processed individually by the different routers between the source and the destination. In a Datagram Packet Switching service there is no guarantee that different packets belonging to the same connection cross the same physical links. Each packet carries information that identifies the connection it belongs to and allows the different routers to forward the packet to the next hop (router) towards the destination of the packet, i.e. one end of the connection. One of the objective of the Internet protocol suite is to provide an abstraction of the underlying computer networks that form the Internet network. The Internet protocol suite allows two Internet users to benefit from a reliable communication channel despite an underlying network of unreliable computer networks.

1.2.3 Internet

ARPANET was a project funded by the Advance Research Project Agency (ARPA) that started in the 1960. Advance Research Project Agency Network (ARPANET) was a network initially designed to connect a limited number of universities in U.S., mainly for a research purpose. While other networks were being developed at the same time, like the X.25. networks in Europe, the ARPANET opened itself to more universities and eventually took over the alternative telecommunication network solutions to become the most used telecommunication network system. The development of the TCP/IP suite, and its standardization in the 1980s, gave birth to the concept of the world wide interconnected networks, forming itself a network called Internet. Hence, Internet is a network connecting networks together world wide in order to provide connection between any user of any network connected to the Internet.

1.2.4 TCP/IP

The Internet network relies on a suite of protocols that allows a set of machines to communicate between themselves, from any location to any location, as soon as they are connected to a network that is part of the Internet network.

To reduce the complexity, most network architectures are organized as a series of layers, each one built on top of its predecessor. Each layer performs a subset of functions required to communicate with another system. It relies on the next lower layer to perform more primitive functions and to conceal the details of these functions. It provides services to the next higher layer. Ideally, the layers should be defined in such a way that changes to one layer do not require changes to the other layers.

The Internet protocol suite is often denoted by TCP/IP, the two main protocols of the suite. The Internet protocol suite consists of four layers, namely the ***Link layer***, the ***Internet layer***, the ***Transport layer*** and the ***Application layer***, Figure 1.2[1]. Each layer is defined by a set of services that it has to provide to the upper layer and a set of services that it receives from the lower layer.

- The Link layer focuses on the local network connections, its main purpose is to transmit a packet along a given link (one hop transmission). The protocols defined in the Link Layer are hardware dependent. In the link layer packets are called ***frames***.

[1] It should be mentioned that there exists another model of the Internet protocol suite, the OSI model, that contains seven layers: the physical layer, the link layer, the network layer, the transport layer, the session layer, the presentation layer and the application layer, illustrated in Figure 1.2

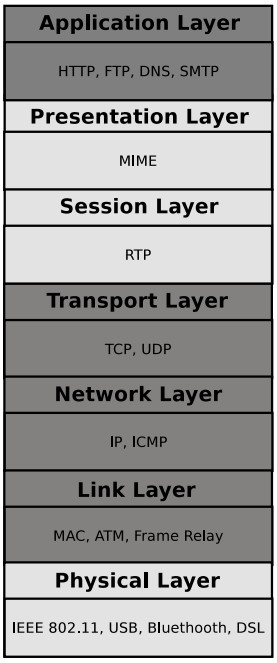


Figure 1.2: OSI stack with examples of protocols in the different layers: Hyper Text Transfer Protocol (HTTP), File Transfer Protocol (FTP), Domain Name System (DNS), Simple Mail Transfer Protocol (SMTP), Multi purpose Internet Mail Extension (MIME), Real Time Protocol (RTP), Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Internet Protocol (IP), Internet Control Message Protocol (ICMP), Median Access Control (MAC), Asynchronous Transfer Mode (ATM), Frame Relay, IEEE 802.11, Universal Serial Bus (USB), Bluetooth, Digital Subscriber Line (DSL). In darker, the layers from the TCP/IP suite, where the two layers Presentation and Session are included in the Application Layer and the Physical layer is not part of the TCP/IP model.

- The Network layer focuses on the global network connections, its main purpose is to transport packets over a set of interconnected networks. The most known protocol within the Network layer is the Internet Protocol (IP) also responsible of defining the addressing of the Internet entities. In the Network layer, packets are called ***datagrams***.

- The Transport layer focuses on the connections at process level, its main purpose lies in the abstraction of the underlying networks crossed by the different packets. The two main protocols of the Transport layer are the User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP). In the transport layer, packets are called ***segments***.

- The Application layer focuses on the protocols used by the applications developed by the end users. Example of famous application layer protocols are HTTP (web browsing), FTP (File transfer) or the Simple Mail Transfer Protocol (SMTP).

In the following, we will discuss in-depth protocol functions such as error control, connection management or flow control. We will discuss each protocol function and explain how this protocol function is realized in the TCP/IP protocol architecture.

In our work we focus on the traffic carried by the TCP. Eighty percent of the Internet traffic consists in TCP traffic. TCP is a complex protocol that implements a connection-oriented transmission service. TCP ensures reliable end-to-end transmission, in particular TCP implements connection management, error recovery and flow control. A specificity of TCP over other protocols, such as UDP for example, comes from its ability to adapt the packet sending rate to the estimated congestion of the network. As we are working exclusively with TCP we will use the term *packet* to designate transport layer packets, instead of *segments*.

1.2.5 TCP

It is important to notice that the presentation of the protocol functions, and their realization by the Transmission Control Protocol (TCP), that we give in this section is neither complete nor exhaustive. The protocol functions of error control, connection management and flow control are complex and some details will be voluntarily omitted to keep the presentation both, clear and simple. On the other hand, the specifications of the TCP mechanisms controlling the sending rate, and the implementation of these mechanisms are out of the scope of this thesis. The authors interested in having more details of the TCP can refer to RFCs 675 and RFC 793, for early versions of TCP, or RFCs 5681, 7332, 6247 and 6298 for more recent versions of TCP. There exist many versions of the TCP protocol, we only present some of them and this section aims to present the main protocol functions and their implementation in TCP.

The presentation and understanding of TCP mechanisms are important as they represent the knowledge that will be necessary to understand the models presented in the studies of the Dropbox application, FTP application and CDN throughput that we present in Chapter 4, 5 and 6 respectively.

TCP connection management, error control and flow control

At the Network layer, Figure 1.2, the protocols are not concerned by the underlying network capacity. Their only purpose is to ensure that packets are forwarded from one link to another. If, at a certain point of the network, the incoming traffic is bigger than the capacity of the outgoing links, the network at this point becomes congested. In case of congestion, packets can be delayed or, in case of resource shortage, dropped. Usually buffers are used to cope

with this eventuality and smooth the traffic to optimize the network usage and improve its performance. However, if a buffer becomes full, it cannot store any more packets and some packets are dropped. Therefore, on its way from the sender to the receiver a packet can be lost, corrupted, duplicated or disordered (it arrives at the receiver before its predecessor or after its successor).

TCP does reliable in sequence delivery and adjusts to the network state (congestion). TCP implements ***connection management***, ***error control*** and ***flow control*** and uses: ***packet sequence number***, ***timer***, ***acknowledgment buffer***and ***receiver window***.

Each time an Internet end user, the sender, wants to transmit an information to another Internet end-user, the receiver, the sender first contacts the receiver and opens a ***connection***. The information that the sender wants to transmit is then divided into fixed length packets that are transmitted along the different networks connecting the two users. Each packet that is sent has a field that contains a unique identifier that consists of its sequence number and its connection identifier. The first time a packet is sent the sequence number field is set to a (pseudo) random value and for each *new* packet sent this field is incremented by one. Note that the sequence number is not incremented in case of a retransmission after a packet gets lost (e.g. dropped by router).

For each packet correctly received by the receiver, a special packet called ***acknowledgment*** is sent by the receiver to the sender. The acknowledgment is a packet sent by the receiver to the sender that acknowledges the reception of a packet sent by the sender according to its sequence number field value. It is important to notice that, in most versions of TCP, positive acknowledgments are used. The receiver can only acknowledge the reception of a packet (not its loss). Additionally, it is common to use cumulative acknowledgments, where the receiver sends in its acknowledgment the information that all the sequence of packets up to the number present in its acknowledgment have been correctly received.

Connection management Connection management involves setting up and tearing down connections. Connection management is a necessary building block in every protocol that provides a reliable data transfer. It ensures

- **at-most-once delivery:** each unit of information sends by the sender to the receiver gets accepted only once by the receiver
- **graceful close:** A connection is not closed by the sender or the receiver before all the information that the sender has to transmit is correctly received and acknowledged by the receiver

By keeping track of the packets that have been sent and correctly received, it is possible to ensure at-most-once deliver. For the graceful close, a combination

of a three way handshake and a timer ensures that both parties have agreed to close the connection and that no more information is to be sent.

The complexity of managing a connection comes from (i) Crash: The sender or receiver may crash and loose state information (ii) Size: the identifiers (connections and packets) need to be re used and the communicating entities cannot store all the packets of all the connections in which they are involved.

To solve (i), a crash counter can be used as part of the connection identifier and a timer can be set to ensure that duplicates will not be received.

To solve (ii), it is necessary to ensure safe re use of past identifiers. Regarding sequence numbers the sequence number field of a TCP packet is 32 bits that allows safe re use. For the connection identifier, a special care is to be taken on the maximum ***lifetime*** of a connection identifier. The lifetime of ***connection records*** (information relative to a given connection stored by the connection management protocol) is used to release connection records after a finite amount of time.

Error control The transmission of data can be subject to errors. In this section, we describe the general principles of error control. Error control comprises error detection and error recovery. We distinguish two types of errors:

- **Corruptions** of data, for instance, due to due to noise, crosstalk, or inter symbol interference. Corruptions can be detected using techniques such as parity bits or cyclic redundancy checks. To detect these two kinds of corruptions, it is necessary to perform an end-to-end error detection at the transport layer.
- **Loss** of data, for instance due to buffer overflow in a switching node or an end system. To detect the loss in an unambiguous way, all data transmitted must be identified with a sequence number.

In TCP, using the checksum field of the TCP packet header, the receiver can detect corrupted data and discard it. Discarding corrupted messages is called ***hard error detection***.

With the use of sequence number, TCP can detect duplicates and can correct re orderings at the receiver side. In addition, using the estimate of a packet Round Trip Time, the TCP sender can use timers to detect losses. To obtain an estimate of the packet round trip time, at the sender side, TCP can measure the time between the transmission of a packet and the reception of its corresponding acknowledgment.

There are mainly two ways for the sender to detect a loss.

One way is with the use of timers. Each time a packet is sent, the sender starts a timer whose value depends on the different RTT values estimated so far. If no acknowledgment is received at the end of the timer, TCP triggers a ***time out*** event and the corresponding packet is considered as lost and is re transmitted.

The second way uses (usually implicit) information from the receiver. Some TCP versions make use of selected acknowledgments some do not, but the receiver can implicitly communicate to the sender that it is still waiting for a missing packet with the use of acknowledgments. After receiving such information, the sender can consider that the packet has been lost and decide to re transmit it.

Congestion control

In the event of users sending more packets that the network can transmit, packets will be dropped. If the senders keep sending packets and retransmitting dropped packet (= lost packets), the network will eventually collapse. Congestion collapse was first observed in 1986 when a backbone in U.S. (the NFSNET backbone) started dropping three order of magnitude from its capacity. To avoid the situations of congestion collapse, congestion control was implemented in the end nodes.

To control the amount of traffic that can be safely handled by the network, TCP defined a sender internal parameter called the ***congestion window***. At any time during the communication between two entities over the Internet, the sender sending rate is limited by the congestion window size value. The congestion window size is initially set to an empirical value and each time a packet is successfully transmitted to the receiver (that is, the sender had received the acknowledgment corresponding to the packet that has been sent), the congestion window size value is increased. On the opposite, each time a loss is detected at the sender side, the congestion window size value is decreased. Note that the speed at which the congestion window increases is dictated by the frequency at which the sender receives acknowledgments. The time between the submission of packet and the reception of its corresponding acknowledgment is the RTT and is influenced by the load of the network(s) on the path between the sender and the receiver.

Flow control

The last important parameter to discuss is the ***receiver window***. Packets arriving at the receiver can be disordered. The receiver can re order packets using the sequence numbers of the TCP packets. However, to do so, the receiver needs to buffer the packets it receives, inspect them and, if necessary, re order them before passing them to the upper layer (cf TCP/IP suite, Figure 1.2).

To inform the sender about its available buffer and limit the sending rate from the receiver side, one of the fields of the TCP packet contains the value of a parameter called the *receiver window*. In each TCP packet sent by the receiver to the sender (for example acknowledgments), the receiver informs the sender about the memory resource that is available to receive more data. Through this mechanism, the receiver ensures that no packet will be lost due resource shortage on its side and it can adjust the sending rate according to its needs or resources.

Concluding remark for TCP congestion control

There exist other mechanisms to prevent congestion, some routers can explicitly inform senders of the presence of congestion for example. However, such mechanisms are not always supported and are not considered in our work. What is important to notice is that, at any moment during a connection, the sending rate is defined by the minimum of three factors: the congestion window, the receiver window and the amount of data present in the sender buffer, awaiting to be sent. Figure 1.3 presents a simplified vision of the different TCP sender windows in the case where the receiver window is limiting the sending rate. Notice that this scenario is voluntarily simplified, we reason based on packets (not bytes), we present very little quantities of information and consider a small period of time.

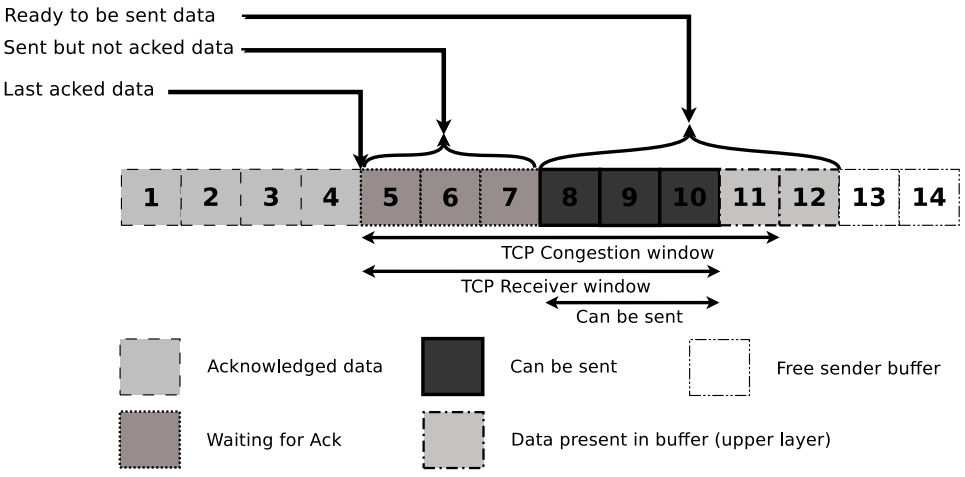


Figure 1.3: Different windows at the TCP sender

In Figure 1.3, we represent a scenario where an application needs to transmit data over the Internet network. The application relies on TCP to safely send data to another entity across the Internet. The TCP sender has a buffer to store packets that need to be sent or wait for their acknowledgments that corresponds to 14 packets. In the scenario represented in Figure 1.3, the sender already

sent four packets (1,2,3,4) and received their corresponding acknowledgments, ensuring that they were correctly received by the receiver. The TCP sender also sent three additional packet (5,6,7) that are not acknowledged yet by the receiver. We also suppose that the initial value of the congestion window and the correct transmission of the first three packets have allowed the congestion window (*cwnd*) to reach a value corresponding to seven packets. On the other hand, the receiver informed the sender that the resources that it allocated for their TCP connection (*rcvwnd*) corresponds to six packets. As three packets have already been sent, the number of packets that can be sent in the scenario observed in Figure 1.3 is (*swnd*):

$$swnd = min(cwnd, rcvwnd) - SentNotAckd = min(7, 6) - 3 = 3,$$

so the sender can send packets 8, 9 and 10.

Let us assume that, upon receiving the acknowledgment of packet 5, the congestion window is incremented by 1 (*cwnd* = 8) but the receiver window keeps its value of 6, Figure 1.4. Therefore *swnd* is:

$$swnd = min(cwnd, rcvwnd) - \text{SentNotAckd} = min(8, 6) - 5 = 1,$$

and the packet 11 can be sent.

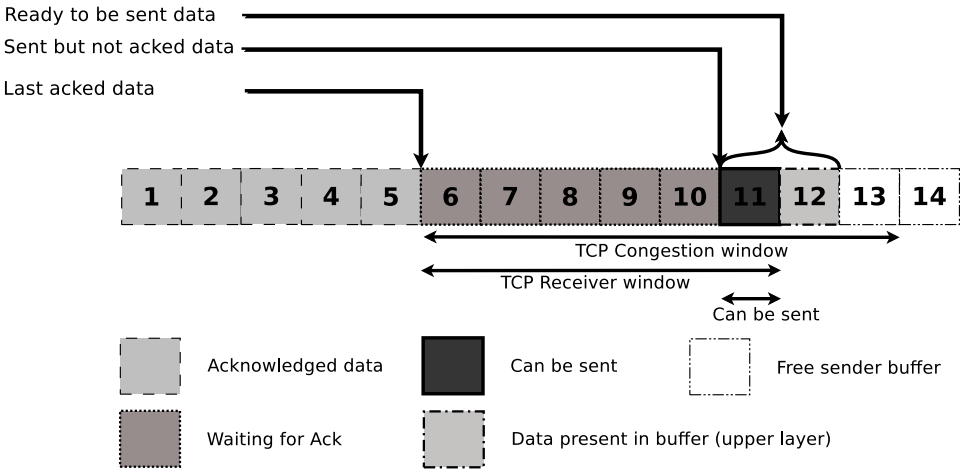


Figure 1.4: Different windows at the TCP sender upon reception of ack

There exist different versions of the TCP protocol, among which the Tahoe version, the Reno version, the Vegas version, the new Reno or the Cubic version.

The different TCP versions vary in their implementations of the congestion avoidance algorithm. For example, *TCP Reno* improves *TCP Tahoe* by reacting to the reception of three duplicated acknowledgments and implements what is know as *Fast Recovery* to react better and faster to a congestion event. *TCP Vegas* improved the estimation of the optimal time out timer value. *TCP new Reno*

improves the performance of TCP in case of a congestion event detected by the reception of three duplicated acknowledgments, using the implicit information that information is still passing through the network if acknowledgments are received from the receiver. Due to the changes in networks with higher speed transfers, and in order to decrease the discrimination of connections with high latency, *TCP Cubic* implements a function defining the evolution of the congestion window as a cubic function of the time elapsed since the last loss event. *TCP Cubic* is the default version used in the Linux kernels 2.6.19. and above.

TCP specifications vs TCP implementation

TCP is an important protocol for the well functioning of the Internet network. All the machines connected to the Internet network implement TCP. However, due to the diversity of the Internet connected devices (from a heat sensor to a super computer), each machine implements its own version of the TCP protocol and uses its own parametrization. All the machines implementing TCP, have to comply with the specification of TCP, in order to be able to communicate one with the other (such as the TCP header) but the implementation of functions such as the one of congestion avoidance can be different and parameterized: the initial congestion window value, the increase of the congestion window value upon the reception of a valid acknowledgment, the computation of the time out value, and so on. Therefore there is no such thing as one TCP version running on all Internet machines but many versions with many parameterizations, answering to different needs and constraints. As we saw, the TCP Cubic version was designed to offer better performance in what is known as Long Fat Network (LFN) (high bandwidth and high latency). However, with the increase of mobile end users, LFN might not be the best network model for mobile users yet. While each platform (Android (Google), Windows (Microsoft), iOS (Apple)) can run its own TCP protocol, they also differ in some parameterization that their TCP versions implement.

Therefore, one should be careful in the distinction between TCP as a specification that defines rules that allows two entities to communicate and TCP as an implementation in the operating system of the machine being considered. The TCP specification defines the packet format (different header fields), while its implementation in a given machine includes algorithm, implementation choices and parameter values that are specific to the machine.

1.2.6 Specificity of the Internet

In this section we present some of the important characteristics of the Internet network that will support or constrain its study with a Causal approach as presented in Chapter 3.

The Internet has grown very fast since its creation in the 1980s as a world wide network of interconnected networks. According to the International Telecommunication Union (ITU) 44% of the households in the world have an Internet access in 2014 and more than 40% of the world uses Internet daily. For the developed countries these two percentages are close to 80%. Consequently, these numbers are very likely to increase in the coming years with the different projects led by societies like Google or Facebook to improve the connectivity for developing countries. According to ITU the percentage of people using Internet has doubled in the last seven years.

The deployment and adoption of the Internet network has been supported by the development of several access technologies (ADSL, Mobile, Fiber, Satellite), the multiplication of actors participating in its development (ISP, Content provider, Online service companies, Developers) and in its growth. Most importantly, an important number of applications, services and usages exist and is being created to help Internet network users in their daily life, with new applications being designed every day.

Technologies

In this section, we briefly present some of the common technologies used by the Internet users to access Internet. The aim of this section is not to give an exhaustive list but, through the presentation of some of the common technologies supporting the Internet, to show the heterogeneity of the Internet technologies and of the models of user Internet access.

Along with the Internet expansion, the number of technologies to access the Internet has also grown very fast and heterogeneous. For wired connections, European countries, like France, often use Asynchronous Digital Suscriber Line (ADSL) while in U.S. there exists a common alternative based on television service infrastructure called Cable Internet access. In the past years, we have seen the different Internet Service Providers (ISPs) proposing to their customers Fiber To The Home (FTTH) accesses that allow higher transfer speed. Alternative solutions such as power-line Internet also exist that allow access to the Internet in areas where the installation of new equipment is often complicated. Most companies and universities use Local Area Network (LAN) solutions where Ethernet is the most spread wired access technologies.

In addition to wired access, there exist several wireless technologies a very common one being the WiFi (802.11) technology. However, with the advent of smart phones, mobile broadband access has become the technology most commonly used to access the Internet network. According to ITU 2014 report 84% of the people living in a developed countries has an active mobile broadband subscription and 21% for the developing countries. According to *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2014 – 2019*,

the global mobile data traffic grew 69% in 2014. The comScore company™ announced in February 2014 that mobile telephone usage overtook laptop in Internet usage. Finally, there are cases where remote areas rely on Satellite broadband access to access Internet. This solution is known to be expensive and implies important delays.

Actors

The understanding of all the actors participating in the existence, diffusion, evolution and usage of the Internet network is out of the topic of our work. Internet evolution is guided by the rule of supply and demand in terms of services and contents. There should exist more economical agreements than there exist telecommunication operators, all these agreements define service level agreements (SLA) between the different parties and impact the quality of service of the end users. Obviously, the evolution of the Internet network is also supported by the evolution of the different technologies supporting its communication (Physical links: Fiber, Broadband access, WiMax, Internet nodes: Routers, Servers, Super computers, End user devices: Tablets, Connected Devices, Cars manufacturers). On the other hand, in addition to the "connectivity providers´´, there is an increasing number of content, or service, providers developing new solutions for all the use cases that one could need or think of (Home automation, Cloud Computing, Online Gaming, Video On Demand, Social Networks, News,...). It is then not possible to list all the actors that, altogether, form the Internet network in its global definition.

However, it is important to notice that from the Internet Service Providers to the Online Content providers there is an important number of entities working together to provide to end users an access to the different resources available on the Internet. Each of these actors plays a role in the Quality of Experience (QoE) of the end users. It is the ability and quality of the interactions of these actors that defines the end user QoE. Understanding how the service of an actor of the network impacts the service of another and finally the end user experience is a complex, if not impossible, problem that will be discussed later in this chapter.

Usage

There is a large number of services based on the Internet network. Some of the most common services fall in the following categories: Mail, File exchange, Web browsing, or Video Streaming. However, the Internet users being more and more numerous, new categories of services appear very often. An example is the paradigm of the cloud computing (lie Amazon Web Service) that has

seen its evolution and adoption boosted by the distributed databases and the computing power of computer clusters.

It is interesting to notice that different usages of the Internet network require different qualities of service. For example telephony (Voice Over IP) is more sensitive to delay and delay variations than it is to packet losses. On the opposite, while a user is concerned by the delay while accessing a file, a special care is taken in file transfers to recover from packet losses.

The point that we try to stress here is the increasing number of services developed in the Internet network, each of which having different requirements in terms of performance and reacting differently to network failures or malfunctioning.

1.3 Causality

In the previous section, we presented Telecommunication networks. In this section, we present the methodology that will be used in this thesis for the study of different systems relying on telecommunication network services. The aim of our work is to adopt a causal approach to study different telecommunication network applications. Therefore, this section gives a brief introduction to causality, its principles and the causal models, more particularly the representation of a causal model as a Bayesian network.

1.3.1 Correlation and causality

There is a recurrent question regarding correlation when dealing with causality: What is the difference between correlation and causality ? Why do we need causality ?

Correlation is not causation, the detection of correlation between two parameters does not identify with enough accuracy the relationship that exists between these two parameters. One can impact the other, or the opposite, or an unobserved (set of) parameter(s) can impact these two parameters simultaneously. The difference between correlation and causation plays an important role if we want to use this knowledge to decide on how to improve our system by partly modifying its behavior. A causal approach tries to uncover the structural dependencies between the different parameter forming the system being studied. Structural dependencies allow to predict the behavior of a given system when we modify the behavior of its parameters with manual interventions. The prediction of the effect of manipulating the parameters of a given system is an important strength of causal models as they are *stable under intervention*. The stability under intervention means that a causal model, inferred from the observations of a system in a given situation, is still valid if we manually change the

system mechanisms, redefining the systems laws. The manual modification of a system parameters is called an ***intervention***. Interventions consist in modifying the behavior of a component of the system, removing the influence from its direct and remote causes, and manually fixing its variations. Causal models and the causal theory [Pea09, SGS01], allow predicting the distributions of the different parameters of a system after an intervention where we modify the system mechanisms. The strength of this approach comes from its usage of passive observations made prior to this intervention exclusively. Our approach does not require additional measurements or active experiments.

Our usage of a causal approach tries to solve the important limitation of a correlation-based approach that is the risk to expose spurious associations between parameters. A spurious association is an association that shows two parameters as dependent but ignores the mechanisms behind this dependence. The use of passive observation does not allow us to learn the behavior of the system if we modify its parameters based on such association. A spurious association is often due to an unobserved mechanism that misses the presence of a given parameter to correctly explain the correlation we observe between two parameters of the system being studied. A naive example would be if someone observes that car accidents happen more often when windshield wipers are on. The correlation of these two events would be a spurious association as it is actually the rain condition that affects both events simultaneously.

1.3.2 Interventions and counterfactuals

Causality can be understood with two different concepts that will be both exploited in our work: ***Intervention*** and ***counterfactuals***.

Intervention

If we observe a system with a set of parameters $\{p_1, \ldots, p_n\}$ and we are interested in the causal dependence between the parameter p_i and p_j, one experimental solution to study their causal dependence is to manually force the parameter p_i to take the value $v_{i,x}$ and observe the changes that happen on p_j. The changes observed for p_j can only be due to the modification of the parameter p_i as the rest of the system was not modified, therefore highlighting a causal dependence between p_i and p_j. This approach is difficult to use in practice as it requires:
− To be able to manually intervene on the system;
− To ensure that the other parameters are not modified.
This concept helps to understand the principle of a causal dependence between two parameters but often it cannot often used in practice.

However, the notions of intervention and causality are linked. It is the relationship between causality and intervention that motivated our work. The estimation of the causal effect of one variable on another allows predicting the effect of intervening on this variable. If X has a causal influence on Y then we know that intervening on X will modify the behavior of Y. We will see in Section 1.3.4 how to use causal dependencies and causal models to predict the effect of interventions.

Counterfactual

A second vision of causality is given by the study of counterfactuals. "*If I observe the parameter X taking the value x, and the parameter Y taking the value y, what would have been the value of Y had X taken the value x'?*". To answer such question it is necessary to know the structural dependencies between X and Y. Counterfactuals represent a approach for understanding causality more than a method to infer causal dependencies between different parameters. In the domain of artificial intelligence where studies are made to teach machine to learn causal dependencies, this vision of causal dependencies is at the center of many researches [Pea09, Chapter 7 & 8].

As for the case of intervention, it is the use of causal model to answer counterfactuals question that interests us. We know that, if Y is causally dependent on X, and we observe $X = x$ and $Y = y$ we might be able to predict the value of Y had X taken a value different from x.

We will use the concept of counterfactuals in to study the impact of the Domain Name Service (DNS) on the performance of the users of Content Distribution Networks in Chapter 6.

Randomization

Finally, a last point to mention, as one of the few techniques used in practice to study causal relationships experimentally, is the one of *random experiments*. Random experiments are often used in medical studies where the effect of a drug on a given disease is being studied. The drug is tested on a group of people suffering from the given disease. A random experiment would consist in randomly dividing the group of people in two sub groups, G_1 and G_2, and administering to one group, G_1, the tested drug and a placebo to the other group, G_2. The difference in recovery between the people of the group G_1 and the people of the group G_2 can be then assimilated to the causal effect of the drug on the disease, as the two group were built in a random manner and the difference can only be attributed to effect of the drug.

In many situations, however, one cannot run active experiments. This is the case for telecommunication networks where experiments are difficult to set up.

We will come back on the difficulty of experiments and simulations in telecommunication networks later in our work. Our problematic is defined by the causal study of systems based only on ***passive observations***.

1.3.3 Definition of causal operators

An important limitation to the adoption and utilization of causality in many domains comes from the absence of an adequate mathematical grammar to express causality. Take as an example the famous Ohm law,:

$$I = \frac{V}{R}. \tag{1.1}$$

This equation means that a potential difference, V, across two point A and B of a conductor causes the apparition of a current I whose intensity is inversely proportional to a constant R.

However, using equations like Equation (1.1) does not convey any causal dependence. An important missing notion being the one of direction. In a mathematical equality such as the one presented in Equation (1.1) variables are allowed to be "moved" from one side of the equality to the other. This symmetry is not valid in a causal relationship. For this reason a causal relationship between X and Y requires additional care in its representation.

Early representations of causal relationships relied on the use of Structural Equation Model (SEM) [Wri21, Haa43, Dun75], where X causes Y is represented as:

$$Y := f(X) + \varepsilon, \tag{1.2}$$

illustrating the asymmetry of the relationship between X and Y.

As mentioned previously, our interest in causal dependencies lies in their stability under intervention. More precisely, the causal relationship between a parameter X and a parameter Y will still exist if we intervene on X. An intervention on X would make X independent from its direct and remote causes and fix its value to x, independently of the other parameter values. We call this action an ***atomic intervention*** and we represent it with the introduction of the ***do operator***. The atomic intervention consisting in intervening on X to fix its value to x is denoted by $do(X = x)$, or simply $do(x)$. If a system is defined by three parameters, X, Y, and Z and the following causal relationships:

$$\begin{aligned} X &:= g(Z) + \varepsilon_X \\ Y &:= h(X, Z) + \varepsilon_Y, \end{aligned} \tag{1.3}$$

which means that Z causes X and X and Z cause Y, the atomic intervention $do(X = x)$ would result in the following equations

$$\begin{aligned} X &:= x \\ Y &:= h(x, Z) + \varepsilon_Y, \end{aligned} \tag{1.4}$$

where X is replaced by the value at which the intervention fixed the variable, x.

1.3.4 Bayesian networks and causal models

In the previous sections we gave an introduction to the notion of causal dependencies and causal effects. There exist several possibilities to represent causal dependencies. We mentioned before the structural equation models that were among the first causal models. Instead, in our work, we use Bayesian networks. Graphical models are often more intuitive when many parameters and inter dependencies need to be considered. In this section we will present Bayesian networks and expose the theory that motivated the use of this model to represent causal dependencies. Different theorems based on graphical criteria are presented, and show the simplification of the problem of causal inference when Bayesian networks are used to model causal dependencies.

Bayesian networks

Bayesian networks [Dar09] are graphs consisting of a set of vertices and directed edges that represent conditional dependencies between a set of parameters. Let us suppose that we want to represent our system, s, corresponding to a set of observed parameters, $\{X_i\}_{i\in[1,N]}$, with a Bayesian network. We create one vertex, V_i, for each parameter X_i and a directed edge, $E_{i,j}$, between V_i and V_j (denoted $V_i \rightarrow V_j$) if the parameter X_j depends on the parameter X_i. The graph we obtain is a Bayesian network representing the dependencies between the parameters of our system s. Figure 1.5 presents the Bayesian network that represents the causal model of the variables X, Y and Z represented by the structural equation model of Equation (1.3). One possible way to represent an atomic intervention on X is given on Figure 1.6, where X is not influenced anymore by its direct (or remote) causes and its value is defined by the intervention, that can be seen as the new and unique parent of X.

In our work we are interested in Directed Acyclic Graph (DAG). DAGs are graphs where all the edges are oriented and that contain no cycle. However, as we will see in Section 1.3.4 where we present the algorithms we use for causal model inference, we will sometimes use a Partial Ancestral Graph (PAG). PAGs are partially oriented graphs, graphs where some of the edges are left unoriented (V_i — V_j).

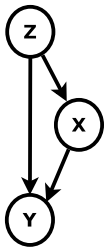


Figure 1.5: Example of a Bayesian network

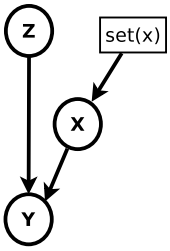


Figure 1.6: Bayesian network representing an intervention on X

Causal model

Properties There exist several possibilities to model causal dependencies (Bayesian networks, Structural Equation Model are two common models). In our work, we are interested in the use of a Bayesian network to model the causal dependencies between the parameters of the system that is studied. In this case, an edge between two nodes represents a causal dependency between the two corresponding parameters.

Bayesian networks provide a very concise and intuitive way to view the different relationships between the different parameters of a system. One of the most interesting properties of causal models comes from their stability under intervention. As mentioned previously, the causal dependencies between the parameters of a system are still valid if we modify the behavior of this system. Important works have been done [Pea09, SGS01] to simplify the causal study of a system by using its representation as a Bayesian network. In Section 1.3.4, we discuss the Bayesian network representation of a causal model. We also present some graphical criteria used in our work to predict the effect of an intervention on a system using data that was collected prior to this intervention, without the need to perform additional experiments or observations.

The inference of the Bayesian network representing the causal dependencies between the parameters of the system that we want to model has been one of the biggest challenges of our work. Many algorithms rely on assumptions that are not verified in the domain of telecommunication networks. We believe that

the absence of the assumptions commonly made by these algorithms is not specific to the domain of telecommunication networks. Hence, the difficulties first faced and then solved should be of interest to a broad range of domains.

In the following paragraphs we present the different steps of the algorithms used to infer the causal model of a given system when we limit ourselves to passive observations exclusively. As shown in our work, it is very important to understand each of the methods being used and the assumption on which they rely to be able, if necessary, to validate them. In the event of a method relying on assumptions that are not verified for the domain on which we intend to apply it, it is necessary to whether mitigate these assumption or revise such method.

Inference Inference algorithms for graphical causal models can be divided roughly into two categories: The score-based algorithms and the constraint-based algorithms. The score-based algorithms solve an optimization problem: The graph is built step by step and the progression is dictated by the optimization of an objective function matching the actual graph with the input data that represent our observations of the system. There has been an important progress in recent years in "*exact Bayesian network inference*". Using the score-based approach, the exact posterior probability of subnetworks (edges, Markov blanket, ...) is computed to search for the best Bayesian network fitting the input data. [Koi06, KS04] propose an algorithm with lower than super-exponential complexity in the number of parameters of the system. These algorithms always converge and allow its process to stop at any moment and obtain a graph with a lower bound on its accuracy. However, in these algorithms, ***completeness*** is assumed (all the variables are observed) and the parameters must follow a normal distribution. These hypotheses do not hold in our work.

Constraint-based algorithms build the graph in two steps. First, an undirected graph (skeleton) is built and in a second step the edges are oriented. The inference of the skeleton starts with a fully connected graph (i.e., a graph in which every vertex is connected to any other vertex). Then we test all the possible conditional and unconditional independences between the different parameters. We remove an edge connecting two vertices when their corresponding parameters are found to be independent. This first phase consists of a series of independence tests. In the PC algorithm [SG91], the sequence of tests can be optimized to decrease the number of independences to test. In the PC algorithm, we start with a fully connected graph and test all the unconditional independences, removing an edge between two nodes whose corresponding parameters are found independent. For the parameters whose nodes are still adjacent, we test for each pair of adjacent nodes if there exists a conditioning set of size 1 that makes them independent. If such set exists we remove the edge connecting the corresponding two nodes, otherwise the edge is not removed. This step is repeated, increasing the conditioning set size by one at

each step, until the size of the conditioning set reaches the maximum degree of the current skeleton (the maximum number of adjacent vertices for a given vertex), which means that no more independence can be found. The second phase, the orientation of the edges, can be performed using two different methods. One method orients the V-structures (subgraphs $X - Z - Y$ where X and Y are not adjacent) in a first step and then orients as many edges as possible without creating new colliders (in $X \rightarrow Z \leftarrow Y$, Z is called a ***collider***) or cycles [Pea09] in a second step. The second method, which is implemented in the kPC algorithm, is based on Weakly Additive Noise (WAN) models [TGS09] and tests the independence of residuals by conditioning on their regressor. To determine the orientation of a given undirected edge, the regression from a child on one of his parent leads to a residual independent of its regressor, while this is not the case if the regression is made from a parent on one of its children (assuming the WAN property holds).

When we started our work, we used the Tetrad software [SGS01] to compare the graphs that could be obtained using score based methods with the graphs that could be obtained using constraint based methods. Tetrad implements several algorithms that we describe in Section 1.3.5. However both, the constraint based algorithms and scored based algorithms are implemented in Tetrad based on the assumptions of linearity and normality. The parameters describing the performance of the applications relying on telecommunication networks do not verify these assumptions. Despite these limitations, it appeared that constraint based methods tend to infer more informative models whose dependencies are closer to our domain knowledge. While the experiments we made with the Tetrad software did not prove that score-based methods cannot give correct results for our data, we decided to use constraint-based methods in our work.

The performance of the kPC algorithm, which uses nonlinear regression, highly depends on the goodness of fit of the regression method used. As the dependencies between the parameters defining TCP performance do not follow any given functional model, the graphical causal models we obtain with the kPC algorithm suggest that the existing nonlinear regression functions could not correctly capture the dependencies, as no orientations could be made in most of the cases. This is the case in the study of the Dropbox application performance that is presented in Chapter 4. The study of the Dropbox performance was one of the first study that was made in our work and the models we obtained with the kPC algorithm suggests us that the use of non linear regression to orient the edges of the causal dependencies in the second phase of the causal model inference was not accurate. Therefore, in our work we use a method relying on colliders to orient the edges of the Bayesian network representing the causal model of our system. Because our choice does not rely on theoretical results but on our observations, we also present the causal models obtained with the kPC algorithm in the study of the emulated network, Section 5.1.3, and in the

study of the FTP traffic, Section 5.2.2 that also show the inadequacy of this approach.

Remarks

It is worth noting that the PC algorithm is able to infer a causal model up to its independence equivalence class (called Markov Equivalence Class and represented by a Partially Directed Acyclic Graph (PDAG) also called pattern). It often happens that several graphs exist that imply the detected independences (see Section 1.3.4 and the paragraph presenting the d-separation graphical criterion). As a consequence, the output of the PC algorithm is a graph where only the independences leading to a unique orientation of the edges are present. The other edges are left unoriented. In our work we use the bnt toolbox implementation of the PC algorithm [Mur01] and select the member of the equivalence class inferred by the PC algorithm based on our knowledge and understanding of TCP networks.

It often happens that domain knowledge informs us on the presence of a causal dependence between two parameters of our model ($X \rightarrow Y$) or, on the opposite, on the absence of a causal dependence between two parameters ($X \not\rightarrow Y$). In [Mee95], the author presents an algorithm that, in the absence of latent variable, is able to infer a causal model that is consistent with both the detected independences and a prior background knowledge. The set of rules presented in [Mee95] is proven to be sound when no latent variable is present and the graphical causal model can be represented by a DAG, or its Markov Equivalence Class can be represented by a PDAG. In the event of latent variables present in the model, the inference of a causal model, where indirect dependencies can be present, is more complex [SMR95]. However, [BT12] also proposes a set of procedures to make use of prior domain knowledge to infer the Maximum Ancestral Graph that represents the (possibly indirect) dependencies between the parameters of the system that one is trying to model.[2]

An important notion in the problem of causal model inference is the one of ***faithfulness***. The faithfulness property is an assumption made by any causal model inference algorithm that stipulates that the properties exhibited by the underlying model, its independences in our case, are stable. This stability means that the properties we detect for the studied system are not entailed by the particular circumstances under which the observations of such system happened, or, expressed differently, that such properties would still be valid if some of the system parameter values would be different. Summarized in a concise way, the faithfulness assumption means that the only independences entailed by

[2]Note that this prior domain knowledge can be used in existing implementations of graphical causal model inference algorithms present in tools such as Tetrad or the BayesNet Toolbox.

the system being studied are structural independences. The faithfulness assumption is related to the relation between causal dependencies and structural dependencies, by opposition to parametric dependencies. In the presence of deterministic dependencies, a given parameterization of the system can cancel the influence of two parameters. If $Y = 3X + Z$, and for a given value x of X we have $Z = -1/3x$, Y might seem independent of X. However such independence is not structural and, would the parameter values be different, such independence would not be true. Hence, such independence would not be stable and would violate the assumption of faithfulness.

When studying a system, one important step is the choice of the set of parameters that will represent the system. The choice of these parameters can only be dictated by domain knowledge. However, some considerations need to be taken into account:

- When representing a system with a Bayesian network capturing the different causal dependencies, a vertex V_i, named according to a parameter X_i, also captures the action of all unobserved parameters that influence the parameter X_i (and no other parameters of the model). Therefore, it is not necessary to have all the parameters of all the mechanisms influencing the performance that one wants to model with causal model but, at least, one parameter per mechanism that rules the system behavior.

- In order to infer the causal model of our system we need to test the different independences between the different parameters that represent our system. The presence of an unobserved variable influencing more than one observed variable is called a ***latent variable***. The presence of latent variables makes the independence tests inaccurate. Even if there exist algorithms to detect the presence of a latent variables in a model [SMR95, SGS01], the model inferred by such an algorithm is not necessarily accurate and such a model cannot always be used to predict intervention. We will present some of the causal models obtained using the IC* algorithm [Pea09, Page 52]. These models often inform us on the *possible* presence of latent variables in our system. However, it is very difficult to detect with certainty the presence or absence of latent variables in a set of parameters representing a system.

- Domain knowledge should dictate the choice of the parameters representing the system being studied in order to obtain a system where all the mechanisms influencing its functioning are captured by parameters that we observe.

- In theory, one could start with an exhaustive list of parameters and remove the parameters that are found unnecessary in the system model. However, in practice, we do not always have access to all the parameters

that influence the behavior of the system. In addition, the algorithm execution time is an important limitation that needs to be taken into account. Section 3.2 discusses the independence test used in our work. The number of independence tests grows very fast with the number of parameters as we are dealing with a combinatorial problem. Increasing the number of parameters that represent our system also increases the size of the maximum conditioning set on which independences need to be tested in the constraint based algorithm used for the causal model inference. As presented in Figure 3.1 in Section 3.3.1, the completion time of the independence test used in our work increases very fast with the conditioning set size.

Finally, a last remark concerns the system observation and data acquisition. An important fact when studying the parameter causal dependencies through statistical test is the presence of ***outliers***. As we will see from the formulas in Section 1.3.4, outliers are important as they often represent points of interest for our predictions. We are often interested in predicting scenarios that differ from the one that are commonly observed. To do so, we need to have observed parameters in situations where they take values in ranges outside the area of the principal mass of their distribution. On the other hand, the presence of outliers can make more complex the detection of structural, causal, dependencies if they are based on statistical tests. In our work, we often restrict our input data to the samples falling in the $[\mathcal{P}_5, \mathcal{P}_{95}]$ during the stage of the causal model inference, where $\mathcal{P}_X$ represents the X^{th} percentile of a given parameter observed values.

Feature selection

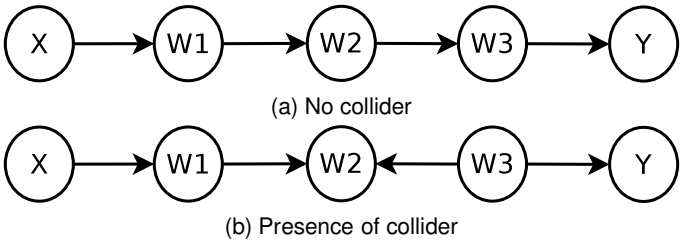
A last point that we would like to insist on concerns the choice of the parameters that will form the nodes of the Bayesian network representing the causal model of your system. As mentioned previously, it is mainly domain knowledge that will suggest the parameters that impact the performance of the system being studied. In the event where our model does not include a parameter impacting more than one of the parameters being observed, such a parameter is called a latent variable. The presence of a latent variable makes the detection of the independence between two parameters more complex. In $X \leftarrow \mathcal{L} \rightarrow Y$, where $\mathcal{L}$ is a latent variable, X and Y might look dependent while they are not. Such a scenario makes a causal study more complex when there are back paths passing through $\mathcal{L}$. The causal theory in [Pea09, SGS01] handles the presence of latent variables. However, as illustrated by the models presented in Figure 5.3 and Figure 5.11 in Section 5.1.3 and Section 5.2.2 respectively, the detection of latent variables is often complex. In our work, we use domain knowledge to define a set of parameters that forms a system where all the dependences

are captured by the observed parameters, leaving ourselves the possibility to extend the list of parameters in the event of a possible latent variable preventing the inference of our system causal model. An example of such scenario is given in Section 5.2.2.

Properties and theorems

In this section, we suppose that we have already inferred the Bayesian network representing the causal model of our system and we present the different properties of such model. We also present the different graphical criteria that we use in our work to predict an intervention from passive observations made prior to the intervention and from the graphical causal model of our system.

D-separation The D-separation criterion is a graphical criterion to judge, from a graph, the independence between two parameters, represented by their corresponding nodes. D-separation associates the notion of connectedness with dependence. Therefore, if there exists a directed path between two nodes, they are said to be d-connected and their corresponding nodes are dependent. On the other hand, if we condition on one of the nodes of this path, then this node is said to block the path and the parameters are d-separated by this node. When studying d-separation, an important notion is the one of ***collider***. A collider, Z, is a vertex part of an oriented subgraph $X \rightarrow Z \leftarrow Y$ where X and Y are not adjacent. The presence of a collider on a path blocks this path and conditioning on a collider unblocks the path. Intuitively, two independent causes become dependent if one conditions on their common consequence. The concept of d-separation is illustrated in the Figure 1.7.



(a) No collider

(b) Presence of collider

Figure 1.7: In Figure 1.7a, X and Y are d-connected ($X \not\perp\!\!\!\perp Y$) but conditioning on any W_i d-separates X and Y, $X \perp\!\!\!\perp Y|W_i$. On the opposite, in Figure 1.7b, X and Y are d-separated by the presence of a collider, W_2, on the path, $X \perp\!\!\!\perp Y$. But conditioning on the collider unblocks the path, $X \not\perp\!\!\!\perp Y|W_2$

Direct effect Our main interest lies in the prediction of the effect of an intervention on a system performance. The performance of the systems we will

study is often represented by the TCP ***throughput***, captured by a single parameter, Y.

The effect of an intervention on a the performance of a system, including the effects mediated by intermediate variables, is called the ***total effect*** and is presented in the next section.

However, it can also be interesting to study the effect of a variable, X, on another variable, Y, without taking into account the effect that an intervention on X could have on other parameters that will, in turn, influence Y. We call this effect the ***direct effect***. The direct effect between two parameters X and Y only makes sense if their corresponding nodes are adjacent in the graph representing the different parameters dependencies.

The study of direct effects highly depends on the nature of the structural dependencies between the parameters of the system we are studying.

Linearity: In the case where parameter dependencies can be modeled with linear relationships we can use the following result from [Pea09].

Theorem 1. ***(Single-Door Criterion for Direct Effects)*** *Let G be any path diagram in which a path coefficient α is associated with the link $X \to Y$, and let G_α denote the diagram that results when $X \to Y$ is deleted from G. The path coefficient α is identifiable if there exists a set of variables Z such that:*

1. *Z contains no descendant of Y; and*
2. *Z d-separates X from Y in G_α.*

If Z satisfies these two conditions, then α is equal to the coefficient of X in the regression of Y on X and Z, r_{XYZ}. Conversely if Z does not satisfy these conditions, then r_{XYZ} is not a consistent estimate of α.

Non linearity: Once dropping the hypothesis of linearity, the inference of direct causal effects is more complicate and requires more work than the use of the Single-door criterion presented in [Pea09]. The reason comes from the fact that the single door criterion is used to identify the set of variables that, if conditioned on, blocks the influence of external variable and isolate the effect of one variable on another. In the case where the dependence between two parameters whose corresponding nodes are adjacent is not linear. The value of the regression coefficient depends on the value at which the conditioning set is set. The same author, Judea Pearl, proposes a method to estimate the direct effect between two adjacent parameters whose dependence is not linear [Pea13] that applies to our problem. We are interested in the estimation, or approximation, of the direct effects to guide the choice of the intervention that would give to the system performance the best improvement. While the theory applies to our

problem, in [Pea13], the estimation of direct effects requires a series of predictions of interventions which is both difficult to compute in practice and highly resource demanding. Therefore we did not estimate the direct effects with the rigorous approach from [Pea13] and leave this as a potential extension of our work.

Remarks on the study of direct effects: Chapter 3 presents the different challenges that need to be overcome when we try to use the different algorithms and theorems presented in this section. However, it is important to notice that the prediction of an intervention requires resources both in terms of data and in terms of computational power. One reason why we are interested in the study of direct effects, is to decide between several possibilities of interventions. In the case where the study of the direct effects can be done in a light and easy way, we can judge from the results of the study of the direct effects which intervention should have the stronger impact on the system performance by intervening on the parameter whose direct effect on the response variable is the strongest.

In the situation where the dependencies are linear, the study of direct effects is reduced to a series of linear regressions and falls into the category of light and easy studies. However, in the case where the dependencies are not linear, the study of the direct effects requires the prediction of a series of interventions. If the reason why we study direct effects is to reduce the number of predictions of interventions the study of the direct effects is not useful anymore.

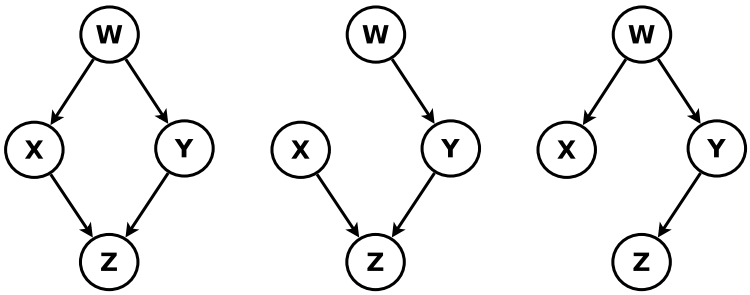
Total effect and do calculus In this section we focus on a the dependence between two parameters. We call these two parameters X and Y, where Y represents the performance of our system. We are interested in the global effect on Y when intervening on X, including the effects mediated by external parameters also impacted by this intervention. We call this causal effect the ***total causal effect***.

We use the notation $do(x)$ to denote the intervention that consists in intervening on the the parameter X to fix its value to be x.

If we use G to denote the Bayesian graph that represents the causal relationships between the parameters of our system, we use $G_{\overline{X}}$ to denote the sub-graph of G where all the edges entering X are removed and $G_{\underline{X}}$ the sub-graph of G where all the edges exiting X are removed, see Figure 1.8. We can use the following rules from [Pea09] to estimate the distributions of the parameters of our system after an intervention based on their distributions prior to this intervention. Note that

- these rules do not rely on any assumption regarding the distributions or functional dependencies of the parameters,

- P represents the (possibly multivariate) probability distribution specified by the probability mass function or probability density function depending on the nature of the parameters.



(a) G (b) $G_{\overline{X}}$ (c) $G_{\underline{X}}$

Figure 1.8: Examples of the different graphical interventions

Theorem 2 (3.4.1 from [Pea09]). *(Rules of do calculus) Let G be the directed acyclic graph associated with a causal model [...] and let P(.) stand for the probability distribution induced by that model. For any disjoint subsets of variables X, Y and Z we have the following rules.*

Rule 1*(Insertion/deletion of observation):*

$$P(y|do(x), z, w) = P(y|do(x), w) \text{ if } (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{X}}} \tag{1.5}$$

Rule 2*(Action/observation exchange):*

$$P(y|do(x), do(z), w) = P(y|do(x), z, w) \text{ if } (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{X}\underline{Z}}} \tag{1.6}$$

Rule 3*(Insertion/deletion of intervention):*

$$P(y|do(x), do(z), w) = P(y|do(x), w) \text{ if } (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{X},\overline{Z(W)}}} \tag{1.7}$$

where Z(W) is the set of Z-nodes that are not ancestor of any W-nodes in $G_{\overline{X}}$

Special case: Back-door criterion To estimate the total effect, on a parameter Y, of an intervention on parameter X, we can use the ***back-door Criterion*** [Pea09]. Notice that the back-door criterion is a specific case of the Rule 2 of the Do-calculus inverting the roles of X and Z and setting $X = \emptyset$. We believe that the Back-door criterion is more intuitive than the application of the ***rules of do calculus***. We will use the back-door criterion in the studies presented in Section 5.1 and Section 5.2.

Definition 1. ***(Back-door criterion)*** *A (set of) variable(s) Z is said to satisfy the back-door criterion relative to an ordered pair of variables (X_i,X_j) in a DAG G if:*

1. *no node in Z is a descendant of X_i; and*
2. *Z blocks every path between X_i and X_j that contains an arrow into X_i.*

Using the back-door definition, we can predict the effect of an intervention.

Theorem 3. ***(Back-door adjustment)*** *If a set of variables Z satisfies the back-door criterion relative to (X, Y), then the causal effect of X on Y is identifiable and given by the formula:*

$$P(y \mid do(X = x)) = \sum_z P(Y = y \mid X = x, Z = z)P(Z = z). \tag{1.8}$$

1.3.5 Works, studies and software

Causal model inference

In this section, we present some of the implementations of algorithms used for causal model inference. The goal of this section is not to be exhaustive but to indicate the tools that were used in our work. The softwares described in this section are often intuitive and, while limited by the assumption of completeness (all variables are observed), normality, linearity or discrete values that the methods they implement make, they offer a very simple way to perform causal inference and reasoning without requiring a deep understanding of the underlying mechanisms or assumptions. These tools rapidly appeared limited for our work, However, they allowed us to become aware of the underlying assumptions made by common causal model inference algorithms and the impact of these assumption on the accuracy of the models being inferred by the methods these software implement.

As we mentioned in Section 1.3.4, Tetrad [SGS01] was used at the beginning of our work. Tetrad is a very complete and easy to use software. Tetrad implements several constraint based algorithms, such as such as the PC [SG91], or iMAGES [RHH+09] algorithms for Pattern inference, LiNGAM [HH09] for Directed Acyclic Graph inference, FCI [SMR95] and FCI extensions for Partial Ancestral Graph (PAG) inference. It also implements score based algorithm such as GES [Max02] (please refer to the Tetrad documentation for an exhaustive list of the algorithms[3]). Some of the proposed algorithms, such as

[3] http://www.phil.cmu.edu/tetrad/

LiNGAM, assume linearity. For others algorithms, such as GES, iMAGES, FCI or the PC algorithms, it is the Tetrad implementation that assumes either linearity or normality. More precisely, for constraint based approaches, it is the absence of an independence test for non linear and non Gaussian data that makes the use of these algorithms ineligible for our studies. The current release of Tetrad (5.1.0-6) implements an independence test for non linear and non normal data [Ram14]. While this test was not present at the beginning of this thesis, our first attempt to use the PC algorithm with this test failed. We tested the Tetrad implementation of the PC algorithm with the independence test from [Ram14] on the dataset corresponding to the DNS study, Chapter 6 and, after several weeks, no result had been obtained. We then decided to not retain this solution in our work.

The R Pcalg package [KMC+12] implements several algorithms such as the PC algorithm [SG91], the FCI algorithm [SMR95] (to deal with the issue of latent variables) or the GIES algorithm [HB12] (to include interventional data). Again, at the time of this writing, the Pcalg package only implements an independence test that assumes that the parameters of the model are normally distributed and linearly dependent. These assumptions are not valid in our work.

An interesting alternative is offered in the kPC algorithm [TGS09] where an implementation of a kernel-based independence test is provided. As we will present in Section 3.2.2, kernel based independence tests do not rely on any assumption regarding the parameter distributions or the functional form of their dependencies . However, the results we obtained with the kPC algorithm and the independence test provided in its implementation from R. Tillman did not lead to consistent and exploitable results. Some causal models inferred by the kPC algorithm are presented in our study of the performance of the Dropbox application in Chapter 4. Some causal models inferred with the kPC algorithms are also presented in Section 5.1.3 and Section 5.2.2.

Finally, the Bayes Net Toolbox (BNT) from Murphy [Mur01] was used in our work. The BNT includes implementations of the PC and IC algorithms and offers the flexibility to adapt these algorithms to the constraints met by the study of telecommunication networks. Noticeably, the existence of a Matlab implementation of a kernel based independence test, presented in Section 3.2.2, motivated the choice of the BNT software for our work.

1.3.6 Summary

In this section we defined causal models, their representations as Bayesian networks and the properties of such models. We explained the reason why we are interested in modeling the causal dependencies of the parameters of a system with a Bayesian network. The graphical criteria that simplify the prediction of intervention were presented. The advantage of such approach comes from

its use of passive observations exclusively. It is important to note that no experiments, active measurements or additional measurements are needed. The study of a system exclusively relies on the observation of a system prior to its study.

1.4 Causality in telecommunication networks

Section 1.2 and 1.3 presented the two fields that define the topic of our work, causality and telecommunication networks. In this section we expose the reasons why we decided to use a causal approach to study telecommunication network applications and what are the benefits offered by such approach.

1.4.1 Motivation

In Section 1.2, we presented telecommunication networks and some of their specificities. It should be clear by now that domain knowledge alone may not be enough if one should take into account all the actors participating the performance experienced by an end user of the Internet network and understand their interdependencies and predict the effect of modifying some component of the network.

As we stated in the introduction to Causality, presented in Section 1.3, we need domain knowledge to define the set of parameters that will represent the system that we want to study. However, when the number of these parameters increases, intuition and domain knowledge are not sufficient to explain their dependencies. When we study a system with an important number of parameters, it is very difficult to estimate what will be the effect of changing the behavior of one (or a set of) parameter(s) on the rest of the system. Towards this end, the causal approach tries to propose a formal and automatic way to represent complex system parameter interactions. The causal approach, and the use of Bayesian networks, provides an elegant solution to present structural interactions between the parameters of a system and predict the effect of an intervention without the need to perform additional observations or experiments.

The prediction of intervention is a very powerful property of causal models as they only require the passive observation of the different parameters constituting the system we want to model.

One interesting characteristic of telecommunication networks is the relative ease with which one can obtain data and measurements. As we will see through three different studies in Chapters 4 - 6, different tools for monitoring telecommunication networks exist. It is then possible to obtain an important quantity of information by observing the traffic at a given point of the network.

The multitude of actors and parameters defining the performance of telecommunication networks, the complexity of their interactions and the availability of network measurements are the three main reasons that motivated the work presented in this thesis.

1.4.2 Benefits

Taking as an example the development of fiber optic network access proposed by many ISPs (Section 1.2.6), we can see that telecommunication networks are in an eternal evolution. The equipment on which telecommunication networks rely is becoming more complex and also more expensive. With the development of many applications relying on the Internet network, some of which representing important investment and expected outcome, modifying network components represents an important cost along with an important risk in case of service unavailability or quality degradation.

To evaluate the user quality of experience it is not always feasible or desirable to perform active measurements including experiments that help predicting the effect of changing one parameter of the system to draw conclusions on the best operation to improve user performance or decrease operators costs.

The benefit of the causal approach lies in its ability to predict interventions without the need to perform controlled experiment or even additional observations. By relying on passive observations only, a causal approach offers a very important potential for studying and improving systems as complex as the telecommunication networks while reducing the risks and costs of performing such a study.

Chapter 2

Related work

In this chapter, we present different works from the different domains related to our research. We divide this chapter in three sections. The first section presents existing works related to telecommunication networks: the mechanisms underlying their performance, some of the important protocols ruling their functioning and different studies of telecommunication networks performance. The second section presents existing works related to causality. We first present the main works that defined the theory used along our research and then the different works that allowed us to use such theory in practice. In the last section, we present other studies that have been made to study telecommunication network performance using the causal theory.

2.1 Telecommunication networks

Telecommunication networks have been used for many years and an important number of studies have been made to understand the mechanisms dictating its performance. An exhaustive list of these works is out of the scope of this section. We present a selection of telecommunication network studies important to understand the domain of our research and how our approach differs from the approaches adopted so far.

As our work mainly focuses on the performance of applications relying on TCP, we first present some studies of the TCP protocol. The TCP studies we present in this section gave us the necessary domain knowledge to define the parameters to model the performance of TCP networks.

Chapter 4, 5 and 6 present the study, via a causal approach, of Dropbox, FTP and DNS performance respectively. Therefore, in this section, we also dedicate one sub section to present works related to each of these applications.

2.1.1 TCP performance study

TCP modeling

There exist several versions of TCP, all of which are well documented. The TCP Reno version, that will be the one implemented on the FTP server in Chapter 5, has been studied and modeled in [PFTK98]. The inferred model (PFTK model) was validated through experimental studies. The findings of [PFTK98] were used in our work as a support to highlight the constraints underlying the study of telecommunication network performance. However, the model developed in [PFTK98] was validated at a time when networks were very different from now. The hypothesis of independence between loss and delay, supposed by the PFTK model, is rejected by the causal models of the FTP traffic performance in the emulated network, Section 5.1.3, and in the Internet network, Section 5.2.2. Our choice to use the TCP Reno version was motivated by the knowledge of the TCP Reno version that allowed us to validate the models and results we obtained in Chapter 5. Therefore, while partly incomplete, the work from [PFTK98] allowed us to design our causal study of FTP performance in Chapter 5 and confront our model to the PFTK model.

However, our approach is very different and can be extended to the study of other TCP versions. The solution we propose does not require a deep understanding of TCP mechanisms, as does the model presented in [PFTK98]. The approach presented in [PFTK98] is highly dependent on the TCP Reno version. Any change in the TCP protocol makes the PFTK model obsolete, as highlighted in the study of TCP New Reno [PMW10]. The obsolescence of a model inferred some time ago and the specificity of such model to the TCP version appear more clearly from the presentation of the TCP Cubic version [HRX08]. TCP cubic is currently implemented in the Linux operating systems. An important difference between the approach presented in Chapter 5 and the approach adopted in [PFTK98] comes from the properties of the causal model inferred in our studies. In the studies presented in Chapters 4- 6 we are not interested in the specificity of a given TCP version. Instead, in [PFTK98, PMW10, HRX08], the authors focus on the TCP version being studied. Approaches that focus on the specificities of a given TCP version and are based on their specifications should be used for the tuning of TCP stack parameters to improve TCP/IP network user performance [SK02].

Other studies have been made to estimate the TCP throughput using a neural network [SRKS07] or Support Vector Regression [MSBZ07] that do not rely on the study of a given TCP protocol version and on its specificities. These studies do not take into account the presence of spurious association and can often lead to biased models. All studies based on the correlations between two events observed simultaneously are exposed the risk of modeling spurious associations between the regressors or the model parameters. This problem is

somehow similar to the one of feature selection where the choice of the parameters, or features, of your model has an important impact on the accuracy of the model obtained, see for example [GE03]. Most importantly, the models inferred from the presence of correlated events are statistical models, by opposition to causal models that are structural. Statistical models cannot be used if we modify the mechanisms responsible of the observations on which the models are based.

In addition, the advantage of our approach comes from the fact that we only use domain knowledge to validate the inferred model but this knowledge is not needed when inferring this model (apart from choosing the smallest list of parameters able to create a self explaining system). In the case of the PFTK model, the obtained equation is derived from a deep understanding and study of TCP as well as the implementation of its mechanisms. On the other hand, graphical causal models offer an understanding of the underlying mechanisms (causal dependences), in addition to the effect of interventions, which approaches that blindly mimic the system under study cannot provide [WB13].

TCP network studies

As said in the introduction of this section, there exist an important number of TCP network studies. Many studies do not rely on any specific TCP version and, instead, try to find explanations of the different observations that can be made from TCP networks.

As mentioned in Section 1.2, the book by James F. Kurose and Keith W. Ross [KR09] presents telecommunication networks in a very exhaustive way, from its principles to the technologies supporting its adoption and development. This book also presents some of the economical considerations and security risks behind telecommunication networks. It exposes the mechanisms behind telecommunication networks without focusing on one aspect in general. While the book has had many re editions to update its content, additional work is needed when studying the performance of a network and many approaches can be used. We can roughly divide these works in two complementary categories.

The first type of work focuses on the study of telecommunication networks through observations or protocol studies. Such works draw general conclusions about the behavior of telecommunication networks and telecommunication network based applications. In this sense they are similar to [KR09]. In [IP11] the authors present changes happening in the Web user behavior analyzing five years of traffic. In [CBS12] the authors summarize many studies made on Broadband network performance and revise some of their findings on the base of six months observations. These works are similar to ours as they try to understand the different mechanisms responsible of the performance that can be experienced by different users observing the Internet network.

The important difference between this first category of study and our work comes from their objectives and the various representations being used. Such studies require an important domain knowledge. They aim to structure the knowledge drawn from the observations of telecommunication in order to explain these observations. Such approaches repeat the risk of exhibiting spurious associations and differ from our approach that uses structural models and allows the prediction of interventions.

The second category of studies concerns the troubleshooting of network performance. In [ZBPS02] the authors observe traffic in the backbone network of Internet and use correlation to group flows sharing similar characteristics in order to explain the throughput obtained by a given flow. Their study is based on a TCP packet-level analysis. In a similar way, in [SUKBC08] the authors present some techniques that allow to define what is limiting the TCP throughput using passive observations exclusively. Studying TCP packet traces, the authors present a set of considerations and metrics that lead to a decision tree from which the factor limiting the throughput can be inferred. To this extent, these works present objectives that are similar to ours but differ in their methods and in their non structural considerations. Many tools also exist to diagnose Web performance. Some of them focus on a Web session performance such as Fathom [DST+12], a Firefox extension that measures the performance of user during a Web session. These measurements are used to detect the deterioration of users performance. The works that focus on the study of Web performance are mainly concerned with the user experience. HostView [JTCT11] performs Web performance troubleshooting that includes a mechanism to capture, in real time, the user Quality of Experience (QoE). Some works focus on Web page rendering mechanisms and Web pages architectures to diagnose Web performance. WebProphet [LZZ+10] builds a critical path in the different steps of Web page rendering and identify the element that limit a user performance.

This second category of works differs from ours while tackling the same topic: the troubleshooting of Internet user performance. They focus on Web browsing and are interested in real time solutions. As presented in Chapter 3, the constraints imposed by the mechanisms underlying telecommunication networks force us to use and adapt complex notions that prevent the design of a real time solution.

It is important to keep in mind the complexity of designing a causal study when we need to select the parameters that will form the model of the system we want to study. When designing a causal study of a given system, we need to define a set of parameters where all the mechanisms impacting the system performance are present via, at least, one of the selected parameters. Towards this end, all the studies presented in this section may help guiding the design of causal studies.

2.1.2 Application performance studies

In the previous section we presented some works related to Web performance. In this section, we present some works more specific to the different applications/systems that we study in our work. We will refer to these works in our causal studies of the Dropbox application, in Chapter 4, the FTP traffic, in Chapter 5, and the DNS service impact on Web performance, in Chapter 6.

As stated, the motivation to adopt a causal approach to study telecommunication networks comes from its superiority over correlation based approaches to block spurious associations and from the structural models that we obtain of the systems being studied. We are then able to predict interventions often impossible to perform experimentally.

Dropbox

The first application we study is Dropbox. As stated previously, the design of a causal study requires a minimum of domain knowledge to define the different parameters that will model the performance of the application that we are studying. In the case of Dropbox, we base our study on the presentation of the Dropbox protocol from [DMMM+12]. In this work the authors present some of the specificities of the Dropbox application, in particular the definition of fixed sized blocks used by the Dropbox applications as the basic unit of information exchanged between the Dropbox client and the Dropbox server. This work also suggests that Dropbox defines its own flow control protocol that, working at the application layer (see TCP/IP model in Section 1.2.4), reduces the use of the mechanisms implemented in TCP for flow control.

Most of the work presented in Section 2.1.1 could also be seen as works related to the study of the Dropbox application performance and are not reported in this section to avoid repetitions.

FTP

Our second study is on the performance of file transfers using the FTP protocol. We first study the performance of the FTP application in an emulated network and, after validating our approach, we study the FTP performance over the Internet network. The reason we chose the FTP application comes from the fact that the FTP performance is mainly dictated by the network behavior. The FTP application can exchange data at the maximum speed allowed by the underlying network. Because the FTP performance is usually the one of the underlying network, many network performance studies are relying on FTP performance studies. FTP can then be used to study cellular

network performance [VDM+11, HA12], WiFi [dCVM+11] or wired network performance [PSN+09]. In [LA04, SJKC06] the authors test the effects of parallelization of resources on file transfer times. All these works differ from ours in the underlying system that is being studied. In our case we focus on the FTP application performance to study the performance of TCP. From this perspective, the works presented in Section 2.1.1 are more related to our study of the performance of the FTP application.

DNS

Our third study analyzes, from a causal point of view, the impact of the DNS service on the performance of users of the Akamai CDN.

Our study differs from works focusing on the CDNs, such as [CJAZ11, TZV+08, RYS09], since we are only interested in understanding the impact of DNS service on CDN performance. On the other hand, many studies have been made on DNS caching strategies, see [FA13] and references therein, that have an important impact on the user quality of experience. In [CAR13] the authors present a study of traffic engineering using DNS mapping and its impact on response time based on the passive observation of DNS traffic. These works, while differently tackling the same question, present complementary results. The main contribution of our study resides in the a methodology based on the inference and usage of a causal model that allows us to estimate the causal effect of the DNS on user performance. Our approach goes beyond the simple parameter observations. Considering the impact of DNS service on Internet user performance, [AMSU10, HMLG11, OSRB12] uses passive measurements to draw conclusions about the role of the different actors related to the choice of DNS service and its impact on the DNS service user Internet experience. Most of these works rely on active measurements and, based on the knowledge of the DNS protocol, interpret the observations of the DNS traffic and the performance of CDN users. Finally, these studies focus on short connections where the slow start of TCP is the phase driving the performance of a user.

Our work focuses on the methodological aspect of our approach and the possibility to predict interventions from which we can obtain a numerical estimate of the impact of choosing one DNS service instead of another on the user performance. Adopting a causal approach allows us to exhibit structural dependencies when studying DNS service, as compared to studies that define criteria to measure DNS performance [ARS13] or that compare different DNS service performance [FSM14] or studying inconsistencies in DNS caching strategies [JSBM02] to observe or propose solutions to improve DNS response time [PAS+04, CWRZ07].

Again, the works mentioned in this section, while different in their objectives,

methods and findings, have formed the base of knowledge that was necessary to design our DNS study. They also represent a different approach to CDN and DNS performance study that should be seen as complementary.

2.2 Causality

In this section we present works related to the causal theory that we use in our work. We divide this section into two sub sections. In the first sub section we focus on the works related to the causal theory and causal model inference that were presented in detail in Section 1.3.4. In the second section, we present some of the existing implementations of the different algorithms presented in the first subsection.

2.2.1 Theory

Causal dependencies and interventions

The work from Judea Pearl [Pea09] motivated the topic of this thesis. Many of the theorems and equations that will be used in our work are directly taken from [Pea09]. While our work is not to be compared with the one from [Pea09], what has been done during this thesis is highly related to this work on which it fully relies. Unlike [SGS01], [Pea09] does not focus on the problem of inferring a system causal model from its passive observations but develops the theory of using the graphical model of causal dependencies to reason and estimate causal effects and predict intervention plans. Many implicit assumptions are made and many challenges are not mentioned (e.g. how to use the theory in practice when we need to estimate a probability density function with a very high dimensionality). To this extend our work extends the domain of application of such theory. The theory developed in [Pea09] is non parametric and can be applied to any system. It is the application of this theory to real case scenarios that is more difficult and is at the center of our work. With practical examples, we show that the theory presented in [Pea09] can indeed be applied to study any system. Our approach is non parametric and most of its results are not dependent on any assumptions. The theory of causality can be applied to any real case scenario as long as we take into account the underlying assumptions of its implementations. Normality and linearity, or discreteness, cannot be assumed for the data obtained from monitoring telecommunication performance. We believe that it is also the case in many other domains. Our work shows that by adapting the existing tools and methods to the constraints of the data a causal approach can be applied and is very beneficial.

An important work is being done at the Carnegie Mellon University (CMU) to support the utilization of the causal theory in a broad range of practical cases.

The book from Peter Spirtes, Clark Glymour and Richard Scheines [SGS01] presents an important number of causal model inference algorithms and their representations as Bayesian networks. The study of direct and indirect effects is made simpler by the assumptions of linearity and normality that are often adopted but many practical examples are given and show how to use the theory in practice.

While these two works [Pea09, SGS01] represent the theoretical basis of our work, these works rely on many other studies and theorems in the field of causality. The same way we cannot present an exhaustive list of all the works that have been done in telecommunication networks, we cannot present all the methods for causal model inference that have been designed and the different studies, based on causality, which extends the causal theory and deal with the constraints of real world systems.

Most of the references and works related to our work can be found in the references given in the two books [Pea09, SGS01].

In the studies presented in Chapter 5 and Chapter 6, we successfully infer the causal graphical model representing the performance of the system that we study. Before using our causal models to predict the effects of interventions, we try to estimate the direct effects [Pea09] between the parameters whose corresponding nodes are found to be adjacent. However, once dropping the assumption of linearity, the inference of direct causal effects is more complicated, and requires more work, than the use of the Single-door criterion presented in [Pea09]. The same author, Judea Pearl, presents a method to estimate the direct effect between two adjacent parameters whose dependence is not linear in [Pea13]. As mentioned in Section 1.3.4, in the case where the system parameter dependencies cannot be modeled by linear dependencies, the estimation of direct effects in our work looses its main interest (namely the study of direct effects to estimate which intervention could better improve the system performance).

Causal model inference

In our work we use the PC algorithm [SG91] for causal model inference, as described in Section 1.3.4, where edge orientation is based on the Meek Rules [Mee95]. While the PC algorithm was designed with the Z-Fisher criterion for testing independence and relies on the assumptions of linearity and normality, it was shown that it still performs correctly when relaxing the hypothesis of normality [VD08].

Other approaches for causal model inference are mentioned in Section 1.3.4, most of which are implemented in the Tetrad project [SGS01]. The Tetrad project implements both constraint based and score based methods, some

of which are mentioned in the following section. In our work, we opted for a constraint based method, however, [Koi06, KS04] propose an scored based causal model inference algorithm with lower than super-exponential complexity in the number of parameters of the system. These algorithms are called *exact* as they always converge and allow to stop at any moment and obtain a graph with a lower bound on its accuracy. However, in these algorithms, ***completeness*** is assumed (all the variables are observed) and the parameters have to follow a normal distribution. These hypotheses do not hold in our work.

We also present the kPC algorithm [TGS09] that relies on Weakly Additive Noise models. Despite its theoretical advantage, the kPC algorithm performed very poorly in practice (see the causal models inferred by the kPC algorithm in Section 5.1.3 and Section 5.2.2). The algorithm from [PJS11] also relies on Weakly Additive Noise model but applies to discrete data exclusively, whereas our data is often a mix of continuous and discrete data.

Another algorithm for causal model inference, whose independence test relies on kernel based criterion, is presented in [SJSF07]. Unlike most of the constraint based algorithms, like the kPC and the PC algorithms that we use in our work, the algorithm presented in [SJSF07] can mitigate the errors made in the inference of the ***skeleton*** (see the paragraph on causal model inference in Section 1.3.4). [SJSF07] uses a score-based approach to decide between different orientations. Unfortunately, we could not obtain an implementation of [SJSF07] and test this algorithm on our data.

Many works also exist to deal with the presence of latent variables [SGS05, ZSJ12] or selection variables [SMR95, CMKR12] in a system model. Most of the algorithms that have been tested in our work show that such algorithms are good at hinting the presence of possible latent or selection variables. Unfortunately, the models inferred by these algorithms are often poorly informative. The notion of possible d-separation (extension of the notion of d-separation to take into account the presence of latent or selection variables) makes the independence tests more critical. The difficulty to obtain coherent models is bigger with the difficulty to detect independences between telecommunication network parameters (see some of the causal models inferred with the IC* algorithm [Pea09] in Section 5.1.3 and Section 5.2.2).

As mentioned in Section 4.4.4, determinism (the existence of deterministic dependencies between variables) violates the hypothesis of faithfulness, Section 1.3.4. There exist works that try to (partially) solve this issue. In [Bau09], the identification of deterministic structures is shown to be difficult, where many conditions have to be met in order to be able to exhibit the underlying deterministic structures. The presence of possible latent variables and the difficulty to detect dependencies between variables where no latent variable is present (see for example the tests comparing the accuracy of Z-Fisher criterion and KCI tests in Appendix A.1) suggest that such algorithm would not give useful results

in our different studies. On the other hand, as mentioned in Section 4.5.2, we try to model the system we observe at a level of granularity above the protocol deterministic mechanisms.

An important notion in Causality is the one of time (a consequence cannot precede its cause). Several algorithms use the notion of chronological events and time series to detect causal dependencies [VDD10, CG08]. However, in the different studies presented in Chapter 4 - 6, our observations use the average value of a given parameter over the duration of a full TCP connection. In such case, the time difference of two connections does not provide any knowledge about underlying causal mechanisms.

As mentioned in Chapter 1, causality and intervention are two notions deeply connected. A possible way to detect a causal dependence is to perform an active intervention [HB14] or mixing observations and interventions [STWB03]. In cases such as gene expressions studies [RJN13], this approach can be considered. However, in telecommunication networks, due to the cost of performing interventions this approach cannot be used in general to infer causal dependencies.

Finally, while in our work we are exclusively focusing on acyclic graphs, it is worth noting that there exist works to infer a system causal graph from its observation in the presence of possible cycles [Ric13].

2.2.2 Toolboxes

In this section, we briefly present the implementations of some of the algorithms described previously and, in some cases, their adaptation to the constraints of telecommunication networks. This section may repeat the information presented in Section 1.3.5. However, the objective of this section is not only to present the existing implementations of causal model inference algorithms but mainly to highlight their limitations and why, in our work, we could not use them.

As mentioned in the previous section, a very complete software for causal model inference (as a Bayesian network) is the Tetrad software. A direct application of most of the methods and theorems presented in [SGS01]. In addition to implement several algorithms such as the PC algorithm [SG91], the GES algorithm [Max02] or iMAGES algorithms [RHH$^+$09] for Pattern inference, it also implements LiNGAM [HH09] for Directed Acyclic Graph inference and FCI [SMR95] and FCI extensions for Partial Ancestral Graph (PAG) inference (please refer to Tetrad documentation for an exhaustive list of the algorithms). Some of the proposed algorithms, such as LiNGAM, assumes linearity while for others, such as GES, iMAGES, FCI or the PC algorithms, it is the Tetrad implementation that does not provide an independence test for not normal and not linear data. The assumptions of linearity and (or) normality that are made by

most of the algorithm implementations present in Tetrad prevent us from using this software.

The Pcalg R package [KMC+12] also provides a very complete solution to causal model inference. Pcalg proposes several causal inference algorithms such as the PC algorithm [SG91] or the FCI algorithm [SMR95] as constraint based algorithm. It also implements the GIES [HB12] as score based algorithm and the back-door criterion [Pea09] to test for the identifiability of the total causal effect of X on Y. Pcalg relies on its implementation of the d-separation criterion [Pea09] or possible d-separation criterion [CM12] for the identifiability of causal effects (see also explanation about total causal effect, identifiability and d-separation given in Section 1.3.4). The reason why we did not use the Pcalg software comes, again, from the absence of an adequate independence criterion to infer the causal model of our system in the case where the parameter dependencies are not linear or the parameter distributions are not normal.

The Bayes Net Tool box [Mur01] implements less causal model inference algorithms but provides more flexibility with the ease to modify the different parts of the different functions implemented. It implements a series of causal model inference algorithms such as the PC [SG91] and IC* [Pea09]. Its main advantage is its modularity and the relative ease with which we can develop and integrate our own functions or the numerous MATLAB implementations of independence tests publicly available. This is the case for the implementation of the kernel based independence used in the kPC algorithm [TGS09] or the the Kernel Conditional Independence test (KCI) [ZPJS12], eventually used in our work.

There is no reason to prefer the BNT toolbox over the Pcalg R package or Tetrad software apart from the richer set of functions available in MATLAB. In our case, it is mainly the MATLAB implementation of the KCI algorithm [ZPJS12] that motivated our choice.

2.3 Causal study of telecommunication network performance

The main challenge, and therefore contribution, of our work has been to adapt the implementations of the methods presented in Section 2.2.1 to the constraints of our problem, namely the non linearity and non normality of the data obtained when studying telecommunication network performance. There are very few studies that have been made for systems such as telecommunication networks, where the assumptions made by the existing implementations of the algorithms of causal model inference and causal effect estimations are not verified. Towards this end, it has been a difficult task to estimate to which extent the

absence of the assumptions of linearity and normality would affect the accuracy of the existing methods to perform a causal inference study.

The WISE system [TZV+08], that uses a graphical causal model to predict interventions in telecommunication network, is the work that is closest to ours. We also present Nano [TMFA09], a system developed by the same authors to detect network neutrality violation.

The WISE system [TZV+08] and the NANO system [TMFA09] adopt a causal approach to study network performance and network neutrality violation. We give an exhaustive presentation of WISE mechanisms and methods and exhibit the differences that exist between WISE and our approach. Then, we briefly present the NANO system that uses a causal approach to block spurious associations and isolate network neutrality violations.

The WISE system [TZV+08] is a complete solution that couples the benefits of the causal approach with the computational power of the Map/Reduce framework [DG08]. WISE is designed to study Content Distribution Networks (CDNs) and to predict how an intervention on content placement strategies or user redirection strategies would impact the response time of a CDN front end server. As done in our work, WISE relies on a causal graphical model to predict interventions on the system being studied. In addition, WISE was designed to handle an important amount of data, which represents an important potential to solve a machine learning problem but also presents the challenge of storing, ordering and accessing such data in an efficient way.

As WISE is the work more closely related to ours, we give in this section a brief description of its functioning that allows us to compare its solutions to the ones we propose and to highlight its limitations. We first present the different steps of the WISE system when it performs the causal study of a given system and when it predicts interventions based on causal model that has been inferred.

A) WISE selects a set of features, the choice being made based on domain knowledge, including the response variable that represents the system performance.

B) Before building the system's causal model, WISE tests all pair-wise independences of any feature with the response variable and discards a preselected feature in the case where it is independent of the response variable.

   (a) In the case where the feature is categorical, WISE compares the conditional distributions of the response variable for the different values of the categorical data and, in the case where all the conditional distributions are similar, WISE considers that such feature is independent of the response variable and discards it. WISE uses a Two-sample Kolmogorov-Smirnov (KS) [Mas51] goodness of fit method to test the similarity of the distributions.

(b) In the case where the feature is not categorical, WISE measures the correlation between the feature and the response variable and, in the case where the correlation coefficient is less than 10%, a quantization process is used to discretize the feature and the KS test described in the previous sentence is used.

C) WISE builds its causal model with the same independence test or using the Z-Fisher criterion and the definition of "no-cause" variables that guide the search of separating variables. In the absence of "no-cause" variables, the PC algorithm is used.

D) To predict interventions, WISE conditions on the interventional variable and then propagates the change corresponding to interventional value to all children of the interventional variable. The conditioning is done by selecting in the dataset the observations where the interventional variable takes the value corresponding to the intervention that WISE wants to predict and randomly picking a value in the corresponding distributions of the children and repeating the process until reaching the response variable.

(a) For the response variable itself, as WISE is interested in its expected value, a non parametric regression function is used.

(b) The different domains, corresponding to the observations of the different features of the model, are divided into fixed width bins. Adjacent bins are then merged until each bin contains a given number of points. To estimate the parameters of the regression function, that will be used to estimate the expected value of the response variable after the intervention and in order to avoid the problem of over-fitting, the regression parameters are estimated for each bin and the final expected response variable value is derived from all the regression parameter estimates.

In B), WISE approach differs from ours. In our work, we consider that the usage of Bayesian networks as causal models simplifies the understanding of the different mechanisms supporting the system performance. Towards this end, the absence of a causal dependence between a given parameter and the response variable is an information that is as useful as when the opposite is found for another feature. For example, in our study of the DNS service impact on CDN performance (captured by the throughput), in Chapter 6, the independence of the standard deviation of the internal RTT and the throughput is an important information that we want to be present in our causal model.

In addition, in B)(a), the approach proposed by WISE does not seem accurate. It is also the case in B)(b) where the feature is not categorical and the Z-Fisher criterion is used, followed by a quantization. Our preliminary tests that use the KS test suggested that the KS test lacks accuracy and cannot be used on our

data. The use of quantization was tested in our work and, from the causal models we obtained, we decided that both the quantization of the continuous parameters and the Kolmogorov-Smirnov test were not accurate to test the independences between the parameters modeling telecommunication network performance.

In C) it appears that the KS independence test (with quantization), or the Z-Fisher criterion, are used. In the case where we are testing $X \perp\!\!\!\perp Y|Z$ by quantizing X and Z or Y and Z and comparing the distributions of $f_{Y|X,Z}$ or $f_{X|Y,Z}$ for the different value of $\{X, Z\}$ or $\{Y, Z\}$, the curse of dimensionality questions the feasibility of such approach. In the case where the Z-Fisher criterion is used, we show in Section 5.1.3 and Section 5.2.2 that such criterion cannot be used to test accurately the independences between the parameters modeling telecommunication network performance. The usage of a quantization and the KS test is neither scalable nor accurate. The authors must therefore use the Z-Fisher criterion which is shown to be inaccurate in our work.

As a side remark, the reduction of the time necessary to perform independence test using the Z-Fisher criterion instead of an accurate independence test, that we do in our work, has a strong impact on the causal inference algorithm. We refer the reader to Section 3.2 that presents the constraints implied by using an independence test taking into account the specificity of the system being studied both in terms of resources and completion time. In Appendix A.1 we also compare the performance of the Z-Fisher criterion with the one of the independence test that is used in our work both in terms of accuracy and resources.

In D), WISE starts from the Markovian assumption [1] that supports many of the theories developed in [Pea09, SGS01]. However, instead of blocking spurious associations by using this assumption, the authors condition on the interventional variable and propagate changes with a regression function whose parameters are estimated based on the quantization of the samples and a nearest neighbor method. Obviously, such approach requires a very important amount of data. The WISE system was designed to benefit from parallel computations both in its filtering, propagation and usage of fixed bin width quantization so it can still handle large amounts of data and computations. Notice that WISE requires an important data pre formatting, storing and pre processing as it is working with terabytes of data.

In D)(a), conditioning on the intervention variable presents the very important risk to unblock a path on which the intervention variable is a collider and to create spurious associations (notice that the same remark will be made for the NANO system). Blocking such associations is one of the main advantage of using a graphical causal model to represent the causal dependencies between the parameters of our system. The single-door criterion (linear case), the

[1] In a the Bayesian network representing the causal model of the system, a node is independent of its ancestors conditionally to its direct parents

back-door criterion, the front-door criterion and more generally the do-calculus from [Pea09] aim mainly to solve the issue of spurious associations. In our work, we rely on the formal theory from [Pea09, SGS01] to predict interventions using passive observations and the graphical causal model inferred from the same observational data to derive the equations that allow us to formally estimate the distribution of the parameters of the system after an intervention. This approach, in addition to its rigor, requires less resources both in terms of data and computational resources.

In D)(b), the definition of fixed width bins that are eventually merged together or divided in order to obtain equilibrated bins that serve the estimation of the distribution of the response variable post intervention, is mainly used to cope with the lack of the data on the fringes. We tested a parametric approach to estimate the post intervention distribution in our study of the DNS impact on CDN performance. The results show that this approach does not perform well, see Appendix B.

To summarize:

- WISE independence tests are based on Kolmogorov-Smirnov test or Z-Fisher criterion that are not adapted to the non linearity and non normality of telecommunication network data. Our usage of the KCI algorithm, when coupled to our bootstrap method, clearly outperforms these two criteria, see the different causal models obtained in the study of the FTP traffic in Chapter 6 and the comparison of the Z-Fisher criterion and KCI test in Appendix A.1.
- The estimation of the post-intervention distribution of the response variable is made through a process similar to conditioning and knowledge propagation. The conditioning phase presents important risks in the case where the intervention variable is a collider. Conditioning on a collider may create a spurious association between the intervention variable and the response variable. As we can see in the different studies presented in Chapter 5 and Chapter 6, the existence of what is known as back door paths is common in the causal studies of complex systems where many parameters, and their dependencies, have to be considered. For this reason, in our work, we use the graphical causal model to derive equations that allow us to obtain an unbiased estimate of the post intervention density of the response variable.
- Finally, our solution, due to its formal approach in the use of the causal theory, requires far less resources both in terms of data and computational power. The formal use of the rules of *Do-calculus* allows us to derive equations where the minimum number of distributions are estimated to compute the post-intervention distribution of the response variable. In the case of WISE, by using a process of knowledge propagation, where

all the ancestors of the response variable distributions have to be estimated, an important, and often unnecessary, number of computations are made. It is also worth noting that, in the study presented in Chapter 6, we show that adopting a more formal approach in our work allows us to predict more complex scenarios, related to counterfactuals. Counterfactuals are very interesting concepts for understanding the causal role of the different parameters of a system and, to our knowledge, scenarios such as the ones presented in Section 6.4.2 have not been treated so far.

The same authors who did WISE have developed the NANO software [TMFA09]. Nano tries to detect network neutrality violation by assessing the causal direct effect between the quality of experience of a user from a given ISP and the type of content being accessed. Based on domain knowledge, Nano selects a set of parameters capturing the different causes of a user quality of experience when accessing a Web content. Nano then observes the different parameters for different ISP, that allows it to define a baseline, set as a normal behavior. To detect a neutrality violation the performance of a given user for who the set of selected parameters takes a given value is compared to the experience of other users served by another ISP but sharing similar parameter values. Again, this approach uses domain knowledge to define the possible confounders and condition on these variables to remove spurious associations. Without a formal causal model this approach presents some risks. One of the confounders could be a collider in the corresponding causal graphical model. Conditioning on a common effect induces a dependencies between two independent causes whose influence tries to be canceled, questioning the obtained results as it was the case for predicting intervention in the WISE system.

On the other hand, while the use of a causal approach is not very common in the domain of telecommunication network performance evaluation, there are many works in other domains such as security [DGK+15], biology [ZLD+12, JZQM12] or social science [CAA+08, Rub74] that demonstrate that a causal approach is suited for the study of complex system with possibly complex parameter distributions or dependencies. However, such works are very specific to the domain they are studying and, while it is interesting noting that causality is a very powerful approach that can be used in a broad range of scenarios, it is out of the scope of this chapter to present such works.

Chapter 3

Adaptation of the causal methods to telecommunication network constraints

3.1 Problem statement

The theory of causality, as developed in [Pea09, SGS01], is non-parametric, does not make assumptions such as linearity or normality, and is not restricted to discrete data. However, most of the implementations of causal model inference methods (PC, GES, Lingam or FCI implementations in Tetrad, bnt toolbox or Pcalg R package) suppose linearity or normality. Note that, in the case of the PC algorithm, it is the absence of an independence test for non linear non Gaussian data that prevents its usage in our case. In the current release of Tetrad (5.1.0-6), the test from [Ram14] is now available but we ran different tests that showed that *i)* Such test requires important resources in terms of computational power and completion time; and *ii)* The models we obtained when we used this independence test were not consistent with our domain knowledge. This test was, therefore, discarded. Similarly, the examples of predictions of interventions in these works ([Pea09, SGS01]) only consider discrete data where probability distribution studies can be simplified by the use of conditional probability tables. Dealing with real data, where these assumptions do not hold, poses a number of challenges.

3.2 Independence tests

We devote an important effort to the study of the specificity of telecommunication networks and the impact of the nature of the parameters commonly used

on the independence tests used in the inference of causal models. We show in our study of the Dropbox performance, Chapter 4, that the inference of causal models using a classical approach does not work. The reason why classical methods do not work comes from the assumptions made in most of the implementations of causal model inference algorithms, namely linearity and normality. Therefore, the first step has been to understand in which proportion the non linearity of parameter dependences and the non normality of the parameter distributions affect the accuracy of the existing implementations. The second step was, then, to adapt the existing implementations to the constraints of our problem, the study of telecommunication network performance.

3.2.1 Constraints of telecommunication network parameters

Distributions

Figure 3.8 presents the histograms of some of the parameters that will be part of the study of the File Transfer Protocol (FTP) traffic in Chapter 5. We can notice several important points:

- The parameters present in the study of telecommunication networks can be continuous or discrete.
- The parameters present in the study of telecommunication networks do not follow any known distribution such Gaussian, Gamma, Beta or log-normal distributions. Their distributions seem generally multi modal and cannot, in the general case, be approximated with a normal distribution.

As a consequence, any independence criterion based on the assumption of normality is bound to fail. Such statement will be verified in the study of the Dropbox protocol performance, Chapter 4, and the different models obtained in Section 5.1.3 and in Section 5.2.2, where the Z-Fisher criterion is used. The Z-Fisher criterion [And84, p.133] tests the absence of partial correlation between two parameters X and Y conditionally to a (set of) variable(s) Z. In the case where the parameters follow normal distributions and their dependencies are linear, the absence of partial correlation corresponds to independence. However, this is not the case when these two assumptions are not verified. Unfortunately, the Z-Fisher criterion is the criterion commonly used to test for independence and conditional independence in constraint based algorithm for causal model inference. In particular, the bnt Toolbox [Mur01], the Pcalg R-package [KMC+12] and Tetrad [SGS01] all implement exclusively the Z-Fisher criterion. As mention previously, Tetrad now implements the test from [Ram14]. However, we tested such independence criterion with Tetrad implementation of the PC algorithm [SG91] on the DNS dataset, Chapter 6, and obtain no results

despite letting the algorithm run for several weeks. We present a study comparing the Z-Fisher criterion with the KCI test [ZPJS12], that we adapted to the constraints of our studies, in Appendix A.1.

Dependencies

Our work focuses on the study of applications relying on the Internet network and, more precisely on the TCP protocol. The performance of the TCP protocol is generally measured using its throughput achieved, which corresponds to the quantity of information received by an Internet user in a given amount of time.

An experimental study of the TCP Reno [PFTK98] version of TCP infers a model of its throughput that can be summarized by the following equation:

$$throughput = C \cdot \frac{MSS}{RTT \cdot \sqrt{p}}, \tag{3.1}$$

which will be later referred to as the **PFTK model**. In this model C represents an empirical constant. The parameter p represents the probability of a loss event, the occurrence of at least on loss in TCP window of packet. The Maximum Segement Size (MSS) represents the biggest packet size accepted by routers present between the two communicating entities. The *Round Trip Time (RTT)* represents the time for a packet leaving from the server to reach the client and for the corresponding acknowledgment to be sent by the client and received by the server.

This model, while limited in its representation of the TCP mechanisms, has been tested and its approximation validated by experimental studies. It should be noticed that a more general version of this model includes the receiver window, mentioned in the presentation of the TCP algorithm in Section 1.2.5. The PFTK model from [PFTK98] highlights the existence of non linear dependencies between the parameters impacting the performance of TCP. A log linear transform was tested in our work but the results we obtained showed that *i)* The PFTK might be too approximative and *ii)* The dependencies are too complex and the Z-Fisher criterion too sensible to non linearity to use such criterion even when we transform the dependencies between the parameters. We present some of the causal model inferred with the PC and Z-Fisher criterion after a log transformation in the studies of the FTP traffic in the emulated network and Internet network in the Section 5.1.3 and the Section 5.2.2 respectively. In our work, we are interested in studying TCP networks performance. Therefore the constraint of highly non linear dependencies will be present and will need to be considered when studying the Dropbox application performance, Chapter 4, the FTP application performance, Chapter 5, or the impact of the DNS on CDN user performance, Chapter 6.

It was shown in [VD08] that the PC algorithm, with the Z-Fisher criterion, still performs correctly when relaxing the assumption of normality. However, the non-linearity has a strong negative impact on its accuracy. The study of the Z-Fisher criterion presented in the Appendix A.1 also confirms the strong negative impact of the presence of non-linear dependencies on the accuracy of the Z-Fisher criterion.

3.2.2 Kernel based independence tests

Definition

In our work we use a kernel based independence test, similar to Hilbert Schmidt Independence Criterion (HSIC) [GFT+08], referred to as the Kernel Conditional Independence (KCI) test from [ZPJS12]. We use the KCI test to test for the independence of two parameters X and Y (denoted $X \perp\!\!\!\perp Y$) or to test for the independence of two parameters X and Y conditionally to a (set of) parameter(s) Z (and denoted $X \perp\!\!\!\perp Y|Z$). A brief survey of independence testing and the comparison between the KCI and HSIC is presented in Appendix E.

The KCI test presents two important advantages. First, the KCI test does not rely on any functional dependence between the parameters for which the independence is being tested. Second the KCI test does not need to estimate the distribution of the parameters being tested. Many technical details are voluntarily omitted in this presentation of the KCI test, mainly for the conditional test, and we refer the reader to [ZPJS12] for further details. In the scope of our work or in our use of the KCI test, we do not need a deep understanding of the Reproducing Kernel Hilbert Spaces or the different operators defined in such spaces to test independences. However, the description of the unconditional test shows the main principles supporting the KCI test and the advantages of the approach. This description also helps us to understand the practical issues that have been faced when using this test in practice and that are presented in Section 3.2.2.

To test for the independence between two parameters, X and Y, conditionally to a (set of) parameter(s) Z, the KCI test uses the Reproducing Kernel Hilbert Space (RKHS) defined on the domains of functions defined on X, Y and Z. We then start this section by presenting RKHS.

We take as an example a domain $\mathcal{D}$ and a Hilbert space $\mathcal{H}$ of real-valued functions defined on $\mathcal{D}$.[1]

Let us then define the linear *evaluation functional* over $\mathcal{H}$ as:

$$L_x : f \rightarrow f(x), \forall f \in \mathcal{H}. \tag{3.2}$$

[1] In this example we chose real value functions as it closer to the situations we face in our work. However, there exists many Reproducing Kernel Hilbert Spaces that are complex-valued function spaces.

In the case where L_x is a continuous function or, equivalently, where L_x is a bounded operator, the evaluation functional can be represented by taking the inner product of f with a function K_x, the *reproducing kernel*. The *Riesz representation theorem* demonstrated that for all x in $\mathcal{D}$ there exists a unique reproducing kernel, K_x, such that:

$$f(x) = L_x(f) =< f, K_x >, \forall f \in \mathcal{H}, \tag{3.3}$$

where $< \cdot, \cdot >$ is the inner product of $\mathcal{H}$.

As $K_x \in \mathcal{H}$, we can then define:

$$K_x(y) =< K_x, K_y >, \forall x, y \in \mathcal{D}. \tag{3.4}$$

The reproducing kernel of $\mathcal{H}$ is then defined as

$$\begin{aligned} K :& \mathcal{D} \times \mathcal{D} \to \mathbb{R} \\ & x, y \to K(x, y) = K_x(y) \end{aligned} \tag{3.5}$$

Symmetrically, every positive-definite kernel,K, generates an Hilbert space $\mathcal{H}_K$ for which $K(x, y)$ is the reproducing kernel. A positive-definite kernel must verify

$$\sum_{i,j=1}^{n} c_i c_j K(x_i, x_j) > 0 \tag{3.6}$$

for $n \in \mathbb{N}$, $\{x_i\}_{i\in[1:n]} \in \mathcal{D}^n$ and $\{c_i\}_{i\in[1:n]} \in \mathbb{R}^n$

Having introduced the notion of RKHS, we can now come back to the problematic of testing the independence between two variables X and Y and of testing the independence of X and Y conditionally to Z.

For a given positive-definite kernel family (Gaussian kernels are often used), we can define three RKHS $\mathcal{H}_\mathcal{X}$, $\mathcal{H}_\mathcal{Y}$ and $\mathcal{H}_\mathcal{Z}$ corresponding to a Hilbert space of functions defined on X, Y and Z respectively and with values in $\mathbb{R}$.

If we observe different values of X, $\{x_1, \ldots, x_n\}$, and in the case where we use Gaussian kernel matrices, the element of $K_X(i, j)$ is defined as:

$$K_X(i, j) = e^{\frac{\|x_i - x_j\|^2}{2\sigma_X^2}}. \tag{3.7}$$

Using the notation $\hat{\mathbf{K}}_X$, the centralized kernel matrix is defined as $\hat{\mathbf{K}}_X = \mathbf{H}\mathbf{K}_X\mathbf{H}^T$, with $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T$, $\mathbf{I}$ the identity matrix and $\mathbf{1}$ the vector of 1's. H is a centering matrix that subtracts the mean of each component $\mathbf{K}_X$. The same notations are used for Y and Z.

The eigenvalue decomposition is defined as $\hat{\mathbf{K}}_X = \mathbf{V_X}\mathbf{\Lambda_X}\mathbf{V_X^T}$, where $\mathbf{\Lambda_X}$ is the diagonal matrix containing the positive eigenvalues $\lambda_{x,i}$. With $\boldsymbol{\psi}(X) = [\psi_1(X), \ldots, \psi_n(X)] =$

$\boldsymbol{V}_X\boldsymbol{\Lambda}_X^{1/2}$, it is possible to define $\boldsymbol{\psi}_i(X) = \sqrt{\lambda_{x,i}}\mathbf{V}_{x,i}$ where $\mathbf{V}_{x,i}$ is the ith eigenvector of $\hat{\mathbf{K}}_X$.

We can also define the eigenvalues, $\lambda^*_{X,i}$, and eigenvectors, $u_{X,i}$, of the kernel k_X. If we call the PDF of X $p(x)$, we have:

$$\int k_X(x,x')u_{X,i}(x)p(x)\mathrm{d}x = \lambda^*_{X,i}u_{X,i}(x'). \tag{3.8}$$

Adopting the same notations for Y, with $\phi_{\mathbf{y}} = \mathbf{V}_Y\boldsymbol{\Lambda}_Y^{1/2}$, and ordering the eigenvalues in ascending order, using the following definition:

$$S_{i,j} = \frac{1}{\sqrt{n}}\psi_{\mathbf{x}}^T\phi_{\mathbf{y}} = \frac{\sum_{t=1}^n \psi_i(x_t)\phi_j(y_t)}{\sqrt{n}}, \tag{3.9}$$

the following result was proved in [ZPJS12]:

If X and Y are independent:

$$\sum_{i,j=1}^n S_{i,j}^2 \xrightarrow{d} \sum_{i,j=1}^n \lambda^*_{X,i}\lambda^*_{Y,j}\cdot z_{i,j}^2, \text{ as } n\to\infty, \tag{3.10}$$

where $z_{i,j}^2$ are i.i.d. χ^2-distributed variables.

The authors notice that:

$$Tr\left(\hat{\mathbf{K}}_{\mathbf{X}}.\hat{\mathbf{K}}_{\mathbf{Y}}\right) = Tr\left(\psi_X\psi_X^T\phi_Y\phi_Y^T\right) = n\sum_{i,j=1}^n S_{i,j}^2. \tag{3.11}$$

As the eigenvalues,λ^*, are not known in practice, empirical values are used instead.

Therefore, the authors define two statistics to test for unconditional and conditional independences.

$$T_{UI} \triangleq \frac{1}{n}Tr\left(\hat{\mathbf{K}}_{\mathbf{X}}.\hat{\mathbf{K}}_{\mathbf{Y}}\right) \tag{3.12}$$

where n is the number of samples.

Under the hypothesis $H_0 := X \perp\!\!\!\perp Y$, the statistic T_{UI} has the same distribution as

$$T_{UI}^{\triangle} \triangleq \frac{1}{n^2}\sum_{i,j=1}^n \lambda_{\mathbf{x},i}\lambda_{\mathbf{y},j}z_{i,j}^2 \tag{3.13}$$

where $z_{i,j}$ are i.i.d. standard Gaussian variables (i.e.,$z_{i,j}^2$ are i.i.d. χ_1^2 distributed variables) and $\lambda_{\mathbf{x},i}\lambda_{\mathbf{y},j}$ are the eigenvalues of the centralized matrices $\hat{K}_X$ and

$\hat{K}_Y$.

Similarly, to test for conditional independence, the following statistic is defined:

$$T_{CI} \triangleq \frac{1}{n} Tr\left(\hat{\mathbf{K}}_{\ddot{\mathbf{X}}|\mathbf{Z}}.\hat{\mathbf{K}}_{\mathbf{Y}|\mathbf{Z}}\right) \tag{3.14}$$

where $\ddot{X}$ is defined as (X, Z).

Under the hypothesis $H_0 := X \perp\!\!\!\perp Y|Z$, the statistic T_{CI} has the same distribution as

$$T_{CI}^{\Delta} \triangleq \frac{1}{n} \sum_{k=1}^{n^2} \dot{\lambda}_k.z_k^2 \tag{3.15}$$

where λ_k are the eigenvalues of a matrix product whose values are obtained from the eigenvectors of $\hat{\mathbf{K}}_{\ddot{\mathbf{X}}|\mathbf{Z}}$ and $\hat{\mathbf{K}}_{\mathbf{Y}|\mathbf{Z}}$.

Finally, the authors show that the distributions of T_{UI}^{Δ} and T_{CI}^{Δ} can be approximated by a Gamma distribution $\Gamma(k, \theta)$:

$$p(t) = t^{k-1} \frac{e^{-t/\theta}}{\theta^k \Gamma(k)} \tag{3.16}$$

where the parameters k and θ can be computed from the traces of the centralized kernel matrices of X and Y. Such an approximation is proved to be robust and reduces drastically the computational resources to test for conditional independence (or unconditional independence) and the completion time of a test compared to estimations of the null distribution of T_{UI} or T_{CI} by randomly generating samples according to equations (3.13) and (3.15).

The advantage of this test is that *i)* It does not require to estimate the distribution of any parameter, hence it makes no assumption related to the distributions of the parameters which dependence is being tested; *ii)* The test only requires the estimation of the eigenvalues and eigenvectors of regular kernels, that greatly simplifies the number of computations; and *iii)* By approximating the null distribution of T_{UI} and T_{CI} with a gamma distribution the authors simplify the null hypothesis testing that would, otherwise, require to draw random samples for T_{UI}^{Δ} and T_{CI}^{Δ}, using Equation (3.13) and Equation (3.15) respectively, to approximate the distributions of the statistics under H_0. Such an approach is computationally demanding, above all in the case where a conditional independence is tested and where $\hat{\mathbf{K}}_{\ddot{\mathbf{X}}|\mathbf{Z}}$ and $\hat{\mathbf{K}}_{\mathbf{Y}|\mathbf{Z}}$ need to be calculated as well as their eigenvalues and eigenvectors.

Advantages of the KCI

As mentioned previously, the reason why we use the KCI test comes from the absence of assumptions it makes concerning the parameter distributions and

the nature of their dependencies. In addition, compared the kernel based independence used in the kPC algorithm [TGS09], an important effort is made to decrease the computational time of testing an independence between two parameters. From the presentation given in the previous section, we could see that the test mainly relies on the computation of the eigenvalues of the centered kernel matrices of the observed values of the parameters whose independence is tested. With the use of Cholesky factorization, the diagonalization and computation of the eigenvalues is greatly simplified, both in terms of computation resources and, consequently, computational time.

In addition, the KCI was shown [ZPJS12] to outperform the previous implementations of the the independence tests based on RKHS and derived from the cross-covariance operator [FGXS08, GKH$^+$08] and to perform correctly in cases where the number of samples is limited and the conditioning set size is important.

Limitations of the KCI

The implementation of the KCI test suffers from two limitations: First, the computation time to perform independence tests with this criterion is still much longer than for the Z-Fisher test; in particular to test the conditional independences. Second, the implementation of the KCI test uses matrix operations (Cholesky factorization) that, although theoretically well justified, may fail due to numerical issues. This is a known numerical limitation(see, e.g., Appendix B in [Tip01]); due to a matrix which, although theoretically invertible, appears singular in practice. In this case, the test will not return an answer.

These two problems can be avoided by using datasets of smaller size (or sub-datasets), but at the expense of a possible loss of accuracy.

Modification of the implementation of the KCI

To solve the two problems present in the use of the KCI test in practice, we modified the independence test by including a bootstrap method [DH97], which works as follows.

For each independence test of $X \perp\!\!\!\perp Y \mid Z$ (X independent of Y conditionally on Z), we re-sample the dataset (with replacement) and create l new sub-datasets of N samples. The test is performed on each re-sampled sub-dataset (by comparing the p-value to an objective level of significance of the test, often denoted as α) and we accept the independence if a fraction of at least δ of sub-tests detects that the variables are independent.

For clarity reason, the different tests that have been performed to validate and test our procedure of KCI+bootstrap are presented in the following sections. We

present the main results in Section 3.2.2. As shown later, this testing procedure KCI+bootstrap does not decrease the accuracy of the test, compared to the KCI test on the full dataset, as long as N is large enough. On the other side, it offers a number of valuable advantages. First, it reduces the computation time and offers an easy and efficient way to parallelize the computation on different machines. Second, it reduces the numerical issues due to Cholesky factorization and therefore allows to obtain a result in cases where the KCI test on the full dataset would simply fail. Also, the bootstrap method offers a way to estimate the confidence in the result of the independence tests by observing the variations of the p-values in the sub-tests.

Parameterization of the solution

Section 3.3 presents a detailed study of the KCI+bootstrap procedure in terms of computation time and accuracy. Here, we briefly summarize the main results that justify our choice of parameters.

Choice of δ, the threshold percentage of accepted tests We choose $\delta = 0.5$, which corresponds to comparing the median p-value obtained on all re-sampled sub-datasets to α. Parameter δ is related to the significance level of the sub-tests (α). Its value reflects a trade-off between error of type I and type II: one can increase δ to decrease the risk of type I error (wrongly accepting an independence) with the drawback of increasing the risk of type II error (rejecting a valid independence) or decrease δ to reduce the risk of type II error but increase the risk of type I error.

Choice of the N, the sub dataset size in the bootstrap method The accuracy is, to some extent, dependent on the variability of the data. If the variation in the dataset is important, more data is required to capture an independence, with a number of observations increasing with the size of the conditioning set. However, increasing the size of the sub-dataset has a strong impact on the computation time. Our experiments showed that, for our system, choosing $N = 400$ yields no degradation in accuracy when compared to the use of the full dataset (consisting often of several thousands of samples). Such observation is not made for values smaller than 400. For $N = 400$, computation times is reasonable while the computation time increases very fast for larger values. For example, for a conditioning set of size 1, a test with N = 400 lasts less than 30s while for N = 500 it already lasts 1 minute (see Figure 3.1). Consequently, we choose $N = 400$.

Choice of l, the number of repetitions of the bootstrap method While large values of l are often advised in the bootstrap method [AB01], the number of repetitions is often dictated in practice by computational constraints. In Table 3.1, Section 3.3.1, we present estimations of the computation times for different values of l. Based on these real constraints, we choose $l = 100$. This choice was initially based on preliminary experiments showing that the bootstrap results become stable beyond 100 repetitions. This choice is also confirmed by the results concerning the accuracy that we obtain in our experimental studies (c.f. Section 3.3.2, which shows that using $l = 100$, together with $N = 400$, yields to the same accuracy of the KCI test as the one if all the observations of the system being studied were used).

Additional remark The choice of the values for δ, N and l values represents a trade-off between computation time and accuracy. This choice most likely depends on the dataset representing the observations of the system being studied. Therefore, a learning phase may be necessary to determine the best value for these parameters. As the final goal is to detect the underlying independences between the different variables of the model, the parameters δ, N and l cannot be fixed through the study of a real case scenario where the independences are not known. To solve this issue in the case of the FTP traffic we generated an artificial dataset where the parameters follow distributions similar to the ones of the parameters of the system we are studying and where the dependencies are defined based on the knowledge we have of the system that we want to study. This learning phase and its results are presented in Section 3.3.2 where, for two different scenarios, the parameterization N = 400 and l = 100 is validated. In the DNS service study, we kept the same values for N and l, as these values are mainly dictated by computation time considerations, and use our domain knowledge to choose the value of δ.

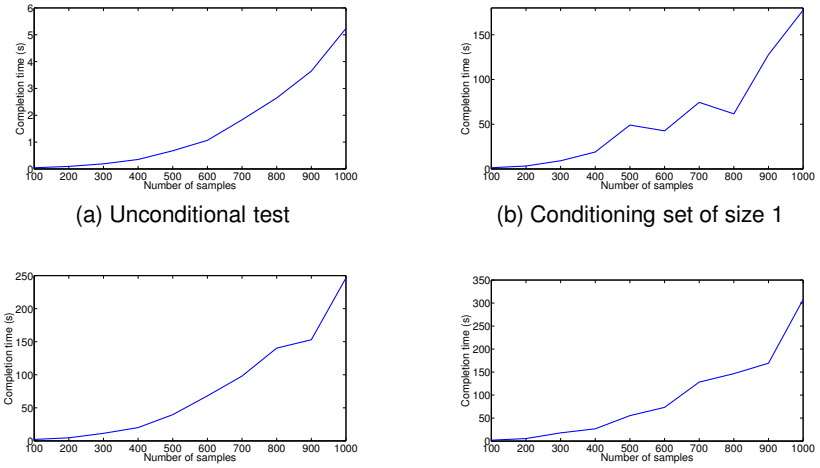
3.3 KCI+bootstrap parameterization

3.3.1 Completion time

We first discuss computation time constraints.

KCI test completion time

In a first experiment, we record the time it takes for different dataset sizes and for different conditional set sizes to complete an independence test with the



(a) Unconditional test

(b) Conditioning set of size 1

(c) Conditioning set of size 2

(d) Conditioning set of size 3

Figure 3.1: Evolution of the completion time of the KCI test as function of the number of samples for different conditioning set sizes

implementation of the KCI test from [ZPJS12].[2]

The results are shown in Figure 3.1. All graphs show that the completion time seems to increase faster than linearly with the number of samples.

In practice, for a sample size larger than 400, the completion time starts to increase very fast with the number of samples. Together with the observation that no important improvements are obtained for sample sizes bigger than 400 samples (see Section 3.3.2), this motivated our choice of the value 400 for N.

Additionally, in Figure 3.2 and Figure 3.3, we present the result of approximating the completion time with a polynomial of degree 3. Figure 3.2 shows that a polynomial of degree 3 gives a rather good approximation of the completion times of the different tests for different sample sizes and different conditioning set sizes. In Figure 3.3, the figures ares repeated in a semi-logarithmic scale. While the results presented in these figures are limited, the observations are consistent with the hypothesis of a superlinear evolution of the completion time of the KCI test as a function of the number of samples.

Influence of bootstrap parameterization on completion time

Taking as an example the system studied in Section 5.2 where we model the causal influence of 11 parameters on the throughput of client downloading objects from an FTP server, we estimate the time it would take for different parameterizations of our bootstrap method to infer the corresponding causal model.

[2]Note that this completion time was obtained with MATLAB 2009b, while the same trends are still valid for latest release of MATLAB.

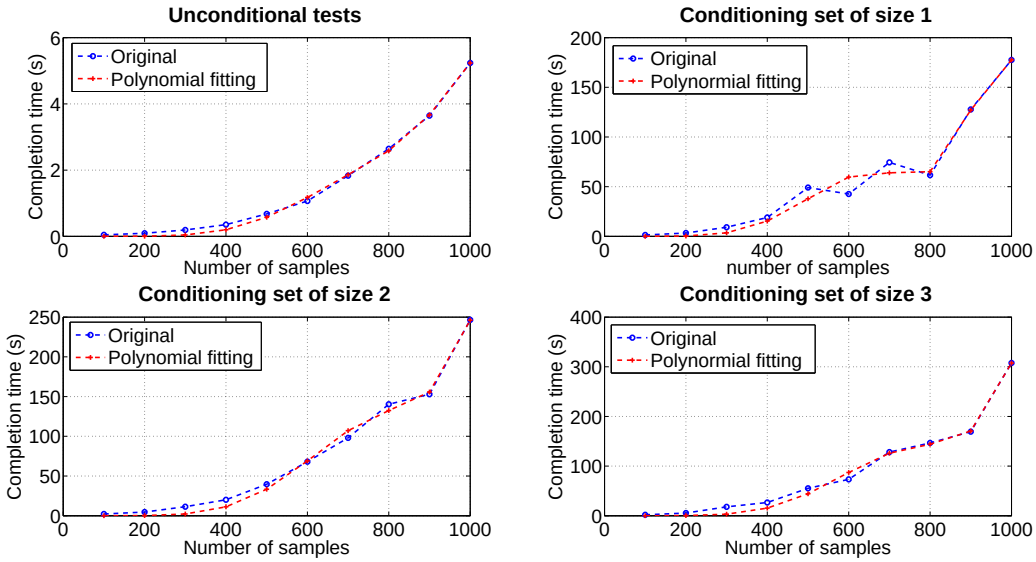


Figure 3.2: Approximation of the KCI completion time with a polynomial of degree 3

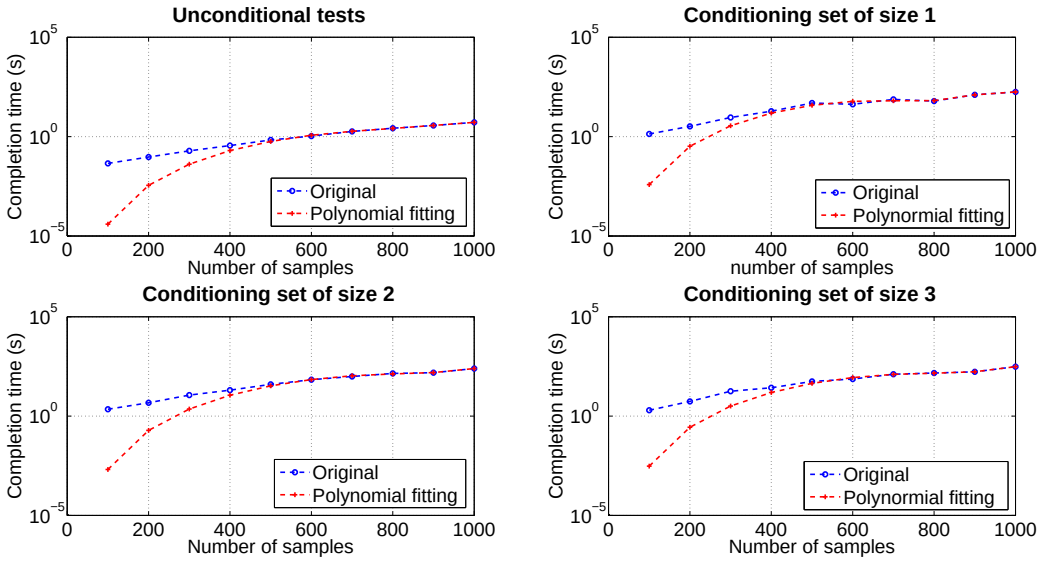


Figure 3.3: Approximation of the KCI completion time with a polynomial of degree 3, Y log scale

To do so, we first use the PC algorithm with the Z-Fisher criterion and count the number of independences that were tested for each conditioning set size. We then measure, for different values of l and N the time that it takes to test one independence for a conditioning set size of 0, 1, 2, ... until the maximum conditioning set size that was observed in the different tests performed

by the PC algorithm when the Z-Fisher criterion is used. We can then obtain a rough estimate of the time it would have taken for the PC algorithm to infer a causal model corresponding to the same scenario if it would have used the KCI+bootstrap method for different values of l and N. The results are presented in Table 3.1.

Note that the different estimations had to be run in parallel on different machines with slightly different performances: 12 CPUs, 2260 MHz, 30 GB or 8 CPUs, 1862 MHz, 15 GB (but note that only one or two Central Processing Unit (CPU) were used during the tests so the difference in the number of CPUs is not important). The use of different machines explains small variations in the estimates (for example, that the completion time estimate for $\{N = 200, l = 1000\}$ is found smaller than for $\{N = 100, l = 1000\}$); but this does not affect the main trends.

Table 3.1 highlights important time constraints that limit the number of sub-tests in the bootstrap. Indeed, for $N = 400$, computation times are of the order of days and go to almost a year for $l = 1,000$. It should be noticed, however, that one very important advantage of the bootstrap method (in addition to detecting independences in cases where the basic KCI test would fail, see Section 3.2.2), is the possibility it offers to parallelize the computation very easily. Indeed, when using the bootstrap method, each test on a re-sampled dataset can simply be performed on a different machine. By using M machines, we can then shrink the completion time by a factor proportional to the number of machines used. As it is now very common for a university or a company to work with a cluster of machines (virtually or not), this is a very important advantage. For instance, at our university, we have around $M = 50$ machines. For $N = 400$, this reduces the computation time from 33 days to 18 hours. However, this still limits the number of re-sampling. For instance, for $N = 400$, using $l = 1,000$ the algorithm would still take 5 days. While 5 days could still seem an acceptable time (although it can become unmanageable with less resources or if tests need to be run several times), the important point is that it corresponds to a very large increase compared to 18 hours. In the model used for this estimation we consider only 12 parameters but this difference will rapidly become difficult to handle as soon as more complex systems, with more parameters and independences to test, are studied. For example, in the study of the impact of choosing one DNS service instead of another on a CDN user experience, Chapter 6, our model consists of 19 parameters.

3.3.2 Accuracy of the KCI and bootstrap tests

We finally compare the accuracy of the KCI test with and without bootstrap, in situations close to the ones we encounter in our work.

The notion of performance/accuracy of the test can only be discussed in cases where the independences are known a-priori. To estimate the type I and II er-

Table 3.1: Impact of bootstrap parameters on completion time

N	l	time (hours)	time (days)
100	1	2.1	0.09
200	1	2.5	0.1
400	1	2.9	0.12
600	1	23.9	1.0
800	1	44.0	1.83
1000	1	69.0	2.87
100	10	5.4	0.22
200	10	19.5	0.81
400	10	45.5	1.9
600	10	201.6	8.4
800	10	427.2	18.7
1000	10	839.1	37.18
100	100	52.3	2.18
200	100	283.6	11.82
400	100	804.3	33.51
600	100	924.8	38.53
800	100	4647.1	193.63
1000	100	9688.6	403.69
100	1000	2857.4	119.06
200	1000	2108.8	87.87
400	1000	5935.6	247.31
600	1000	15657.8	652.40
800	1000	30503.5	1270.97
1000	1000	31796.9	1324.86

rors of the test in a conditional independence setting, we consider the graph of Figure 3.5. In order to stay as close as possible to the constraints and nature of the data we are studying in our work, we generate the parameters of an artificial dataset from some of the parameters observed in our studies, presented in Chapter 5.

We generate X_1 by randomly re-sampling (with replacement) the throughput, from the dataset of Table 5.3 first (the causal study of FTP traffic, presented in Section 5.2), and from the dataset of Table 5.1 after (the causal study of an emulated network traffic, presented in Section 5.1). We generate X_2, X_3 and X_4 using non-linear functions:

X_1 = re-sampling(tput);
$X_2 = \sqrt{10X_1} + \mathcal{N}(0, 0.8)$
$X_3 = -5\sqrt{0.5X_1} + \mathcal{N}(0, 0.8)$
$X_4 = \sqrt{6X_2 - 2X_3} + \mathcal{N}(0, 0.8)$,

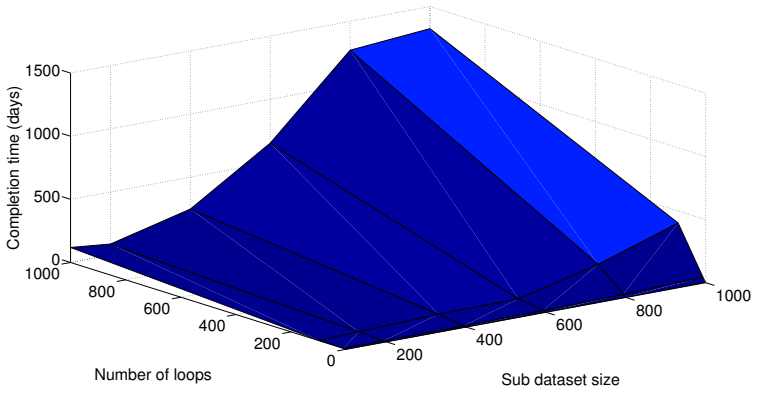


Figure 3.4: Evolution of the completion time for different values of the bootstrap method

where $\mathcal{N}(0, 0.8)$ represents a Gaussian noise with mean 0 and standard deviation of 0.8

To estimate both types of errors, we test the following independences:

- $\mathcal{I}_1$: $X_1 \perp\!\!\!\perp X_4 | \{X_2, X_3\}$: H_0 should be accepted;
- $\mathcal{I}_2$: $X_2 \perp\!\!\!\perp X_3 | \{X_1, X_4\}$: H_0 should be rejected.

H_0 stands for the hypothesis that the independence tested is true.

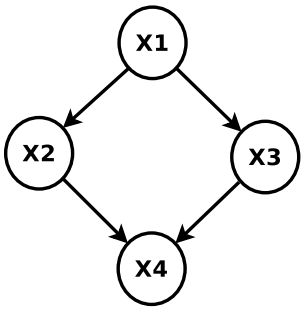


Figure 3.5: Causal graph of the artificial dataset of 4 variables

Generating X1 from the throughput observed at the FTP server in the dataset of Table 5.3 in Section 5.2

Figures 3.6a and 3.6b present the results in terms of size and power of the test as a function of the parameter α (recall that a test rejects H_0 if the p-value is

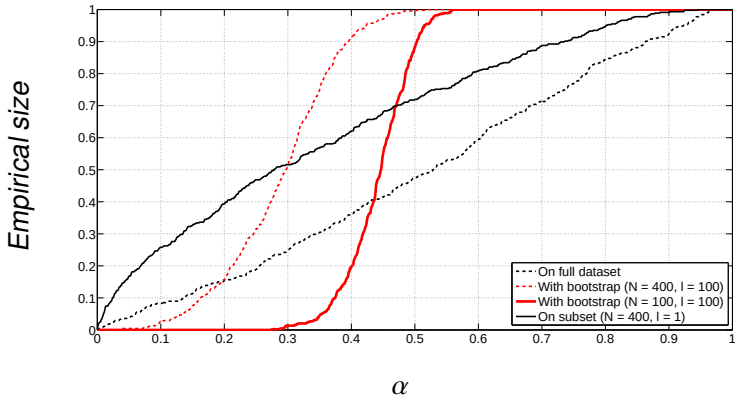
smaller than α). The result for the KCI test on the full dataset is shown with the black dotted line, as the baseline curve. To assess the accuracy of the test using the bootstrap method on smaller re-sampled datasets, we compare two sets of parameters ($N = 400$, $l = 100$), the red dotted line, and ($N = 100$, $l = 100$), the red solid line. For comparison and interpretation, we also include the results for ($N = 400$, $l = 1$), the black solid line, which corresponds to a method doing the test on one re-sampled dataset of smaller size rather than on the full dataset (almost equivalent to taking a subset except that the re-sampling is done with replacement).

The size, defined as the probability of rejecting H_0 under H_0 (probability of type I error), is estimated by testing independence $\mathcal{I}_\mathbf{1}$ on 446 datasets generated as described above (each with $1,000$ samples). Figure 3.6a shows that, as expected, the size for the full dataset test and for the test on a subset are close to the first bisector ($size = \alpha$). For the tests with bootstrap, however, the sizes are very different from α. This is normal since the test can no longer be simply defined as comparing a p-value to α. The power, defined as the probability of rejecting H_0 under H_1 (complementary of the probability of type II error) is estimated by testing independence $\mathcal{I}_\mathbf{2}$ on 446 datasets generated as described above (each with $1,000$ samples). Figure 3.6b shows that, naturally, for a given α, the power for the test on a subset (black solid line) is smaller than the power for the test on the full dataset (black dotted line). The comparison of powers with tests using bootstrap is not meaningful, however, because, for a given α the sizes are very different.

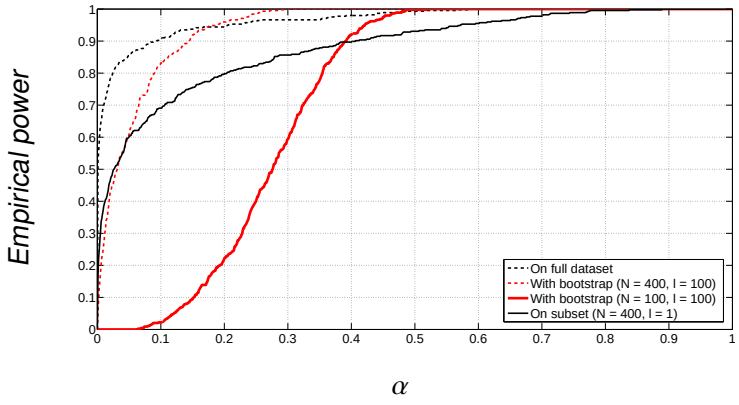
Instead, Figure 3.6c shows the real accuracy trade-off between size and power achieved by each test. The most important result is that the test with bootstrap with parameters $N = 400$ and $l = 100$ (red dotted line) achieves the same performance as the KCI test on the full dataset (black dotted line). This validates, in a situation very close to the one encountered in our system (conditional independences, non-normal distributions and non-linear dependences), both the bootstrap procedure and the choice of parameters. As a comparison, the test on a subset of 400 samples achieves a very poor performance (which shows the importance of the bootstrap) and the test with parameters $N = 100$ and $l = 100$ achieves a smaller but still reasonable performance (which shows that further reductions of the computation time are possible but not without a small degradation of the accuracy).

Generating X1 from the throughput observed at the FTP server in the dataset of Table 5.1 in Section 5.1

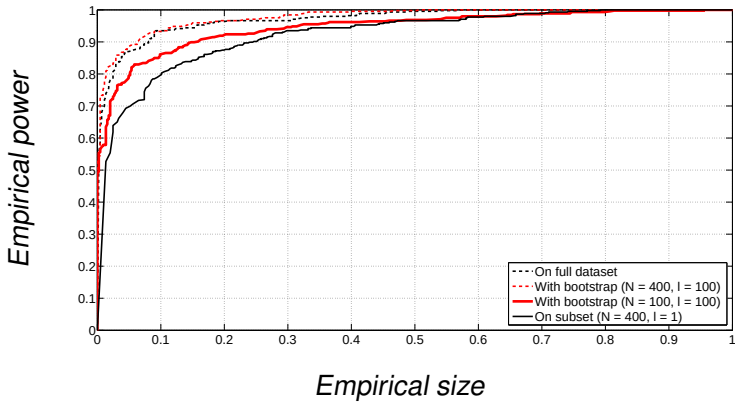
We repeat the same experience as previously but, this time, X_1 is generated from the throughput of the emulated network, Section 5.1. The size, the power and the power as a function of the size are represented in Figure 3.7a, Figure 3.7b and Figure 3.7c respectively.



(a) Empirical test size (i.e., fraction of incorrectly rejected independence $\mathcal{I}_1$) as function of test parameter α



(b) Empirical test power (i.e., fraction of correctly rejected independence $\mathcal{I}_2$) as function of test parameter α



(c) Empirical test power as a function of the empirical test size

Figure 3.6: Comparison of size and power of the KCI test with and without the use of bootstrap for X_1 generated from the Internet network throughput

We focus our discussion on the graph of the power as function of the size, Figure 3.7c. We can observe that the performance of the KCI test when using l = 100 and N = 400 is not as close to the performance obtained with the full dataset as when generating X_1 from the real FTP traffic throughput, Figure 3.6c. However, we still observe less than 5% difference which, for our work, we consider acceptable as compared to the loss in terms of resources that would occur if choosing a higher number of loops or a bigger dataset to increase accuracy. Again, the choice of N and l results from a trade-off between precision and resources and, as in the previous section, the choice of N = 400 and l = 100 offers a trade-off that comply with our needs. An important point to notice here is that this setting was tested in the emulated network scenario, Section 5.1. The graphical causal model we obtain based on the sequence of independence tests using this setting (N = 400, l = 100) was used to predict interventions that could be then verified by manually intervening on the system which further validates our choice of parameters.

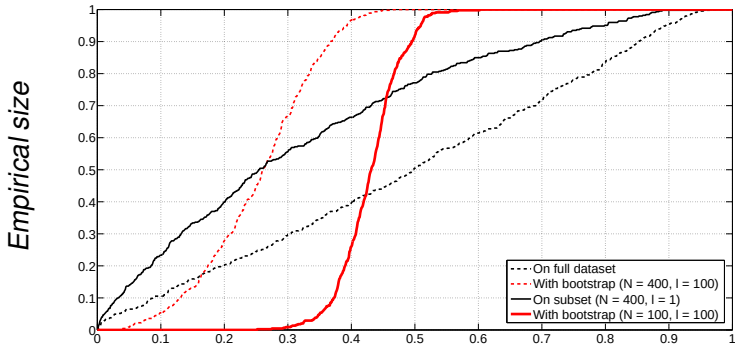
Concluding remarks

The important difference between the previous two cases is that, in the case where X_1 was generated from the emulated network throughput, the setting $\{N = 400, l = 100\}$ was used to infer the graph that supported the predictions we made and that could be verified, see Section 5.1. As independences are not known in advance we use two criteria to validate our setting:

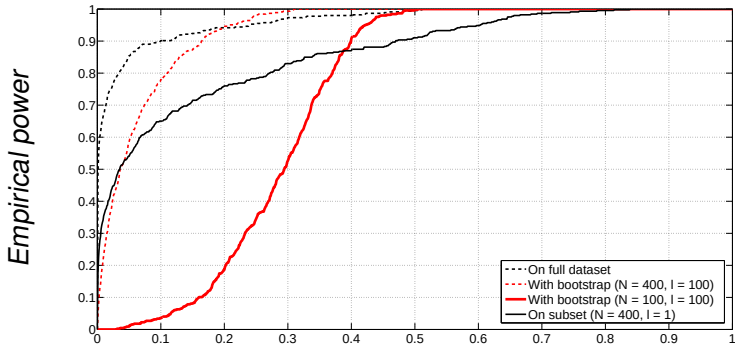
1. We generate an artificial dataset with known independences, distributions similar to the ones of the data we observe in our work and dependences inspired by the system we observe and our domain knowledge. We, then, use this ground truth to parameterize our algorithm.

2. For a given parameterization, for a system where the dependences are not known, we use the obtained setting to infer a graph that is used to predict an intervention that can be performed and verified on the system after. This is the case of the study of the emulated network traffic in Section 5.2. This last point validates the different steps.

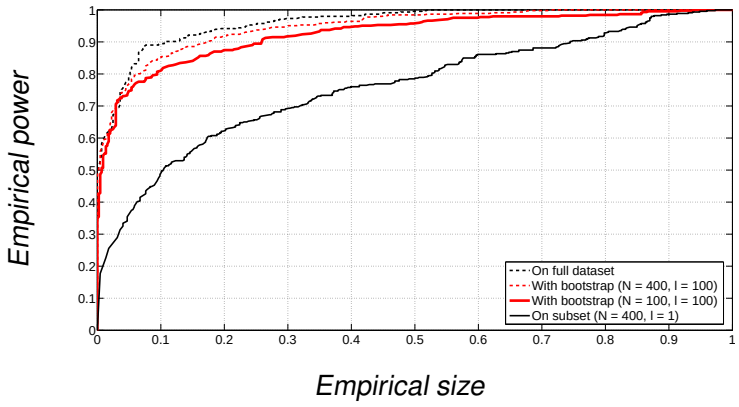
The fact that, for both scenarios, the setting N = 400 and l = 100 meets our objectives in terms of accuracy and resources validates this choice. The small difference in the performance of the test for the two scenarios we observe is also validating the approach as i) We can see that the two datasets are different but the same setting can be applied, ii) The second scenario, where the performance is not as good as in the first one, corresponds to the case where the distribution of X_1 follows the distribution of a parameter of the emulated network where the predictions could be verified and the approach validated.



(a) Empirical test size (i.e., fraction of incorrectly rejected independence $\mathcal{I}_1$) as function of test parameter α



(b) Empirical test power (i.e., fraction of correctly rejected independence $\mathcal{I}_2$) as function of test parameter α



(c) Empirical test power as a function of the empirical test size

Figure 3.7: Comparison of size and power of the KCI test with and without the use of bootstrap for X_1 generated from the emulated network throughput

3.4 Probability density function

3.4.1 Telecommunication network parameter constraints

We repeat here one special case of the do-calculus presented in Section 1.3.4, the back-door criterion and its application to estimate the Probability Distribution Function (PDF) of a parameter Y after an intervention on a parameter X with the back-door adjustment where the (set of) Z variable(s) verifies the back-door criterion:

$$P(y \mid do(X = x)) = \sum_{z} P(Y = y \mid X = x, Z = z)P(Z = z). \tag{3.17}$$

We can see from Equation (3.17) that the estimation of the distribution of the parameters after an intervention relies on conditional probability densities of the parameters before this intervention.

In Equation (3.17) the variables are assumed to be discrete, which is not the case for all variables in the different systems that will be studied in our work. In addition, the parameters we consider in our system do not have an a priori distribution nor can they easily be approximated with a mixture of known distributions such as normal, gamma, beta, and so on. Some examples of the distributions of some parameters that will be present in our studies are presented in Figure 3.8. See [NF11] for an example of an estimation of a univariate distribution with a mixture of pre determined distributions).

We address these challenges using copulae [JDHR10].

3.4.2 Kernels and copulae

A copula is a multivariate probability distribution function for which each marginal follows a uniform distribution. A copula is used to describe the dependencies between the different marginals of a given multivariate distribution. More formally a copula, C can be as

$$\begin{aligned} C: \quad & [0,1]^n \longrightarrow [0,1] \\ & U_1, \ldots, U_n \longmapsto C(U_1, \ldots, U_n) \end{aligned}$$

The Sklar Theorem stipulates that, if F is a multivariate Cumulative Distribution Function (CDF) with marginals $(F_1, \ldots, F_n)$, there exists a copula C such that

$$F(x_1, \ldots, x_n) = C(F_1(x_1), \ldots, F_n(x_n)). \tag{3.18}$$

In the bivariate case, differentiating Equation (3.18) w.r.t. to X_1, X_2 gives the following result:

$$f(x_1, x_2) = c(F_1(x_1), F_2(x_2))f_1(x_1)f_2(x_2), \tag{3.19}$$

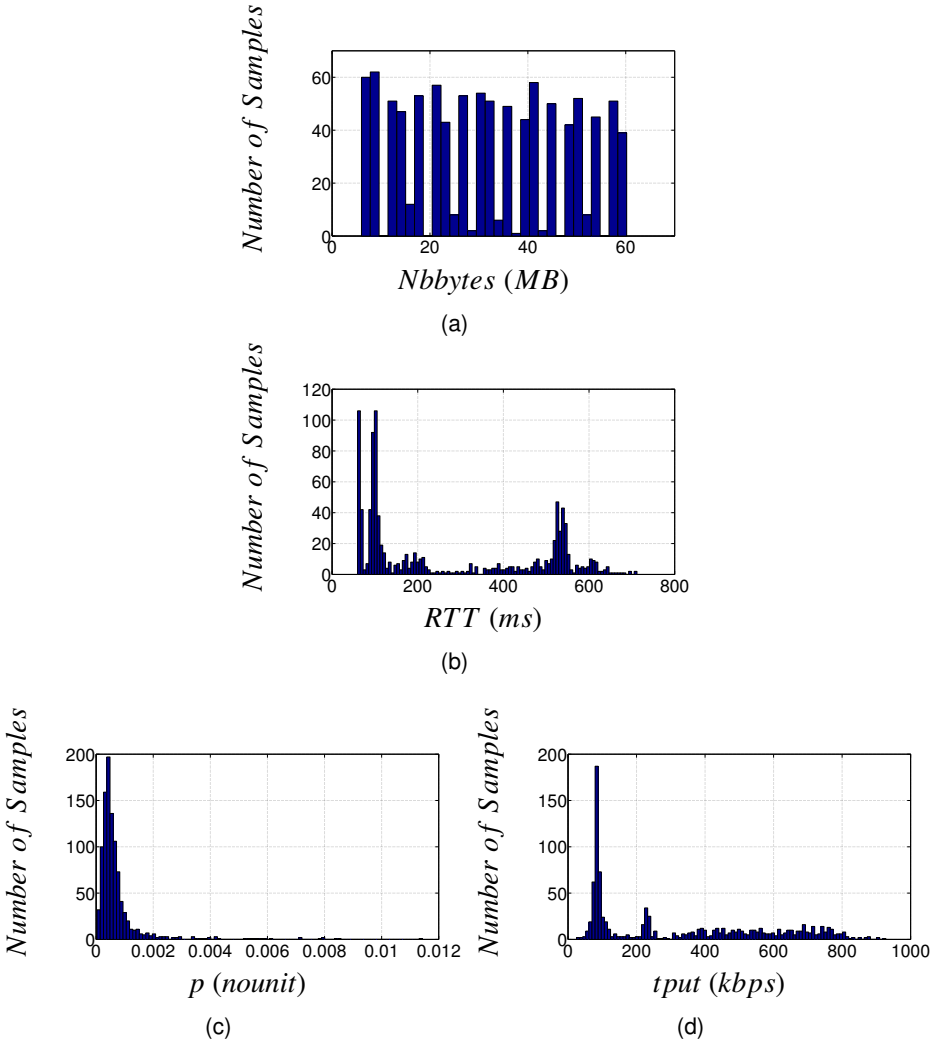


Figure 3.8: Histograms of the *nbbytes (a)*, *rtt (b)*, *p (c)* and *tput (d)* parameters observed in the study of Real FTP traffic, presented in Section 5.2

where $f_i(x_i)$ is the probability density function of the marginal F_i and c is the derivative of C. Assuming $f_2(x_2) > 0$ for all x_2, we have

$$f_{X_1|X_2}(x_1, x_2) = c(F_1(x_1), F_2(x_2))f_1(x_1). \tag{3.20}$$

For the back-door criterion, we often place ourselves in the case illustrated in Equation (3.20). When $dim(X_2) > 1$, we use Equation (3.18) to estimate the multivariate CDF of X_2 and use this estimation in Equation (3.20).

Copulae, as any model, need to be parameterized and the choice of the copula to model the dependencies between the marginals should be dictated by the

real dependencies. In our work we often chose T-copulae as they better capture the tail dependencies between the parameters influencing the throughput.

The T-copula is an elliptical copula and, in the bivariate case, is given by

$$C_{\rho,\nu}(u,v) = \int_{-\infty}^{t_\nu^{-1}(u)} \int_{-\infty}^{t_\nu^{-1}(v)} \frac{1}{2\pi(1-\rho^2)^{1/2}} \left(1 + \frac{x^2 - 2\rho xy + y^2}{\nu(1-\rho^2)}\right)^{-\frac{(\nu+2)}{2}} \mathrm{d}x\mathrm{d}y \tag{3.21}$$

where ρ is the linear correlation coefficient and ν the number of degrees of freedom. One interesting property of the T-copula is the presence of more points (compared to the normal copula for example) in the tail.

An example of a T-copula (ν = 10, ρ = 0.8) is presented in Figure 3.9.

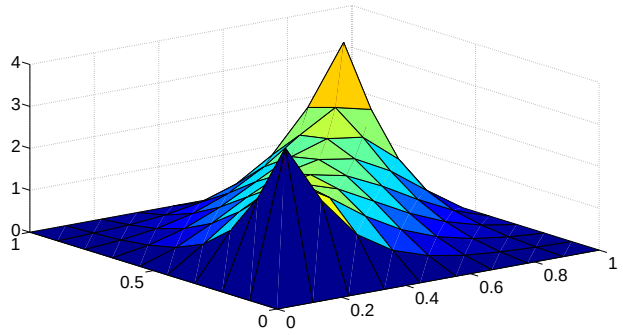


Figure 3.9: Example of a T-copula

As we will see in the study of Domain Name System (DNS) service performance, Chapter 6, the estimation of the parameters of the T-copula modeling the marginals dependencies requires a minimum amount of observations that is not always available. In such case we can use Gaussian copulae defined as:

$$C_\rho(u,v) = \Phi_\rho(\Phi^{-1}(u), \Phi^{-1}(v)) \tag{3.22}$$

where ρ represents the correlation matrix for u and v, Φ the CDF of the standard normal distribution. Gaussian copulae offer less flexibility in their parameterization but in case of low dimension still offer a good trade-off between the resource they require and the accuracy they provide. Figure 3.10 represents an example of a Gaussian copula (ρ = 0.5).

The estimation of the marginals, in our work, is done via kernel functions. Once again, the choice of the kernel (normal, triangle, etc) should be done according to the nature of the parameters we observe. We chose normal kernels as they

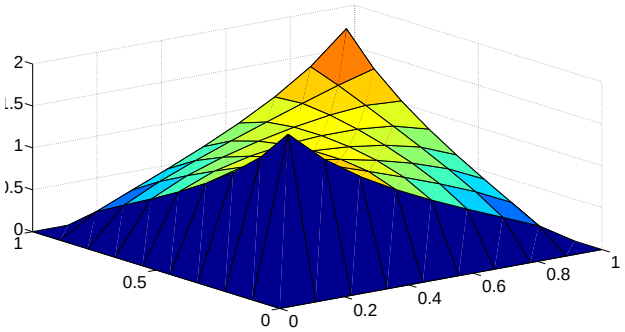


Figure 3.10: Example of a Gaussian copula

give smoother estimations of the marginals that will be the support for future predictions.

A brief introduction to copulae can be found in Appendix D

3.5 Studying telecommunication applications

The previous section showed that, in practice, applying the causal theory to real case scenarios requires a certain amount of work to solve some practical issues when trying to apply existing methods to the study of systems where many constraints need to be considered. Now that we can assume that these issues have been solved, we still need to mention some intrinsic constraints, not directly linked to the application of the causal theory but more to the field of telecommunication network studies.

3.5.1 Adopting a new approach to study a known system

The previous section tried to highlight the difficulties that exist for the application of the causal theory. It is therefore highly ambitious to apply this theory to the study of systems comprising a high number of parameters and complex relationships that lead to models often difficult to interpret.

One important challenge in the application of the causal theory to the study of telecommunication networks has been to define the correct trade-off between exploiting the benefits of the causal approach by disclosing new mechanisms, dependencies or properties and applying the causal approach to systems where the interpretation of causal models could be supported by our domain knowledge.

On one hand telecommunication networks have been widely studied and many theoretical and experimental works have been published [PFTK98, MSBZ07], see also [LGBVBP10] and references therein. On the other hand, most of the existing studies present methods specific to a given application or context. While most of the studies that will be presented in Chapter 5 and Chapter 6 target a given application and rely on domain knowledge in their interpretation of the results, the method we developed in our work aims to be re used in a different context.

However, due to the relative novelty of our approach and the inherent limitations, presented in this chapter, finding an interesting problem where results can be interpreted and still disclose new mechanisms has been an important challenge.

3.5.2 The difficulty of choosing the granularity of our models

One important notion to keep in mind when designing a causal study is the one of ***determinism***. We call a relationship between two (or more) parameters, X and Y, *deterministic* if it can be written

$$Y = f(X). \tag{3.23}$$

Notice the presence of the = sign instead of the := used in Equation (1.2). The Equation (3.23) symbolizes the absence of any randomness in the relationship between X and Y such that knowing the value of X defines the value of Y unequivocally. An example would be $Y = 3 * X$ and, in such case, we can write $X = 1/3 * Y$ equivalently. Such dependencies are not causal and violate the faithfulness assumption (see Section 1.3.4).

It is important to notice that in telecommunication networks the only randomness comes from failures or congestion events. The implementation of a given protocol at a given host is mainly deterministic. It corresponds to a sequence of instructions that are run upon a given input event or trigger from external environment. The applications running on client and server machines reacts with deterministic mechanisms to (possibly) random events. The presence of deterministic relationships between parameters of our systems makes the inference of its corresponding causal model more complex. In our work, we do not address the issue of uncovering causal dependencies in the presence of deterministic dependencies that violates the *faithfulness* assumption used in most of the causal model inference algorithms. Some works exist to infer system causal models in the presence of deterministic dependencies [Bau09]. It is therefore important to chose the level of granularity of the models of the systems we study. If we try to understand the mechanisms that correspond to the implementations of a given protocol on a (set of) local or remote machine(s), there is a high risk to focus on a problem more related to reverse engineering

where the dependencies between the different parameters are deterministic. On the opposite, if we stay at a too high level, we might not be able to capture mechanisms that rule the functioning of a given system and fail to model the system performance. Trying to model the functioning of a car without opening the hood would very likely lead to an accurate model.

A counter-example, that will be presented in Chapter 4, concerns the study of the Dropbox application. As a matter of fact, Dropbox implements its own congestion control mechanisms at the application layer which is, somehow, bypassing TCP mechanisms (Dropbox still works on top of TCP however the rate at which the data is provided to TCP turned out to vary with the value of parameters such as loss or delay which are typically used by the TCP protocol). The causal study of the Dropbox application performance finally converted into an exercise of reverse engineering of the Dropbox protocol. First, it is not our goal to reverse engineering proprietary software. Second, Dropbox has its protocol defined by a set of (proprietary) specifications implemented in the different pieces of softwares of the different agents of the Dropbox system. An important risk is to eventually try to model deterministic dependencies with purely statistical considerations and violated the assumption of *faithfulness* that supports the inference of the causal dependencies with the algorithm presented in 1.3.4.

It is therefore important to chose the application that we want to study but also the degree of abstraction (or granularity) of our model.

3.5.3 Adopting a progressive study approach

In the last two subsections we presented the different challenges, inherent to the subject of our work that influence the choice of our studies, namely the existence of many telecommunication network studies and the determinism of the protocols that rule the telecommunication network communications.

To overcome both of these constraints, and the complexity of applying a causal approach to real case scenarios, our approach has been the following. First we study an emulated network where we can perform the intervention and validate our method. This step is very important as the reason of using a causal approach stays with its ability to predict interventions. Second, we study FTP traffic where we set up our own FTP server, which simplifies its study. Finally, we use measurements from a third party to study the impact of DNS service on telecommunication network performance.

Chapter 4

Dropbox protocol study

4.1 Introduction

In Chapter 3, we presented the different constraints inherent to the study of telecommunication networks and the solutions that were developed to overcome such constraints and re use the existing solutions and implementations that support a causal study. From Chapter 3 it could appear that the development of the presented solutions has been a smooth and straightforward process, which is not the case. In addition, the solutions we presented in Chapter 3 require more resource in terms of data and computational power when compared to using the Z-Fisher criterion to test independences and histograms to model parameter distributions. One could legitimately wonder if such additional cost is justified.

In this chapter, we briefly present a causal study of the Dropbox traffic performance. The study of the Dropbox application was the first causal study that we did. Many building blocks are missing from this study. The causal model inference of the system that we study relies on two algorithms used naively. The first algorithm is the PC algorithm [SG91] and its use of the Z-Fisher criterion [And84] to test for independence. The second algorithm is the kPC algorithm [TGS09] and its implementation of an independence test based on the Hilbert Schmidt Independence Criterion [GFT+08], along with its implementation of a non linear regression to orient the edges in the Bayesian graph corresponding to the causal model of the system we study.

In a first study, we use Intrabase [SBUK+05] to isolate the periods of the Dropbox traffic during which the application is not limiting the performance. We can focus our study of Dropbox performance on such periods where only the network, and Dropbox reactions to networks events, impact the user performance. We then present a second study where we define metrics that better capture the different parameters that influence the performance of the Dropbox appli-

cation, taking into account the specificity of the Dropbox protocol. In these two studies we face the different challenges inherent to the study of telecommunication network based applications and complex systems. We are able to draw important conclusions that support the choices presented in the design of our final solution in Chapter 3 and helped to design the studies presented in Chapter 5 and in Chapter 6. Hence, while this study did not lead to the discovery of exploitable mechanisms and predictions, it played an important role in the understanding of the constraints of causal studies in the field of telecommunication networks.

4.2 Dropbox presentation

In the past few years Dropbox has become the most popular cloud service for file hosting. In [DMMM+12] the authors observe in some networks that the Dropbox traffic amounts for 100GB of daily traffic that represent 4% of the total traffic and is equivalent to one third of the Youtube traffic. So far studies have focused on comparing different cloud storage services, reverse engineer the Dropbox protocol or looking for security issues. In this study we focus on the performance of the Dropbox protocol from the client side. While Dropbox seem to implement its own algorithms to control the rate at which data is exchanged between a given client and a server, it still relies on the TCP protocol. In the following we present the the different metrics that impact the TCP throughput of clients using the Dropbox application and we study their dependencies.

As presented in [DMMM+12], Dropbox hosting service relies on Amazon servers. Every time a file is uploaded to a Dropbox server, clients and servers exchange several messages before a file is uploaded to the server. Meta data and indexing information are exchanged between the client and the server in order to place a client object copy on one of the Amazon server, located in US. In our study we only focus on the file uploads and do not consider the first part of the communication during which meta information is exchanged. In [DMMM+12] the authors reveal that a Drobpox file transfer consists in chunk transfers. Each file that needs to be uploaded is first divided at the application layer, Dropbox, into chunks of 4 MB that are passed to the TCP protocol that is responsible of sending them in sequence. Each chunk needs to be acknowledged by the server at the Dropbox application layer before the next chunk can is passed to TCP to be transmitted, cf Figure 4.1.

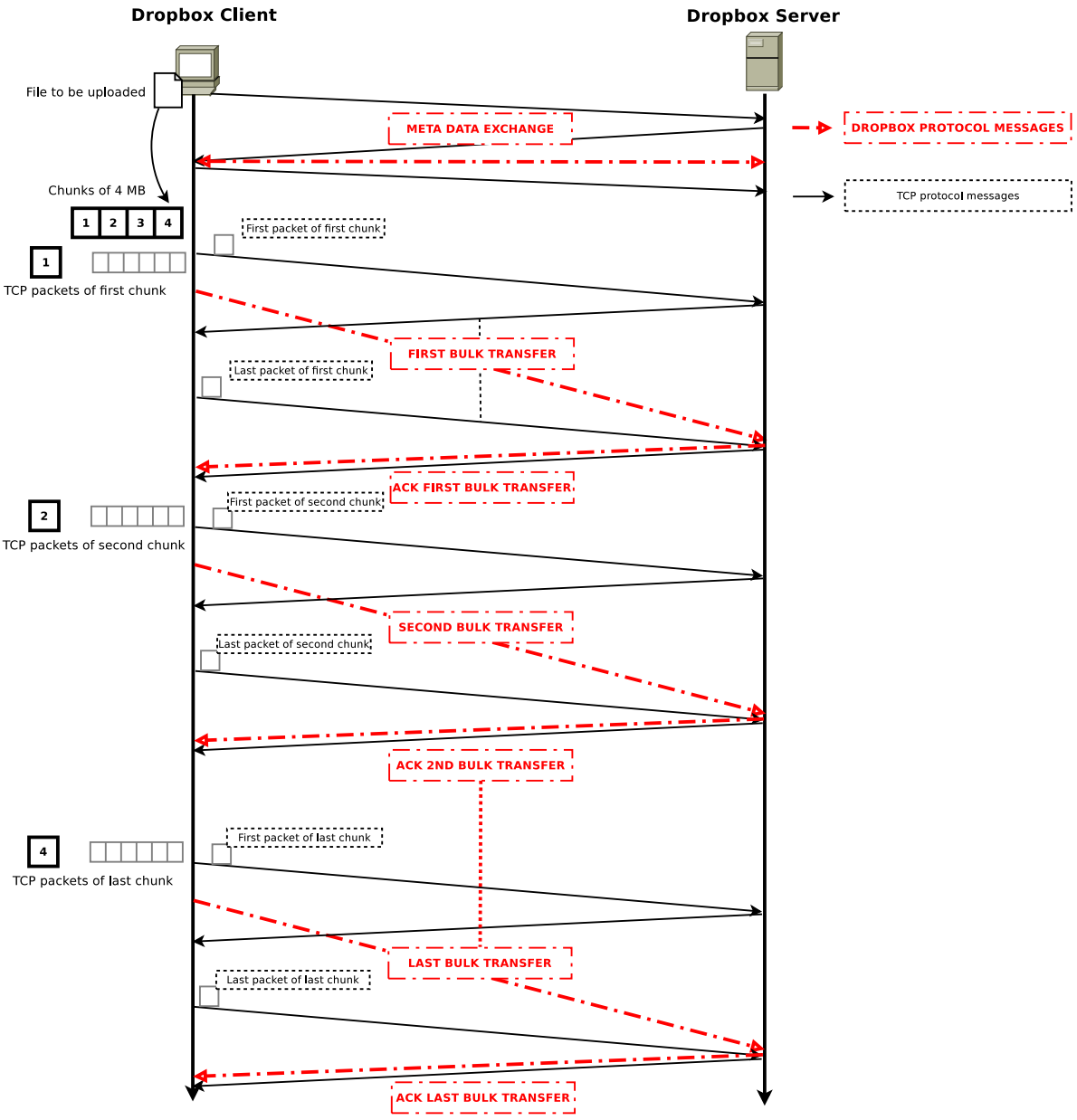


Figure 4.1: Upload of a file to a Dropbox server at the Dropbox and TCP level

4.3 Studying one upload with Intrabase Isolate and Merge algorithm

4.3.1 Intrabase

The Intrabase tool [SBUK+05] identifies the limiting factor for the throughput of a TCP connection. Intrabase is based on the different mechanisms of the TCP protocol, presented in Section 1.2.5. By deriving several metrics that captures the different possible limiting factor of the TCP throughput and comparing their values to (often empirical) threshold, Intrabase can find which factor limits the throughput observed for a given TCP connection. In the OSI model, the application layer gives and receives data from the transport layer. In the case of TCP transport protocol, the packets carries the data provided by the application. The rate at which the application provides data to TCP is one possible limiting factor to the TCP throughput. The first step in Intrabase consists in the division of a TCP connection into Application Limited periods and Transfer periods. If the application is the factor limiting the TCP throughput, we might not have access to the parameters that could explain the observed TCP throughput.

Dropbox implements its own protocol for controlling the rate at which data is exchanged between clients and servers. It seems to implement its own rate control and, for voluminous files exchanged, several parallel connections can be opened between a given clients and the Dropbox servers that it contacts. Intrabase, instead works at the level of the TCP connection and, observing packets exchanges, can identify the limiting factor in the throughput of the TCP connection. As we know that Dropbox is controlling the speed of data transfer, we decided, for each TCP connection, to study the Transfer periods exclusively. Transfer periods are periods during which the application layer is providing the transport Layer enough data so that the TCP throughput is dictated by the TCP and network parameters (see Section 1.2.5).

4.3.2 System observation

In this study we upload a 900 MB file on the Dropbox cloud servers and record the corresponding traffic. The traces were recorded with Wireshark. Intrabase identified 27 connections out of which 3 were data exchange connection between our machine and an Amazon server with more than 120 packets (the minimum number of packets necessary for Intrabase analysis). The existence of several connections to upload an object on Amazon server(s) can be explained by the usage of a distributed file system for Dropbox storage which might include redundancy and error recovery. In addition, to speed up the upload, several connections are opened in parallel.

Based on the traces we defined the following metrics:

- Bytes: The total number of bytes exchanged between Server and Clients
- RTT: The Round Trip Time
- Retrscore: The number of retransmitted packets divided by total number of transmitted packets. We use the retrscore as an approximation of the loss frequency.
- Avgwin: The receiver advertised window, learned from the TCP packet field in the packets sent by the server to the client. The value of this metric is obtained by averaging the receiver window values read in the TCP headers during a bulk transfer.
- Duration: The duration of a Bulk transfer (where a Bulk transfer corresponds to the period that Intrabase identifies as a Transfer period and that should approximate a Dropbox chunk transfer).[1]

4.3.3 Bulk transfer statistics

Observations

In our dataset, each sample represents a bulk transfer period and for each bulk transfer period we measure the metrics described before. We obtain a dataset of 2758 samples. We use the passive observations, summarized in Table 4.1, to infer the causal graph exposing the different causal relationships between the metrics impacting the duration of a bulk transfer.

Table 4.1: Summary of the metrics capturing the Dropbox performance in the case of a 900MB object upload and Intrabase statistics with their average value (**Avg**), minimum value (**Min**), maximum value (**Max**), variance (**Var**), standard deviation (**Std** = $\sqrt{Var}$) and coefficient of variation (**CoV** = $\frac{Std}{Avg}$)

Parameter	Avg	Min	Max	Var	Std	CoV
Size (MB)	11	0.23	110	180	14	1.3
avgwin (B)	940	320	1000	23000	150	0.2
rtt (ms)	120	110	130	0.03	0.005	0.2
retrscore (%)	0.3	0	7.1	0.01	1.0	3.4
duration (s)	26	0.42	250	1600	40	1.6

[1] While we usually use the throughput as performance metric, it turned out that the PC algorithm, with the Z-Fisher criterion, performs better when modeling the duration. However, as throughput = Bytes / Duration, we could not explain these differences apart from the issues of the Z-Fisher criterion to detect independences between parameters which dependencies are not linear.

Causal models

Figure 4.2 presents two causal models inferred by the PC algorithm for different values of the significance level used in the independence tests. By increasing the significance level we require more confidence in the tested independence to accept it. The rejection of the independence between the size of the object being uploaded (*bytes*) and the loss frequency (*retrscore*) creates a new V-structure that allows the orientation of all the edges of the graph.

The causal model inferred with the PC algorithm for a significance level, α, of 0.1 finds the retransmission score (*retrscore*) independent of the duration. If TCP needs to transmit several times a packets to have it received by the receiver, the time to transmit a given amount of information should increase. Additionally, in the presence of loss events, TCP reduces its data sending rate. The presence of losses should increase the duration of a given connection. On the opposite, the number of bytes transmitted (*bytes*) is impacting the retransmission score (*retrscore*) that is a normalized parameter (number of packets retransmitted divided by the total number of transmitted packets). Apart from these two remarks (that should question the validity of the model we obtain) it appears that the model we obtained is not very informative, mainly due to the lack of novelty in the (valid) dependencies it highlights.

Figure 4.3 presents two causal models inferred by the kPC algorithm for different values of the significance level used in the independence tests.

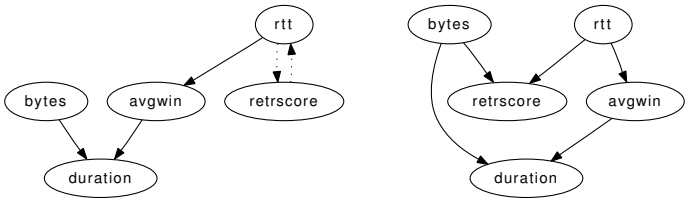
The orientation phase of the kPC algorithm relies on the Weakly Additive Noise Model assumption (see [TGS09]) and operates a non linear regression of the two parameters whose nodes are found adjacent in the first stage of the algorithm. If the kPC is trying to orient the edge between two parameters X and Y, $X - Y$, it performs a regression of Y on X to estimate the residual ε_{XY} and performs the opposite regression to estimate the residual ε_{YX}. If we observe $\varepsilon_{XY} \perp\!\!\!\perp X$ and $\varepsilon_{YX} \not\perp\!\!\!\perp Y$ then, under the WAN assumption, we can deduce the orientation of the edge as $X \rightarrow Y$.

The original implementation of the kPC algorithm proposed by Robert Tillman et al. relied on the multi ridge regression from the Spider Toolkit[2]. Unfortunately, this regression function is malfunctioning and could not be used.[3] It was replaced by a Relevant Vector Machine [Tip00] based regression whose performance, both in terms of accuracy and completion time, seems to be not as good as the performance with the original multi ridge regression as shown in the lack of orientations found in the models presented in Figure 4.3a and Figure 4.3b.

[2] http://people.kyb.tuebingen.mpg.de/spider/

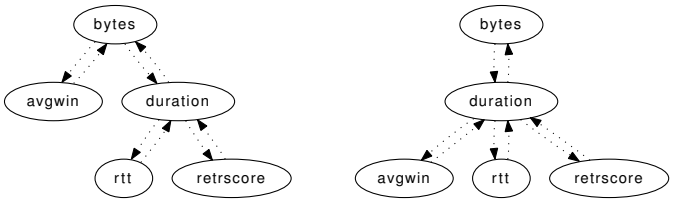
[3] Note that we contacted the person responsible of maintaining the Spider toolkit who indicated us that such malfunctioning was known and he suggested us to use a different non linear regression function instead.

In addition, observing the adjacencies only, all explanatory variables are independent one from another and they are all direct parents of the response variable, the duration (*duration*). Such models are not very informative and question both, the inference method and the input data.



(a) α = 0.05 (b) α = 0.1

Figure 4.2: Causal models of the performance of the upload of 900 MB object on Dropbox server, inferred using the PC algorithm for different significance level, α, in the independence tests



(a) α = 0.05 (b) α = 0.1

Figure 4.3: Causal models of the performance of the upload of 900 MB object on Dropbox server, inferred using the kPC algorithm for different significance level, α, in the independence tests

While the PC algorithm manages to infer, relatively, sensible graphs, the kPC is not able to perform the orientation phase of its graphical causal model inference. Assuming that the arrows should globally point to the response variable, *duration*, we can see that the graphs inferred by the kPC algorithm are not providing much information about the mechanisms ruling the Dropbox transfer performance.

4.3.4 Concluding remarks

The study presented in this section showed many limitations inherent to the causal study of a telecommunication network applications. As stated in the Introduction, Section 4.1, the study of the Dropbox application performance was a very preliminary study where most of the mechanisms presented in the previous Chapter are not implemented. In particular we can notice three important points.

First, because our objective is to design a simple study, the choice of a very limited set of parameters leads to models that are too simple to capture all the mechanisms influencing a Dropbox user quality of experience.

Second, we are here facing a case where the application has an important impact on the TCP throughput observed for a given connection. The Dropbox application will not provide more data to TCP unless the application acknowledgement for a given chunk has been received, preventing TCP to send more data even if the network could carry it. We try to take into account the impact of the application with the use of the Isolate and Merge algorithm to isolate the periods during which TCP and the network behavior are the only components impacting the user network performance. However, this approach is not accurate and fails in capturing the mechanisms used by the Dropbox protocol to control the rate at which data is exchanged between the Dropbox server and the Dropbox client.

Lastly, for this study we are only considering one file upload. As the file size is 900 MB, several parallel connection are opened between the client and the Dropbox server. The models inferred by both, the PC algorithm and the kPC algorithm, present graphs with few dependencies resulting in very uninformative models. We can see that, for the graphs inferred by the PC algorithm, it is necessary to use a high significance level (α) in the independence tests to detect dependencies between the parameters of our model that allow to obtain a consistent model. For low level of significance, we accept more independences that results in sparser graphs. Increasing the level significance allows the detection of weaker dependencies that make the orientation phase easier and provide more informative models where more of the expected dependencies can be observed. For the graphs inferred by the kPC algorithm, almost no dependencies between the explanatory variables is detected and the models we obtain are of very little use to better understand the roles of the different parameters of our model in the performance experience by the Dropbox users. These two remarks suggest that the dependencies between the different parameters of our system are too weak. It is clear that both algorithms do not manage to detect all the dependencies between the different parameters of our model. The range of situations that we observe does not present enough variability to exhibit the mechanisms ruling Dropbox performance.

In the next section, we try to solve these issues by defining metrics that better capture the Dropbox mechanisms and by observing the traffic corresponding to several uploads made at different times of the day, under different network conditions.

4.4 The study of Dropbox at the packet level

4.4.1 Presentation

We present a new characterization of Dropbox uploads traffic focusing our study to the packet level. Working at a lower level gives us a better understanding and modeling of the Dropbox protocol. We define metrics that show that Dropbox implements its own congestion control algorithm, that adapts the data exchanges to the traffic conditions (congestion, loss).

The upload of a file is done in one connection which is divided into periods of transfer and periods of silence. We call the transfer period a Dropbox Chunk (DC or chunks). A Dropbox chunk size is fixed (around 4MB). We call the silence periods between two Dropbox chunk transmissions: Inter Chunk Periods (ICP). If the file is too big there may be several connections as it was the case for the previous study where a 900MB file was uploaded. In this study, we upload files small enough to be uploaded in one connection, which does not prevent the generalization of the conclusions drawn from this study to bigger file uploads.

When studying the behavior of TCP packet exchanges during the transmission of the Dropbox chunks, we can see another subdivision which is not present in [DMMM+12]. We can observe, inside a chunk, a succession of transmission and silent periods during which no data is transmitted. The same way we defined Dropbox chunk, we define ***flights*** as the periods during which data is being transmitted, inside a chunk. We will simply name***inter-flights*** the periods between the transmissions of two consecutive flights where no data is exchanged between the client and the server.

4.4.2 Observations of the Dropbox traffic

Our study represents a 24 hours experiment. Each hour we upload a 200MB file to Dropbox with a 100MB Ethernet connection from EURECOM. For this study, we use some of the metrics defined in Section 4.3.

Because studying the performance of the Dropbox application at the Chunk level (or connection level) requires the understanding of the mechanisms that are used by the Dropbox protocol to control the sending rate we focus on the per-flight performance.

Therefore, we first divide the connections into chunks and then each chunk into flights. The Dropbox chunk detection is done with the detection of Dropbox application acknowledgement, symbolizing the end and good reception of the Dropbox chunk by the server. The detection of packets flights is done by observing the periods of silence between the transmission of groups of packets

and fixing a threshold based on the measured RTT. After a given time where no packet is sent, we consider that the next packet transmission that we observe belong to a new flight.

For each flight we record the following metrics

- **Beginning**: Time of the first packet of the flight
- **Ending**: Time of the last packet of the flight
- **Duration**: Difference between ending and beginning
- **RTT**: Time elapsed between the sending of a data packet by the client and the reception of the corresponding acknowledgement from the server, averaged among the packets of the flights
- **Loss**: Number of retransmitted segment divided by the total number of segments sent during the flights
- **Size**: Sum of all the bytes of data segments belonging to this flight
- **Inter-flight**: Difference between beginning of this flight and ending of the previous one (Zero for the first flight)
- **Throughput**: Average throughput obtained along the transmission of this flight
- **Chunkid**: The chunk it belongs to

Table 4.2 summarizes the different observations of the metrics we selected to model the per-flight throughput of the Dropbox traffic. Our dataset consists of 627 samples. Each sample represents a flight period during which the different parameters are measured. In Table 4.2 we present the average (Avg), minimum (Min), maximum (Max), standard deviation (Std) and coefficient of variation (CoV = $\frac{\text{Std}}{\text{Avg}}$) computed over the 627 flights periods.

4.4.3 Results

PC algorithm

Figure 4.4 presents the causal model inferred by the PC algorithm for two different levels of significance, used in the different tests that are performed during the inference of the Bayesian graph.

Figure 4.4a presents the causal model inferred by the PC algorithm for a significance level of 0.05. As mentioned in the previous section, lower values of

Table 4.2: Summary of the different metrics at the flight level for the hourly Dropbox uploads with their average value (**Avg**), minimum value (**Min**), maximum value (**Max**), variance (**Var**), standard deviation (**Std** = $\sqrt{Var}$) and coefficient of variation (**CoV** = $\frac{Std}{Avg}$)

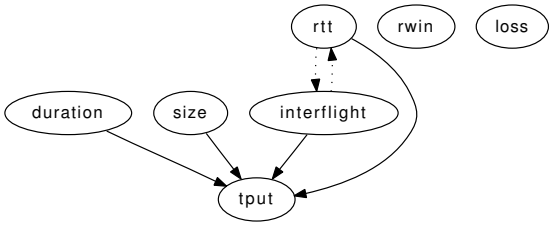
Parameter	Avg	Min	Max	Std	CoV
Duration (s)	0.05	0.02	0.91	0.06	1.1
Size (MB)	4400	1100	4500	410	0.09
RTT (ms)	100	98	110	2.7	0.03
Rwin (kB)	550	280	1400	150	0.3
Loss (%)	0.20	0	72	3.0	18
Inter-flight (s)	0.08	0.06	0.14	0.01	0.1
Throughput (Mbps)	1.0	0.12	2.12	0.3	0.3

the significance level in the independence tests result in sparser graphs. The Bayesian graph representing the causal model of our system when testing the parameter independences with α = 0.05 presents few dependencies between the parameters of our model. We can make the hypothesis that, again, the variations in the values of the parameters we observe do not present enough variability to be able to detect the parameter dependence with such significance level.

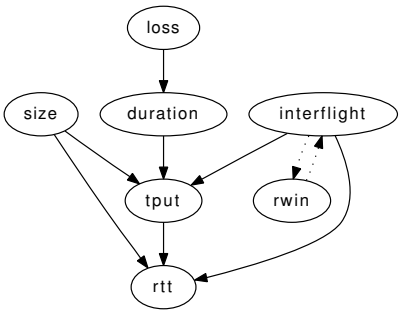
On the opposite, the Figure 4.4b presents the causal model inferred by the PC algorithm for a significance level of 0.1. Higher value of the significance level used in the independence tests result in more connected graphs. However, the Bayesian graph in Figure 4.4b exhibits counter-intuitive dependencies such as the throughput that is modeled as a parent of the RTT.

In our study, the system we observe does not present very important variations (see Table 4.2). The fact that the Z-Fisher criterion relies on wrong linear and normal assumptions might be less visible in the case where the dependencies are difficult to detect and a low value of α is used. However, the inadequacy of the Z-Fisher criterion appears more clearly when we increase the value of α and more dependencies are detected.

The models we obtain with the PC algorithm are not exploitable for two reasons i) Their simplicity suggests that some important dependencies are missing ii) When we increase the confidence level of the independence tests to detect weaker dependencies, we obtain inconsistent results both, when comparing the model obtained with α = 0.1 with the model obtained with α = 0.05 and when interpreting the dependencies obtained with a higher level of significance with our understanding of telecommunication networks.



(a) α = 0.05



(b) α = 0.1

Figure 4.4: Causal models of the per-flight Dropbox performance. Models inferred with the PC algorithm for different independence test significance level, α

kPC algorithm

Figure 4.5 presents the causal model inferred by the kPC algorithm for a independence test significance level of 0.05, Figure 4.5a, and 0.1, Figure 4.5b. We can observe that, for both levels of significance, the Bayesian graphs we obtain are very sparse, we cannot draw any interesting knowledge from these models. These results support the hypothesis made previously concerning the lack of variations of the observed parameters that makes their dependencies difficult to detect.

In the case of the kPC algorithm, the HSIC is used to test the parameter independences and should perform better than the Z-Fisher criterion. However, we could observe that the implementation of the HSIC present in the kPC performs very poorly. On the other hand, we are trying to model the performance of the TCP in the flights of packets, part of the Dropbox chunks, part of a user connection. The detection of dependencies at this level is expected to be more complex than when we study an application performance at the connection level directly, as it is the case for the studies presented in Chapter 5 and Chapter 6.

4.4.4 Concluding remarks

We would like to mention, first, that the first attempt to model the TCP performance at the connection level did not give good results. As the Dropbox

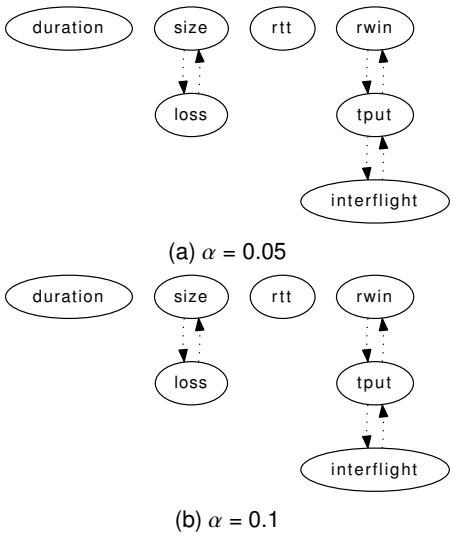


Figure 4.5: Causal models of the per-flight Dropbox performance. Models inferred with the kPC algorithm for different independence test significance level, α

protocol is dividing connections into silent and transmission periods, if we observe the performance by averaging parameters values on the connection (like the throughput) too many parameters and mechanisms are absent from the model (see the remarks made on the relationship between a parameter and the mechanisms it represents in Section 1.3.4).

In this second study, we tried to go deeper in the understanding of Dropbox mechanisms by defining, first, Dropbox chunks, based on the Dropbox protocol, and, then, packet flights. Such subdivision tried to remove the impact of the Dropbox protocol on TCP performance and to be able to study the impact of the network only. Unfortunately, this subdivision does not solve the issues met in the previous study and, worse, highlight additional problems.

First, the role of Dropbox in the inter-flight time is a parameter that has an important influence on the performance when studied at the flight level. Some tests where we observed Dropbox traffic when uploading files from a public area, using a Hotspot access, showed that Dropbox, in the case of a loss event, seems to implement its own recovery mechanisms. Additional studies where we observed the Dropbox traffic when uploading files from Boston, US, showed that the inter-flight is 10 times smaller for a RTT whose value is half of the one we observe in the previous two studies. Such observations suggest that our model misses important parameters to capture the mechanisms implemented by the Dropbox application to control the performance of its users.

Second, by studying a lower level performance (per-flight performance), the study of the dependencies becomes more complex and the variation of the observed parameters becomes smaller.

Finally, the global approach of "reverse-engineering" the Dropbox protocol to define the parameters that will capture its mechanisms is somehow necessary to define the parameter set defining our system, but this is not the final goal of a causal study. By increasing the granularity of our study, we obtain models that are not exploitable for predicting interventions that would explain the causal dependencies underlying the system performance. One important notion to keep in mind in a causal study is the one of *determinism* and its violation of the faithfulness assumption. Therefore, in addition to the study of mechanisms that might not be in line with our performance study, we might be trying to model deterministic mechanisms.

4.5 Lessons learned

The study of the Dropbox application performance was abandoned and no result regarding the Dropbox application performance or a way to improve it, was found. The study of the Dropbox application performance represents a very early stage of our research. However, such study is important for the lessons that could be learned regarding the design and realization of a causal study.

4.5.1 Causality is not always the good approach

In the case of Dropbox, it appears that the Dropbox protocol, whose mechanisms can be partially observed but are mostly unknown, is controlling the performance of the object uploads speed. In such situation, it becomes very difficult to create a model of the performance of the system as we miss the minimal quantity of information that allows us to design the system study. The first conclusion learned from this study, where no exploitable causal graph could be inferred, is the impossibility to model some systems with a causal approach in the case where the important parameters are absent. This absence can come from the fact that they cannot be measured or that they are not known.

In the case of the Dropbox application, it is the absence of knowledge of the Dropbox protocol mechanisms that prevented us from making the study of the Dropbox performance. It might be possible to include more parameters, infer the corresponding causal model and repeat the reasoning presented in this study until we obtain a coherent model. However, such approach is similar to reverse engineering and is not in our objectives.

We can summarize this section as: before starting a causal study, it is necessary to gather a sufficient knowledge that ensures us that we can infer a causal model for a given system and that we will benefit from such model. In the case of the Dropbox application performance study, too many parameters and mechanisms are unknown.

4.5.2 Degree of abstraction

As we said in the previous section, the objective of a causal study is not in reverse engineering the system that is being studied. As we are studying applications and protocols, such systems are implemented in the different machines running them. Causal studies are based on statistical considerations. Therefore, a causal approach is neither designed nor efficient in uncovering the deterministic mechanisms implemented in the different components that form the distributed system that is being studied.[4]

As we mentioned in the Introduction chapter, Chapter 1, we are interested in the study of the performance of applications relying on TCP protocol. We could then ask ourselves if the problem met with the Dropbox study might not be the one we would meet in the study of any other systems that relies on TCP. Fortunately, the answer is no, but the Dropbox study obliged us to observe Dropbox performance at lower level than the one of TCP connections.

The second lesson learned from the Dropbox study is the importance of the choice in the degree of abstraction of our model. We must find a trade-off between the abstraction of the mechanisms and the correct representation of such mechanisms in our model, i.e. the set of parameters we choose to describe our system.

4.5.3 Complexity and completeness

Even if we find a correct level of abstraction of the system mechanisms that allows us to model the system performance and use this model to improve the system performance, an additional problem was suggested by the Dropbox application study that is the one of feature selection.

As we saw in the Dropbox study, our choice of a reduced set of parameters to obtain a simple model of Dropbox performance failed. To obtain a usable model we need a set of parameters where each mechanisms impacting the system performance is represented by at least one parameter. As we could also observe in this chapter, this is only possible with enough knowledge of the system mechanisms.

In the Dropbox study, we used two algorithms, the PC algorithm with the Z-Fisher criterion and the kPC algorithm with the HSIC. The Z-Fisher criterion has a very simple and fast implementation in different programming languages, while the kPC algorithm implements independence test based on kernel that

[4]There exist works to infer a causal model in the presence of deterministic dependencies [Bau09]. However, such works show that such studies become more complex and, therefore, are not tackled by our work.

is highly resource demanding, with a time that increases very fast with number of independence tests and the size of the conditioning set in the independence tests. In the two following chapters, we use the KCI+bootstrap procedure presented in Section 3.2.2. In this case the completion time and resource constraints for inferring models with an important number of parameters is an important constraint that impacts the design of our causal study.

The third lesson learned from the Dropbox study is related to feature selection. We need to find the correct trade-off between an important number of parameters, that makes the causal model inference impossible in practice, and a small number of parameters, that simplifies the causal model inference but increases the risk of being unable to capture important mechanisms and of obtaining incomplete models.

4.5.4 Design of the experiments

The important difference between the first study, Section 4.3, and the second one, Section 4.4, comes from the range of situations that we observe. In the first case, we observe a single upload while in the second case we spread our observations on a full day during which several uploads are observed. The second experiment tries to avoid the limitations highlighted by the first one. Such observation concerning the experiment design might seem naive but plays a very important role for both the model inference and the predictions that can be made from the inferred model, as we will see in the study of the DNS impact on CDN object access performance in Chapter 6.

The experiment during which we observe the system that we want to model should capture enough situations to allow the observation, and then inference, of the different mechanisms impacting the performance of the system. On the other hand, if we increase the range of situations under which the system is observed, we increase the risk of actually observing very peculiar situations where the parameters we use to model the system mechanisms are not sufficient.

Hence, an additional lesson learned, that will appear even more clearly in the next two studies of Chapter 5 and Chapter 6, is the importance of the experiment design. While one should keep in mind that it is impossible to observe all the possible situations in a limited amount of time, it is important to observe the system under many different situations. This remark is important to detect the causal dependencies between the parameters of the system that we are observing and for the predictions of interventions.

4.5.5 Preprocessing of the data

The last point, that is very well known in the domain of machine learning and data mining, is related to the preprocessing phase. In the previous section, we insisted in the importance to observe different situations. However, it is also important to know what is present in your observations and how to structure your observations in a set of parameters that correctly captures the different situations that were observed.

In the case of the Dropbox study, the naive approach where we tried to model the TCP throughput at the connection level led to totally inconsistent models. It is the observation of the connections at a packet level that made us aware of the inadequacy of our model to capture the Dropbox performance. The same considerations have been necessary in all the studies that were made in our work.

The most important lesson that we can make after the study of the Dropbox performance is to dedicate enough time in the understanding of the system measurements constituting the basis of our study. The model inference and intervention predictions rely on the input data and error of measures or representations may have a very negative impact on all the steps of the causal study of a system.

However, the representation and summary of a highly dimensional dataset is a very complex problem for which there does not exist a universal solution.

Chapter 5

Study of the File Transfer Protocol

In the previous chapter we presented the study of the Dropbox protocol where we could see all the challenges that need to be overcome to correctly perform a causal study. The Dropbox study presented the results that one would obtain when trying to apply the existing tools and theories in a naive way. In this chapter we follow the opposite direction by carefully validating the different steps used in our causal approach to study the performance of the FTP users.

We first validate our methods with the study of the performance of an emulated network. In an emulated environment we can manually perform the interventions that our causal study allows us to predict and compare our prediction with a ground truth. In Section 5.2, we apply the methods validated in Section 5.1 to the study of the File Transfer Protocol (FTP) application performance with file transfers happening over the whole Internet.

5.1 Emulated network

In this section, we validate the approach described in the Chapter 3 in the context of an emulated network. We use the Mininet emulation tool [LHM10], which allows us to perform the interventions that our causal study allows us to predict and to study the accuracy of our predictions.

5.1.1 Mininet network emulator

Mininet is a powerful network emulator based on Container Based Emulation (CBE) [HHJ+12]. It offers the possibility to run, on a single machine, a realistic

network emulation with real kernel, switch and application code. An emulator makes it easy to recreate different network conditions and manipulate different parameters to control the performance of the network. Mininet gives us access to four parameters to control the delay, the bandwidth, the loss and the delay jitter (= delay variations) of each link connecting two nodes in our network. In addition, each router has a buffer size that can be controlled to obtain different network behaviors.

However, like any emulator, Mininet has its own limitations, which we will discuss later in this section when we present and interpret the results. Nevertheless, using this emulator serves well our purpose, which is validating the application of causal modeling to network performance prediction.

5.1.2 Experimental setup

Figure 5.1 presents the emulated network we use for our experiments. The traffic is recorded at the FTP sender hosted by **S1** using the Tcpdump tool [Fre09]. The traffic consists of the different file downloads issued by the client **C1**. The packets are going through the link **L1**, the router **R1**, the link **L5**, the router **R2** and the link **L2**, allowing us to manipulate the different parameters along the path. To make the network traffic more realistic we add a server **S2** serving requests from a client **C2** and generating cross traffic that will interfere with the transfers between **C1** and **S1**.

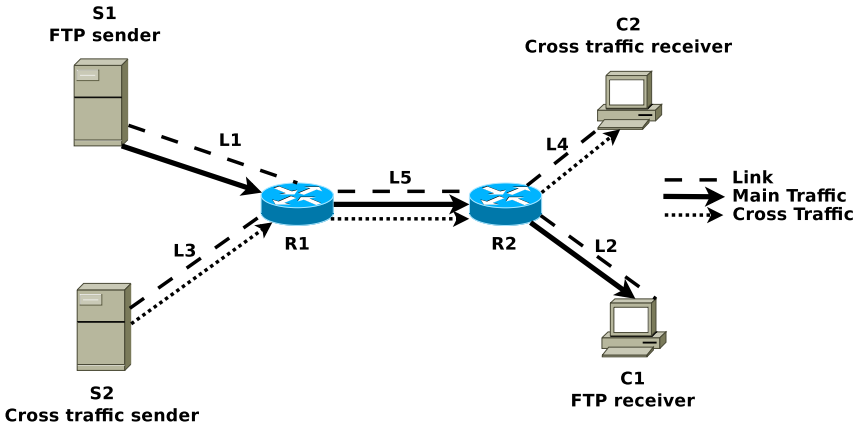


Figure 5.1: Emulated network using Mininet

Once we have captured the packet trace, we use Tstat [MLCN05] and Intrabase [SBUK+05], to compute the different metrics summarized in Table 5.1. The *delay* parameter represents the propagation delay between **S1** and **C1**, i.e., the time for a bit to propagate, at a speed close to the speed of light, along the links connecting the two end-hosts. *Delay* does not take into account the impact of buffering. The loss event parameter (p) is closely related

to the retransmission score (*retrscore*). *Retrscore* represents the fraction of retransmitted packets and is an approximation of the loss probability. Instead, p approximates the probability of a loss event. As losses in networks often occur in bursts (buffers, when overflowing, drop packets), an important parameter to observe is the occurrence of loss events which is approximated by p.

We record the following parameters:

- Bandwidth: The maximum capacity of the path between the client and the server [*bw*]
- Propagation delay: The time, when no queuing happens, for a packet to go from the server to the client [*delay*]
- Size of buffer: The maximum number of packets that can be stored by a router, R_X, in case of congestion [*queueX*]
- Narrow Link Available Capacity: The estimated capacity of the path between the client and the server that the client has access to (takes into account cross traffic) [*buffersize*]
- Receiver Window: The client advertised receiver window, which captures the amount of packets that the client can process [*rwin*]
- Buffering Delay: The fraction of the time it takes for a packet to cross the network that is due to queuing in router buffers [*bufferingdelay*]
- Round Trip Time: The time it takes for a packet to cross the network (and be acknowledged) [*rtt*]
- Timeouts: The number of retransmissions triggered at the server for time out reason (no acknowledgement received) [*timeouts*]
- Retransmission score: The fraction of packets that had to be retransmitted by the server (approximate the loss frequency) [*retrscore*]
- Probability of a loss event: The fraction of loss events, where a loss event is the occurrence of a burst of packets being lost (dropped by a router) [p]
- Number of bytes: Total amount of bytes sent by the server to the client [*nbbytes*]
- Throughput: Amount of bytes that the server was able to send in a given amount of time to the client [*tput*]

Table 5.1: Summary of Mininet network emulation experiments dataset

Parameter	Definition	Min	Max	Avg	CoV
bw	minimum bandwidth (MBps)	1	25	7.1	0.91
delay	propagation delay (ms)	30	180	86	0.48
queue1	size of **R1** buffer (pkts)	10	400	98	1.10
queue2	size of **R2** buffer (pkts)	10	400	100	0.99
nlac	Narrow Link Available Capacity (kBps)	12	3070	630	5.00
rwin	Receiver window advertised by **C1** (kB)	74	2155	288	0.65
bufferingdelay	part of the RTT due to queuing delay (ms)	1	6760	120	2.40
rtt	Round Trip Time (ms)	84	6910	310	0.99
timeouts	number of timeouts (units)	0	682	79	1.50
retrscore	fraction of retransmitted packets (no unit)	0	0.61	0.04	5.10
p	fraction of loss events (no unit)	0	0.64	0.04	8.40
nbbytes	number of bytes sent by the server (MB)	6	150	110	0.21
tput	throughput (kBps)	6	1100	280	0.81

5.1.3 Causal model

The causal graph of our emulated FTP traffic is presented in Figure 5.2. It was inferred using the PC algorithm with the KCI test, implemented with the bootstrap method (Section 3.2), for testing independence. We observe that the graph is globally in good accordance with the domain knowledge of how the different parameters impact the performance of TCP.

We see in Table 5.1 that the values of *retrscore* and p are very similar. This is due to the definition of p and a possible limitation of Mininet in the way the packets are dropped: the difference between p and *retrscore* is mainly due to the burstiness of losses happening in the network. Mininet uses internally Netem [AJPMI11] to simulate losses or the drop of packets by the router, which seems to drop only one packet at the time. This limitation could explain the difference in the orientation found between p and *retrscore* in the model presented in Figure 5.2 (*retrscore*$\rightarrow p$) and the orientation found in the next study of the FTP traffic performance, when exchanges are made over the Internet network ($p\rightarrow$ *retrscore*, see Figure 5.10 in Section 5.2.2).

The models presented in Figure 5.2 and in Figure 5.10 both indicate that *rtt* impacts p. This can be understood as follows: an increase in *rtt* is causing an increase in the burstiness of the sending process as the time spent in the network increases and so the probability that consecutive packets meet a full queue and dropped, which results in burst loss. On the other hand, the presence of a causal relationship between p and *retrscore* is quite obvious, while its

orientation is harder to explain, in particular when both parameters have very similar variations.

Taking into account the remark made previously about *retrscore* and p, we can see that the *rtt* and *retrscore* are two of the parents of *tput* (*throughput*). The third parent of *tput* is *rwin* (the *Receiver Window*) which represents the capacity of the receiver to handle the quantity of data that is sent by the Server. The PFTK model (cf Equation (3.1), Section 3.2.1) assumes that the TCP performance is not "receiver limited" and ignores the parameter *rwin* in Equation (3.1). In the model of Figure 5.2 all the empirical parameters that are known to impact TCP throughput are correctly detected and shown as the direct parents of *tput*. In the following section, we present in Figure 5.3 the model obtained using the IC* algorithm [Pea09] which indicates that there are no latent variables between the three parameters, *rtt*, *retrscore*, *rwin*, and the parameter *tput*.

The time a packet needs to cross the network is mainly influenced by the time it spends waiting in buffer queues. This fact is captured by the graph in Figure 5.2 where the *bufferingdelay* is the unique parent of *rtt*.

Finally the existence of a dependence between the *Bandwidth* (*bw*) and the *Delay* (*delay*) is clearly the effect of the experiment design. In our simulations we vary the *Bandwidth* and the *Delay* to observe a broader range of situations and the choice of the values for these parameters is not totally random. The dependence found between *bw* and *delay* shows that the experiment setup creates a wrong association between these two parameters. However, it should not influence the validity of the model (the bandwidth is an exogenous parameter). On the other hand, it is interesting to see that the causal model correctly captures this dependence. We repeated the experience but, instead of selecting values for the parameters for the emulated network components likely to create congestion (such as the delay, the bandwidth and the buffer sizes of the routers), we select values for the parameters of the simulation randomly. The corresponding model is presented in Figure 5.8 and the such dependence is not present anymore. However, as we are still limited in the number of simulations that can be made, the dataset we obtain presents less variability and the causal model we obtain is less informative.

Other causal models

In this section, we present other causal models that we obtained by using either a different constraint based algorithm or a different independence criterion.

First, we present the causal model obtained when using the IC* algorithm [Pea09]. The IC* algorithm is able, to some extent, to detect the presence of a latent parameter in the set of parameters representing our system. As a reminder, a latent variable is a variable that is not observed but that impacts (at least) two

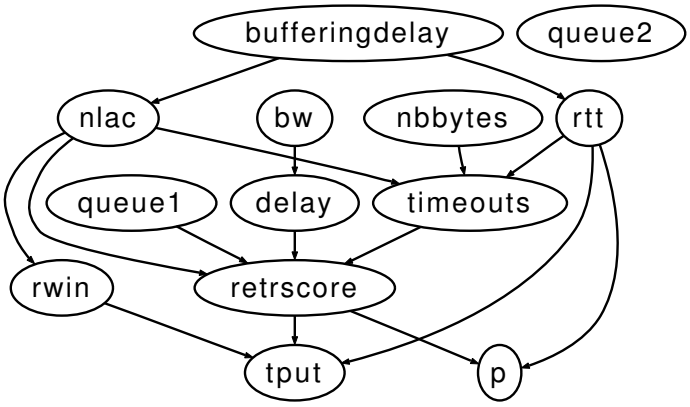


Figure 5.2: Causal model inferred by the PC algorithm, with KCI test, for the emulated network traffic

of the observed variables. The presence of latent variables usually makes the causal study of a system more challenging for mainly two reasons (i) the detection of conditional independence is made sometimes impossible (if L is a latent variable such that $X \leftarrow L \rightarrow Y$, it is complicate to test independence between X and Y) (ii) the presence of latent variables makes more difficult the prediction of intervention as we might no be able to block spurious associations. Eventually, to close this parenthesis on latent variables, it is the absence of a latent variable in our model that makes the causal study challenging, even if we cannot label or observe a latent variable in our system but it is present in our model, then the use of the do-calculus is still valid. Causal models are solving the problem of spurious association and models where latent variables are present are perfect cases where the causal approach is very effective and useful, blocking the influence of unobserved variable on the system performance to better predict the effect of intervening on observed parameters.

Second, as mentioned in the presentation of the different algorithms for causal model inference, Section 1.3.4, one of the algorithms we initially tested was the kPC algorithm [TGS09]. The kPC algorithm is of particular interest for our work as it does not rely on linear or normal assumptions. However, through the exposition of the model we obtain when we use the kPC algorithm to model the performance of an FTP client in the emulated network presented in Figure 5.1, we see that the kPC algorithm performs poorly.

Third, we present results obtained when we use the Z-Fisher criterion instead of our KCI+bootstrap solution to test for parameter independences in two different constraint based algorithms, the PC algorithm and the FCI algorithm [SGS01]. The FCI algorithm is similar to the IC* algorithm as it is able, to some extent, to detect the presence of latent variables. The tests with the Z-Fisher criterion are important in our work as they represent the test that is usually used in constraint based algorithm for causal model inference. By showing the (bad)

models that we obtain when using a criterion based on the wrong assumptions, namely linearity and normality, we justify the important of work that was made to design an independence criterion that fits our data constraints. As a side remark, the models we obtain when the Z-Fisher criterion is used also support the fact that the work closer to ours, the WISE system [TZV+08], cannot work for our telecommunication network studies.

Eventually, to answer the remark made on the possible dependence that we created in the experiment design, we present the causal model that we obtain if repeating our experiment but choosing the values of the simulations parameters randomly.

IC* Figure 5.3 presents the model inferred using the IC* algorithm [Pea09] to model the FTP client throughput in the emulated network. The graph output by the IC* algorithm may contain four types of edges:

- A marked arrow, signifying a directed path from X to Y, in the underlying model.
- An unmarked arrow, signifying either a directed path from X to Y or the presence of a latent common cause, L, of X and Y, in the underlying model.
- A undirected edge signifying the presence of a latent common cause of X and Y, in the underlying model.
- An bidirected edge signifying any of the previously mentioned possibilities plus the possibility of a directed path from Y to X, in the underlying model.

The original model that we obtain shows a bidirected edge between the $NLAC$ and the $bufferingdelay$, that gives no information on this dependence. However, it can be noticed that orienting this edge in one direction or another does not change the Markov Equivalence Class of the model and may be the reason why it cannot be oriented. Therefore, we use our domain knowledge to orient this edge. Another property of this model is that all the edges going into $tput$ are marked edges, showing that the parameters we observe are all direct causes of the $throughput$ ($tput$), which supports our choice of intervention in the following section (Section 5.1.4).

kPC Figure 5.4 presents the model inferred by the kPC algorithm when applied to the emulated network scenario. We can see two important negative points in this model. The first one being that, globally, this model gives very little information. Most of the parameters known to be impacting the throughput are

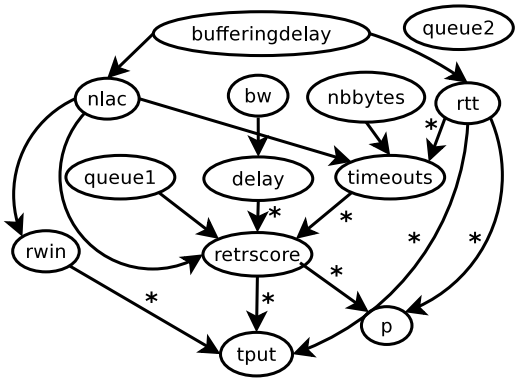


Figure 5.3: Model for the network emulation data using the IC* algorithm

independent one from another and the edges connecting them to the throughput are left unoriented (the same observation was made in the study of the Dropbox application performance in Chapter 4). This last observation brings us to the second point which comes from kPC algorithm orientation phase. The kPC algorithm highly relies on the non linear and non parametric regression. To orient an edge between two parameters found to be dependent it tests the independence between the regressor and the residual for the two possible orientations. If one orientation leads to the independence of the residual on its regressor and the other orientation does not, then the edge is oriented accordingly. Otherwise the edge is left unoriented. This model shows the difficulty that exists to model the non linear dependence between two parameters.[1].

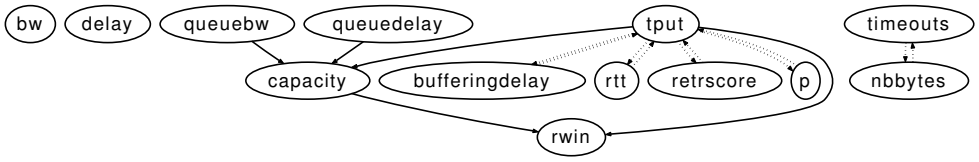


Figure 5.4: Causal model inferred by the kPC algorithm for the emulated network scenario

Z-Fisher criterion Figure 5.5 represents the model inferred by the PC algorithm when we use the Z-Fisher criterion instead of our modified version of the KCI test.

We first notice that the model of Figure 5.5 is quite different from the one presented in Figure 5.2 where the KCI+bootstrap is used. The absence of *retrscore*

[1] It is important to remember that the kPC algorithm implementation from [TGS09] originally uses Support Vector Regression but due to error in its implementation from the Spider toolbox, Multi Ridge Regression is used instead. Such change seem to have a negative impact on the accuracy of the algorithm and the completion time.

as a parent of *tput* is an important dependence that is not present here. The path $bufferingdelay \rightarrow bw \rightarrow delay \rightarrow queue2 \rightarrow queue1 \rightarrow timeouts$ does not find any explanation from our domain knowledge of the TCP mechanisms and cannot be explained, as in the previous case, by the nature of experimental design. While the model inferred using the KCI test presents properties and dependencies that are supported by the domain knowledge of TCP performance, the model inferred with the Z-Fisher criterion does not (for an experimental comparison of the two criteria we refer the reader to Appendix A.1).

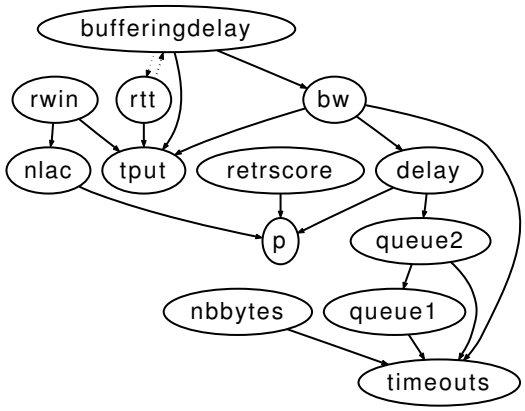


Figure 5.5: Causal model inferred by the PC algorithm, with Z-Fisher criterion, for the emulated network traffic

The study [VD08] showed that the Z-Fisher criterion can still perform correctly in the case where the parameters being tested are not normally distributed. However, this study also showed that the Z-Fisher criterion suffers from non linearity. As consequence, Figure 5.6 presents the causal model obtained with the PC algorithm and Z-Fisher criterion when applying a log-linear transform to the dataset presented in Table 5.1. Based on the Equation (3.1), applying a log transformation to the data could have could have brought the dependencies closer to linearity but the graph we obtain tends to disagree with this hypothesis. It can be observed that some independences seem to be better captured by this model, when compared to the independences present in Figure 5.5. However many of them are not in line with the mechanisms ruling the behavior of TCP. The throughput (*tput*) causing the receiver window (*rwin*) is one example, and the *delay* causing p another. We can also notice the presence of a cycle ($tput \rightarrow rwin \rightarrow nlac \rightarrow tput$). The model we obtain after a log transform of the parameters of our system shows that the dependencies between the different parameters cannot be modeled by simple models such as the PFTK model (Equation (3.1)). Note that the same study from Padhye et al. [PFTK98] extends the model presented in Equation (3.1) to take into account the possible receiver limitation, introducing in the throughput model a *minimum* function between the right side of Equation (3.1) and an estimate of the client available

storage. Such extension suggests that log-linear transform will not succeed.

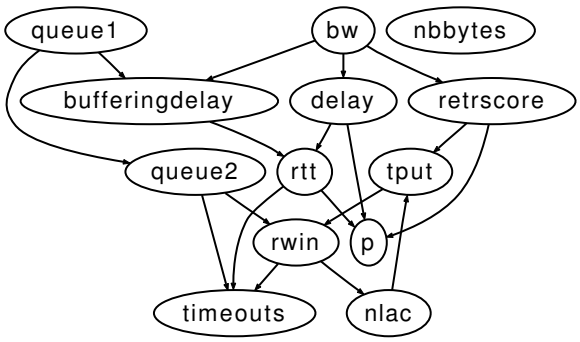


Figure 5.6: Model for the network emulation data after applying log-linear transform, using Z-Fisher criterion

Finally, we present the model obtained with a different software, the R-package Pcalg that also uses the Z-Fisher criterion to test the different independences and build the Bayesian network corresponding to the causal model of the system being studied. The use of the Pcalg software allows to test another implementation of the Z-Fisher criterion with the FCI algorithm [SGS01]. The FCI algorithm, by opposition to the PC algorithm, does not return a DAG (or more precisely an equivalence class of DAGs represented by a Partial Ancestral Graph (PAG)) but a Maximum Ancestral Graph (or more precisely an equivalence class of MAGs that can be represented by a PAG) with the following edges $\circ\!\!-\!\!\circ$, $\circ\!\!-$, $\circ\!\!\rightarrow$, $\longrightarrow$, $\longleftrightarrow$, $-$, with the following interpretations

- There is an edge between X and Y if the corresponding parameters are conditionally dependent for any set of variables containing all selection variables[2] and a subset of observable variables.
- The presence of a tail means that the tail is present in every MAG of the equivalence class (in $X \rightarrow Y$, the left part of the arrow leaving from X is called tail, in the absence of a tail we would have $X \circ\!\!\rightarrow Y$)
- The presence of an arrow head means the arrow head is present in every MAG of the equivalence class
- A $\circ$ edgemark means that there is at least one MAG where this edgemark is an arrowhead and one MAG where this edgemark is a tail

Roughly, the presence of a undirected edge indicates the presence of a possible selection variable and a bidirected edge the presence of latent variable.

[2]selection variables are unobserved parameters which values condition the observation of the sample, f.ex the connection was not aborted.

Figure 5.7 presents the model inferred using the FCI algorithm with the Z-Fisher criterion. As it was already observed, Z-Fisher does not seem to be able to correctly capture the dependencies between the different parameters. If observing the subgraph $nlac \circ\!\!\rightarrow p \leftarrow\!\!\circ retrscore$, the presence of $\circ$ edgemarks in V-structure with a collider, p, is misleading but comes from the use of a criterion called possible-d-separation, which extends the d-separation criterion presented in Section 1.3.4. As this criterion tends to consider a bigger set of possibilities, by having an inaccurate independence criterion, it leads to a less informative model.

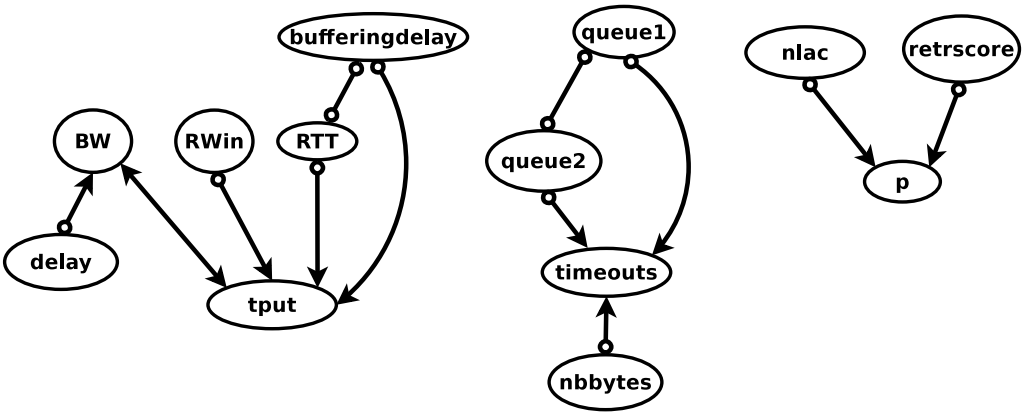


Figure 5.7: Causal model inferred by the FCI algorithm and Z-Fisher criterion for the network emulation data

Randomization of the emulation parameter values Finally, to answer our remark on the dependence between the bandwidth (bw) and the propagation delay ($delay$) in the causal model presented in Figure 5.2, whose presence was attributed to the experiment design, we repeated the simulation presented in Section 5.1.2 but we select the values of the parameters of the simulation (delay, jitter bandwidth, buffer size) randomly. We randomly pick a value for the different link capacities, delays between two end systems and buffer sizes at the two routers, and create a network. For each network configuration, **C1** makes several FTP downloads from the server **S1** where we record the traffic and observe the values of the different parameters of our system.

The corresponding dataset is presented in Table 5.2 and the corresponding causal model in Figure 5.8. We can observe that the graph is lacking many important edges when compared to the one obtained with no randomization (Figure 5.2). In the experiment we set up in Section 5.1.2 we chose the values of the parameters in order to create situations that exhibit some TCP properties due to network congestion or application limitations. Here, by randomly selecting the values of these parameters, these situations happen less often and the dependencies are more difficult to detect. Due to time constraints we could not

test a bigger range of network scenarios. While many edges we found in the model presented Figure 5.2 are not present in Figure 5.8, we can still observe the parameters p, rtt and $rwin$ as direct parents of the throughput ($tput$). We also observe $delay$ and $bufferingdelay$ as direct parents of the rtt. Eventually, the dependence between the bandwidth (bw) and propagation delay ($delay$) is not present anymore.

Table 5.2: Summary of the randomly emulated network dataset

Parameter	Definition	Min	Max	Avg	CoV
bw	minimum bandwidth (MBps)	1	25	6.7	0.77
delay	propagation delay (ms)	50	170	96	0.37
queue1	size of **R1** buffer (pkts)	10	400	98	1.1
queue2	size of **R2** buffer (pkts)	10	400	86	1.0
nlac	Narrow Link Available Capacity (kBps)	39	1.06e+5	1.15e+5	2.5
rwin	Receiver window advertised by **C1** (KB)	69	793	201	0.58
bufferingdelay	part of the RTT due to queuing delay (ms)	0.43	288	38.2	1.1
rtt	Round Trip Time (ms)	82	1074	221	0.67
timeouts	number of timeouts (units)	0	4	272	2.0
retrscore	fraction of retransmitted packets (no unit)	0	0.7	0.006	4.2
p	fraction of loss events (no unit)	0	0.38	0.004	5.1
nbbytes	number of bytes sent by the server (MB)	70	154	110	0.21
tput	throughput (kBps)	41	797	228	0.7

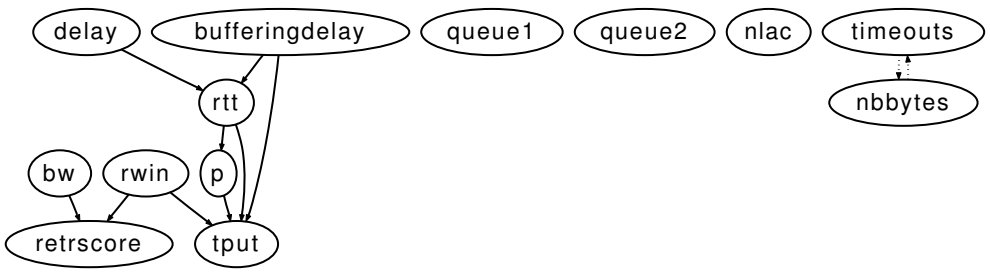


Figure 5.8: Causal model obtained for the emulated network scenario when randomization is used

For the rest of our study of the emulated network traffic, our causal model is the one inferred by the PC algorithm with the use of the KCI+bootstap independence test, Figure 5.2.

5.1.4 Predictions

In this section, we validate our approach to predict the effect of an intervention. For this purpose we compare the prediction we obtain with the back-door criterion (Section 1.3.4) for the dataset Table 5.1 (and in the associated graph of Figure 5.2), to the performance observed in a controlled emulation where the same intervention is actually made. Given the model shown in Figure 5.2 and the interventions that we can perform, the best parameter to intervene on is the retransmission score (*retrscore*). The retransmission score can be well approximated by the sum of the percentage of packets dropped at the routers $\mathbf{R_1}$ and $\mathbf{R_2}$, a parameter we can manipulate in our emulated network. Our dataset consists in 1100 connections out of which only 355 connections experienced a loss. For 353 of these connections the recorded *retrscore* is 0.01 (1%). As it can be seen from the back-door criterion, Equation (1.8), Section 1.3.4, it is necessary to have, in our dataset, samples where the value at which we want to set the parameter of intervention is present. As the value of 1% for the *retrscore* is the most commonly observed, for this scenario, we will perform an intervention on the loss rate (*retrscore*), set its value to 1% and estimate the impact of this intervention on the performance of our system (represented by the throughput).

To estimate the distribution of the *throughput* after intervention we apply the back-door criterion (cf Equation (1.8) Section 1.3.4) with the blocking set $Z = \{nlac, rtt\}$. This criterion ensures the identifiability of the causal effect representing the intervention on the retransmission score and, applying the back-door adjustment, provides us the way to compute the post intervention distribution of the throughput. The result we obtain is an expected value of the *throughput* post intervention of 82.67 kBps. The one observed prior to the intervention was 280 kBps, where most of the connections did not experience any loss.

To validate our approach we make new experiments with Mininet where we intervene on the retransmission score that we fix to 1%, which is very easy to configure in Mininet. The other parameters are unchanged and are the same as for the pre-intervention scenario. Now, the average *throughput* we obtain is 98.2 kBps. This value is relatively close to the value predicted using the back-door criterion, which validates our approach.

To better appreciate the benefits of the causal approach, we compare it with the "naive" approach. In the naive approach, we select in our initial dataset the values of *throughput* corresponding to the observations where 1% of loss was observed, in which case we obtain an expected *throughput* of 133.09 kBps. This means that the naive approach significantly overestimates the post-intervention *throughput*. The naive approach is equivalent to conditioning on *retrscore*. However, as we explained in Section 1.3.4, we must condition on non collider nodes to block paths creating spurious associations (for example $tput \leftarrow retrscore \leftarrow \ldots rtt \rightarrow tput$). If the backdoor paths between *retrscore* and

throughput are left unblocked, the expected throughput we obtain also takes into account the effect of other variables, such as *rtt* or *nlac*, on *retrscore*. As presented in Section 1.3.4, an intervention on the retransmission score (*retrscore*) consists in isolating *retrscore* from the influence of its direct and remote causes and in fixing its value (or distribution) as defined by the intervention. In this case the impact of parameters such as *rtt* or *nlac* must still be taken into account, but their impact on the variable of intervention, *retrscore*, is spurious and must neglected, which is implied by the backdoor formula. This result illustrates the superiority of causal approach over purely statistical ones. A study based on correlation where one, by finding the throughput and retransmission score correlated, would adopt the "naive approach" to predict the effect of modifying the retransmission score would obtain a wrong result. The causal approach detects and blocks the spurious associations and allows for a correct estimation of the impact of interventions.

Figure 5.9 presents the probability density functions corresponding to the throughput prior to intervention (in dash-dot line), the post-intervention throughput estimated with the back-door criterion (dashed line) and the throughput observed when we manually modify the loss (solid line). Again, while not perfectly matching, the graph-based prediction (dashed line) and the real density after manual intervention (solid line) show similar shapes. In particular, both densities have a single mode around the same value ($\sim$ 100 kBps). The fact that the experimental throughput values after intervention (solid line) show less variance can be explained by the small number of samples available to make the estimation. Also our method to estimate the post-intervention throughput (dashed line) uses normal kernels and a T-copula, which tend to widen and smooth the estimated post-intervention density.

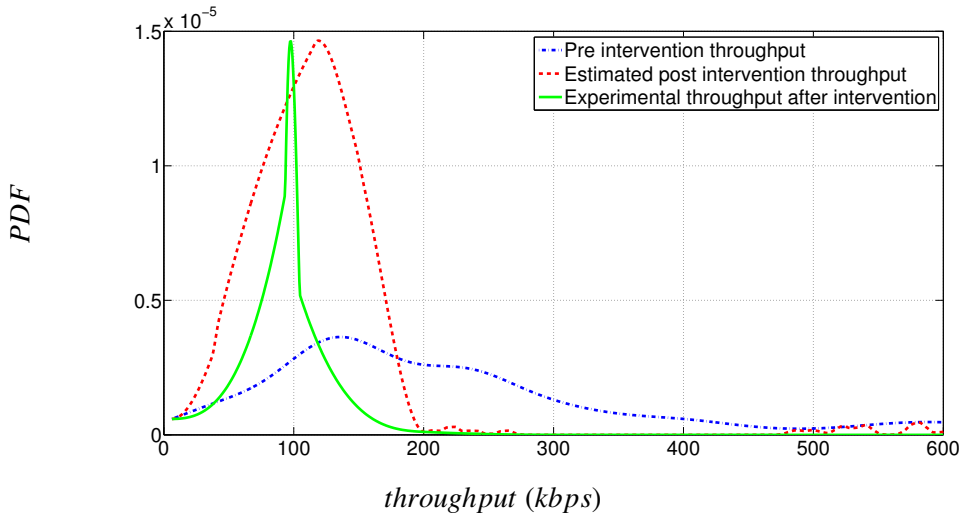


Figure 5.9: Effect of an intervention on the retransmission score, on the throughput distribution

5.1.5 Concluding remarks

In this section we presented the study of the performance of an emulated network that aims to validate both our approach and its parameterization using a network emulation system called Mininet where the prediction of an intervention could be tested. Despite the limitations of Mininet, the comparison between our prediction and the actual intervention validates our causal approach and demonstrates its usefulness. In the next section, we will apply our causal approach to real traffic data sent over the Internet.

The possibility to verify our prediction is a key to validate our methods. The parameterization of the independence test, of the choice of kernel to estimate the marginals and the choice of copula to model the multi dimensional distributions of the different parameters can only be judged from the accuracy of the prediction. The importance and difficulty to have a ground truth to which we can compare our predictions was already mentioned in the parameterization of the KCI+bootstrap procedure, Section 3.3.2.

5.2 FTP traffic

We set up a real FTP server at our institute where we record all the incoming and outgoing traffic. The FTP clients are machines located somewhere else in Europe that download files from the FTP server. This scenario is more complex than the emulated one since more factors are unobserved, such as all the ones related to buffering and the network load, and observations are also subject to more variations, e.g., in the path the packets travel.

5.2.1 Experimental set up

In this experiment we set up an FTP server in EURECOM in the South of France where all the traffic is recorded using the Tcpdump tool. Different clients, from France, Germany and Spain are downloading several files, of different sizes and at different times, from that server. Using the Tstat and Intrabase tools we compute from the packet traces the different metrics presented in Table 5.3.

5.2.2 Causal model

Using the PC algorithm with the KCI test, implemented with the bootstrap method, Section 3.2, we obtain the graph presented in Figure 5.10.[3] Below

[3]For more insights on possible latent variables we refer the reader to Figure 5.11, in Section 5.2.2, which presents the output of the IC* algorithm [Pea09] where we can see that almost all parents of $tput$ have marked directed edges, excluding the possibility of latent variables

Table 5.3: Summary of FTP traffic dataset

Parameter	Definition	Min	Max	Avg	CoV
dist	distance between server and client (km)	14	620	250	0.95
tod	time of the day (s)	740	82000	46000	0.53
nbbytes	number of bytes sent by the server (MB)	6	60	32	0.51
nlac	Narrow Link Available Capacity (kBps)	48	43000	5900	1.70
nbhops	number of hops from server to client (hops)	9	27	11	0.24
rtt	Round Trip Time (ms)	60	710	270	0.76
bufferingdelay	part of the RTT due to queuing delay (ms)	4.2	470	84	1.20
retrscore	fraction of retransmitted packets (no unit)	0.001	0.01	2e-3	0.92
p	fraction of loss events (no unit)	3e-5	0.01	7e-4	1.30
tor	fraction of *retrscore* due to timeouts (no unit)	0	0.01	6e-4	1.70
rwin	Receiver Window (kB)	11	254	137	0.68
tput	throughput (kBps)	24	928	332	0.77

we discuss the most important features of the graph.

First, we see that the graph is fully coherent with the domain knowledge of TCP. In particular, the parents of the *tput*, namely *{p,rtt,rwin}*, are the empirical parameters influencing the *throughput* and {*p*,*rtt*} are present in the PFTK model (*rwin* is not).

Another interesting outcome is that the timeouts, once normalized by the number of packets (*tor*), do not have an impact on the *throughput* (*tput*). Consequently, intervening on this parameter will not lead to a direct improvement of *throughput*. Intervention on a parameter can be seen as disconnecting this parameter from all its parents and creating a new parent that represents our intervention. As there is no path between *tor* and *tput* in the graph $G_{\overline{tor}}$ (the modified graph where all the edges entering the *tor* node are removed) there will be no effect on the throughput if we were to intervene on the *tor* parameter.

The causal graph shows a dependence between the RTT and p. Such dependence was already present in the model inferred in the previous study, Figure 5.2, and could be explained by the relationship between jitter and loss. It is important to notice that this dependence is not present in the PFTK model and also shows the limitations of empirical models. This last point illustrates the advantage of a causal study over a purely statistical or empirical one. By simply studying the correlations between p and *tput* and between *rtt* and *tput*, one would miss the dependence between *rtt* and p and obtain biased estimates, as it was the case when studying the intervention on the retransmission score in

the emulated network, Section 5.1.4.

The presence of an isolated subgraph between *dist* and *nbhops* illustrates a remark we made for the emulated experiment. The distance is linked to the propagation delay, which was found not to have any dependence with the *RTT* (see Section 5.1.3). Similarly, the number of hops, approximating the number of routers traversed along the path from the client to the server, is highly dependent on the distance and does not present important variations in our dataset (see Table 5.3). This remark shows an important factor that will impact the accuracy of the independence tests, and consequently the graph obtained by the PC algorithm: the variations in the values of the observed parameters. We can see from Table 5.3 that few losses are observed (p and *retrscore*). This means that the PFTK model, which bases its study on network congestion, does not capture mechanisms that often determine the TCP performance. As defined in Section 5.1.2, p represents the probability of a loss event while *retrscore* represents the probability of a packet loss. Given the difference between p and *retrscore* and the low values we observe for these two parameters, their relative and absolute variations make it difficult to fully capture their dependence.

The influence of the *Time Of the Day* (*tod*) on the *RTT* (*rtt*) captures the peak hour effect, when the network is heavily loaded. On the other hand, the dependence found between the *Time Of the Day* (*tod*) and the *Receiver Window* (*rwin*) is more subtle. It can be explained by the fact that a TCP client may implement the Receiver Window Auto-Tuning (RWAT) function [FSS02] to adapt the size of the receiver window as a function of the Bandwidth Delay Product (BDP) which in turn is influenced by the time of the day. However, intuitively we expect that the *tod* should impact the *RTT* and *Narrow Link Available Capacity* which, in turn, would influence the *Receiver Window*. As the *Narrow Link Available Capacity* is computed using the PPRate tool [ENUK06], its estimation is less accurate when run at the server side where it can underestimate the path capacity. In this case, the effect that the peak hour (*tod*) has on the path capacity, which in turns impacts the receiver window, might not be captured by the estimate of the *nlac* parameter.

The path *nlac*→*bufferingdelay*→*rtt*→*tput* shows, again, that the buffering delay, in routers traversed along the path between the server and the client, is the main factor influencing the *RTT* and plays an important role in the TCP Performance.

Other causal models

As we did for the previous study of the emulated network, Section 5.1.3, we present the causal models that we obtain if using different algorithms or the commonly used Z-Fisher criterion for testing independences in algorithms such as the PC algorithm or the FCI algorithm. We first present the causal model we

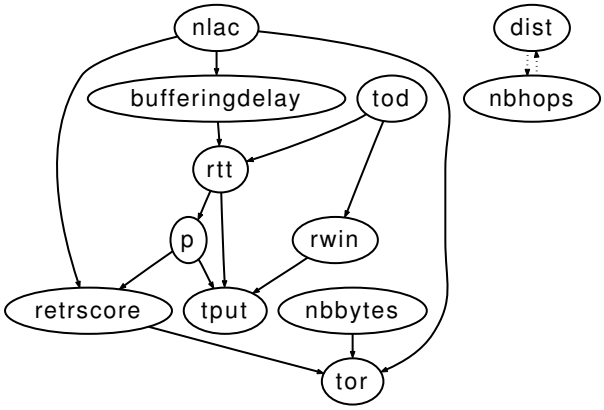


Figure 5.10: Causal graph modeling FTP transfers for real traffic

obtain with the IC* algorithm and the KCI+bootstrap independence test in order to detect the possible presence of latent variables that could prevent us from estimating some of the interventions that we will study in Section 5.2.3. Again, we present the causal model inferred by the kPC algorithm before presenting the different results obtained when we use the Z-Fisher criterion to test parameter independences. For the kPC and the different algorithms implementing the Z-Fisher criterion as independence test, the models presented aim to show that available solutions are not suitable to study telecommunication networks, as it was observed in the study of the emulated network, Section 5.1.3.

Result using the IC* algorithm Figure 5.11 presents the causal model inferred by the IC* algorithm to model the throughput in the study of FTP traffic. We can observe that all the edges are directed, apart from the one between *dist* and *nbhops*, and all the parents of the *throughput* show marked edges going into *tput*, excluding the possible presence of latent variables, apart from the one from p to *tput*. The presence of an unmarked edge means that the algorithm could not find whether there is a latent variable between the two parameters or not. In this case it is our understanding of the TCP mechanisms that help us discarding the possibility of a latent variable between the loss event probability (p) and the throughput (*tput*). The bidirected edge between *dist* and *nbhops* cannot be oriented by the orienting method we use in the PC algorithm. In this case, using the Weakly Additive Noise model (WAN) approach could have helped the orientation of this edge. Taking into account that *nbhops* is discrete, with few different values, and that the distance is a continuous variable, it would require adopting an accurate functional model for the nonlinear regression of *nbhops* on *dist* and of *dist* on *nbhops*.

A last remark about the absence of latent variables in the model presented in Figure 5.11 concerns the time of the day parameter (*tod*). When we first

started this study, we did not include the time of the day in our model. The model inferred by the PC algorithm, with the KCI+bootstrap method, presented unexpected dependencies and orientations, suggesting the presence of a latent variable, which was also indicated by the IC* algorithm. It is our domain knowledge that allowed us to reason about our model and add the time of the day (*tod*) variable in our system and obtain a consistent model.

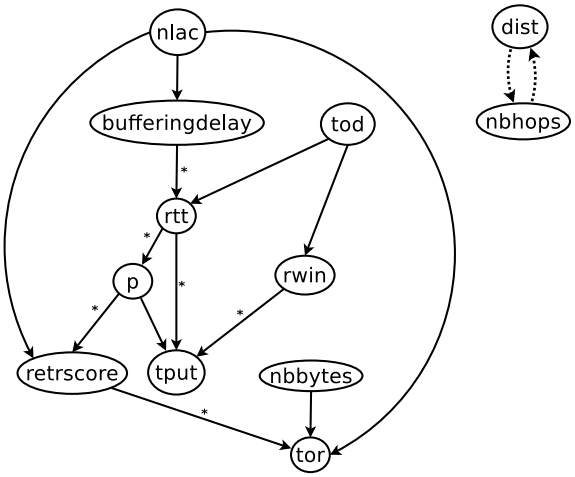


Figure 5.11: Model for the real FTP data using the IC* algorithm

Results using the PC algorithm with the Z-Fisher test To illustrate how important it is to use the test best adapted to the nature of our data, which is nonlinear and not normally distributed, we present in Figure 5.12 the model inferred by the PC algorithm using the Z-Fisher criterion instead of the KCI test. The inferred model has to be compared to the one we obtain when we use the independence test we presented in Section 3.2.2 that is presented in Figure 5.10. Most of the dependencies and orientations present in this model are incorrect, given our domain knowledge, the TCP mechanism and the literature. The graph in Figure 5.12 shows the receiver window (*rwin*), the time of the day (*tod*) and the number of hops (*nbhops*) as parents of distance (*dist*) which is an exogenous variable. Similarly, the RTT (*rtt*) is an empirical cause of TCP throughput, while this model arrives at the opposite result. It seems that the Z-Fisher criterion fails to capture the dependencies between the TCP performance parameters.

As before, in the study of the emulated network, we apply a log linear transform to the parameters of our systems in an attempt to transform the dependencies between the different parameters of our system and convert the original dependencies in new dependencies closer to linearity. Figure 5.13 presents the causal model obtained with the PC algorithm and Z-Fisher criterion when applying a log-linear transformation. Based on Equation (3.1), this transformation

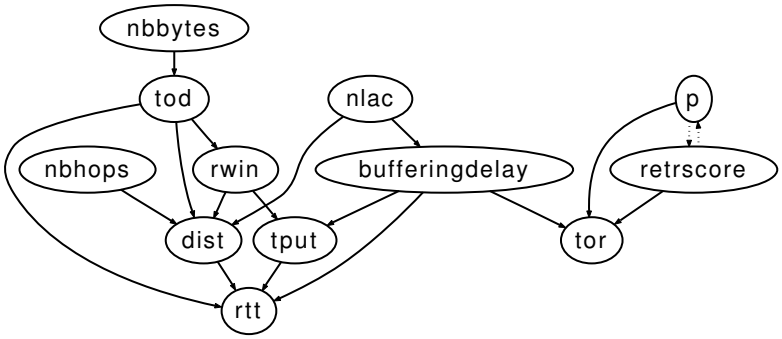


Figure 5.12: Causal model inferred by the PC algorithm, with Z-Fisher criterion, for the real FTP traffic

of the data tries to obtain approximately linear dependencies.

Here again some of the causal dependencies present in the model confirm the inability, even after log-linear transform, of the Z-Fisher criterion to capture the dependencies between the parameters of our model. Some of the direct parents of the throughput (rtt, $rwin$, $retrscore$, p) are the ones we expect to find. On the other hand, some of the exhibited dependencies are somehow counter intuitive (for example $rwin \rightarrow nbhops$). These parameters are the ones used in PFTK model, showing its validity in terms of parameters directly influencing the throughput, and its limitation in modeling their dependences. The model also presents a causal dependence between the time of the day (tod) and the distance ($dist$), and the narrow link available capacity ($nlac$) and the distance ($dist$) that are oriented in a counter intuitive way. Note that the edges are part of a V structure entering a collider and cannot, then, be reoriented without changing the Markov Equivalence Class to which the model belongs.

Finally, we present the model obtained when using the FCI with the Z-Fisher criterion in Figure 5.14. Here the conclusions are similar to the ones concerning the model inferred by the FCI algorithm for the emulated network dataset, as the Z-Fisher criterion is again used to test the independences between the different parameters.

Result using the kPC algorithm Figure 5.15 presents the model inferred by the kPC algorithm when trying to build a causal model of the FTP throughput. Here again, the algorithm gives a very uninformative model. Some of the parameters we know as impacting the throughput are found to be its direct parents but the graph mainly shows them as being all independent. The interest in causal study lies in its ability to detect associations between the parameters that would be naively used as explanatory variable for the study of the target variable (the throughput in our case). In this case, the algorithm does

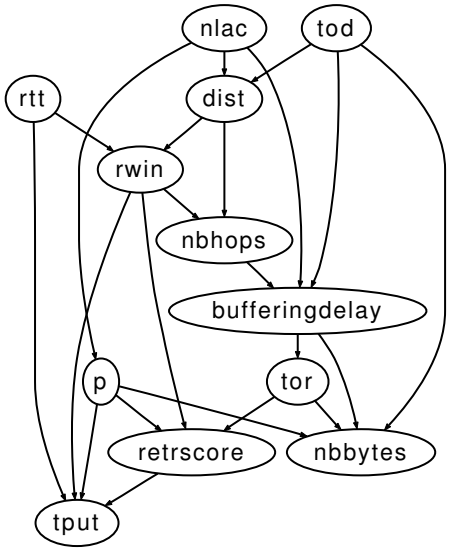


Figure 5.13: Causal model inferred with PC and Z-Fisher when applying log-linear transform to the real FTP scenario

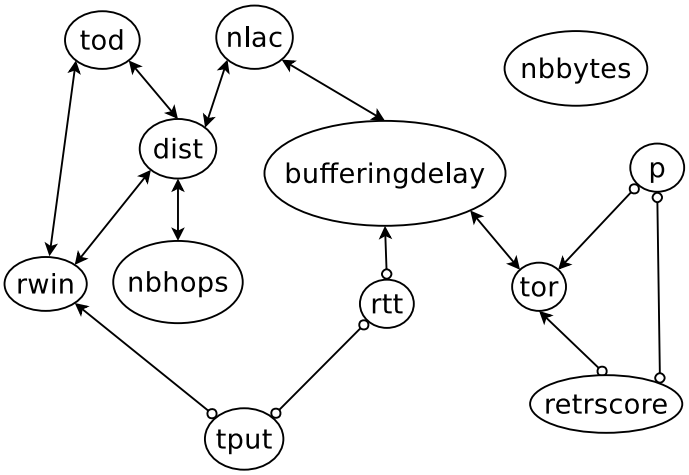


Figure 5.14: Model obtained when using FCI algorithm with Z-Fisher criterion in the real FTP scenario

not rely on Meek rules [Mee95] to orient the edges of the skeleton. The use of non-linear regression under the assumptions of Weakly Additive Noise Models, allows the kPC to infer a DAG and not a PAG. Given these considerations, we cannot re orient the edge between $nlac$ and tod while this would have given a more intuitive model. This example shows the difficulty, when working with real data, to capture non-linear dependencies and their impact on methods that rely on their characterization.

For the rest of our study of the FTP traffic, we use the model inferred by the

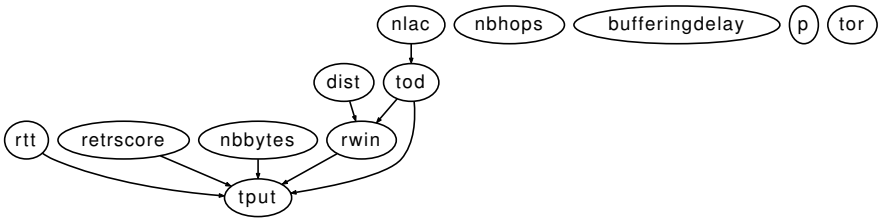


Figure 5.15: Graphical causal model inferred by the kPC algorithm for the real FTP scenario

PC algorithm when implementing the KCI+bootstrap independence test. The models in Section 5.2.2 were only presented for a comparison purpose and a justification of the additional work necessary to implement the KCI+bootstrap procedure.

5.2.3 Prediction

Having inferred the causal graph of our system, we now demonstrate its usefulness to predict interventions on the three most important variables (see PFTK model, Equation (3.1), Section 3.2.1), namely RTT, p and $rwin$. It should be noticed that, in this scenario we cannot directly intervene on the network to validate the results as we could do in for the emulated network (cf Section 5.1.4).

Intervening on the Round Trip Time

From the causal graph presented in Figure 5.10, we want to predict the effect of manually intervening on the RTT. To do so, we use the back-door criterion (cf Section 1.3.4) which ensures the identifiability of this intervention. There is one trek between rtt and $tput$ blocked by tod.[4] We can apply the back-door adjustment, Theorem 3, with the blocking set $Z = \{tod\}$. We use normal kernels to estimate the marginals of the *throughput* ($tput$), the RTT (rtt) and the *Time Of the Day* (tod) and a T-copula to model the conditional probability density function of $f_{tput|rtt,tod}$.

We know from Table 5.3 that the value of the *throughput* before intervention has an average value of 332 kBps, for an average observed RTT of 270 ms. The intervention of setting the RTT value to 100 ms gives an expected value of 480 kBps, with a standard deviation of 176 kBps (compared to 256 kBps prior to intervention). This result is good in the sense that it shows that, by decreasing

[4] "A trek between distinct vertices A and B is an unordered pair of directed paths between A and B that have the same source, and intersect only at the source" [SGS01]

the value of the RTT and its variance, we can improve the *throughput* via intervention. We can precisely quantify the expected improvement in the *throughput* if we modify the RTT.

The improvement of the *throughput* (*tput*) when decreasing the RTT is an expected behavior of TCP performance. However, the naive approach, that computes the effect of setting the RTT to 100 ms by selecting observation where this value of RTT is observed, would produce an incorrect estimate of the post intervention *throughput*. Figure 5.16 compares the results of the two methods. The dashed line with cross markers represents the expected value of the *throughput*, for a given value of RTT, that will be obtained using *the naive approach*. The solid line with diamond markers represents the expected value post intervention computed with the back-door criterion adjustment. We also added the information of the number of samples that were used to compute the different values of the *throughput*.

We see in Figure 5.16 that the expected value of the *throughput* after intervention is smaller than the value we would obtain with the naive approach. This result is supported by our causal model, Figure 5.10, where conditioning on RTT to predict the *throughput* also takes into account the effect of the *receiver window* on the RTT. As mentioned in Section 5.1.4, when studying how an intervention on a parameter, X, would impact Y, all external variables ($\notin \{X, Y\}$) maintain their dependence with Y but not via their (possible) dependence with X, whose behavior is only dictated by the intervention. Therefore, the impact of the *receiver window* on the RTT is spurious and must be neglected, which is done in our approach by the use in the back-door formula and $Z = \{tod\}$. However, this is not the case in the naive approach that includes the spurious impact of the *receiver window* on the RTT and overestimates the effect of an intervention on the RTT. The fact that both curves are quite similar can be explained by the fact that only one variable, the *tod*, is used to block the only back-door passing through *rwin*, which does not exhibit important variations in our dataset. It should be noticed that in the field of telecommunications, one should not expect improvements orders of magnitude higher than the ones we observe here due the complexity of the systems, which does not mean that improvements are not useful.[5]

Intervening on loss event frequency

We now predict the effect of an intervention on the loss parameter p. In this case the blocking set is $Z = \{rtt\}$.

Setting the p parameter to 0.01% allows to achieve an expected *throughput*

[5] Amazon claimed that every 100 ms of latency costs it 1% of profit.

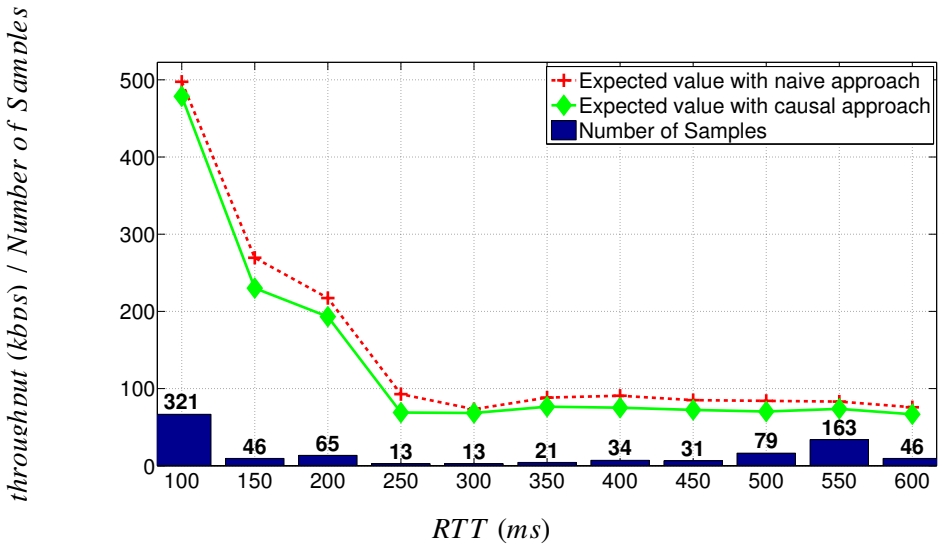


Figure 5.16: Effect of an intervention on the RTT, on the throughput distribution

value of 525.5 kBps as compared to a *throughput* value of 332 kBps for a value of p of 0.07% prior to intervention.

Figure 5.17 compares the naive approach and the causal approach to predict the impact of intervention. Again, we see that the naive approach overestimates the effect of an intervention on p. We obtain a difference between these two methods which is slightly bigger than the one observed in Figure 5.16 when intervening on RTT.

Intervening on the receiver window

One parameter absent from the the PFTK model, Equation (3.1), but present in the model Figure 5.10 is the receiver window. After investigations, we observed that the distribution of the receiver window in our dataset has essentially two peaks around 72 kB and 235 kB, which reduces the range of interventions we can predict, see Figure 5.18. The average value of the receiver window prior to intervention is 137 kB for an average throughput of 332 kBps, see Table 5.3. Using the back door criterion with $Z = \{tod\}$, we could predict the effect of intervening on the receiver window and fix its value to 235 kB. This intervention would result in a 9% improvement of the throughput (365 kBps) for an increase of 70% in the value of the receiver window. Instead, with a decrease of 70% of RTT or p we predicted an improvement of the throughput of 44% and 58% respectively. This observation is consistent with our domain knowledge and Intrabase [SBUK+05] result that shows that the impact of the receiver window on the throughput is less important than the one of loss and delay.

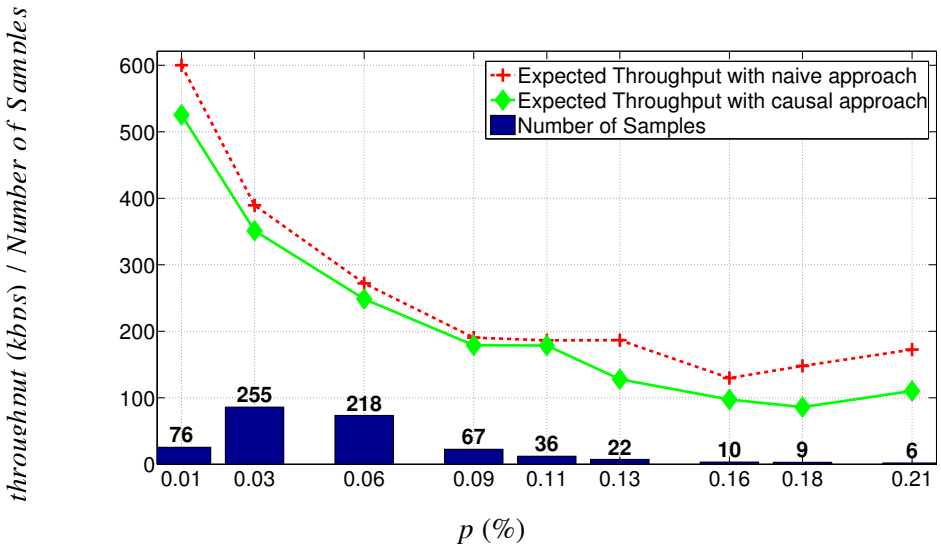


Figure 5.17: Effect of an intervention on p, on the throughput distribution

This does not imply that an intervention on the receiver window should never be made. To decide on an intervention, an operator must balance and compare benefits and costs. Intervening on the loss or delay means upgrading network equipment, which may be more costly than intervening on the receiver window. Here, we do not have the necessary information to assess the cost (and monetary benefits) of interventions, hence our goal is not to give generic recommendations on which intervention should be performed. Rather, we provide a formal method to predict the expected gain in term of performance for each intervention. Then, these predictions can be used to decide on the best intervention in terms of cost versus performance gain (and the corresponding economical gain).

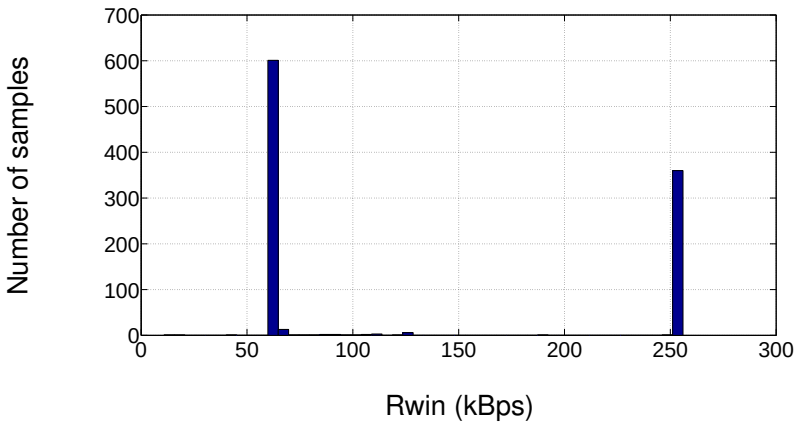


Figure 5.18: Receiver window variations in the FTP dataset

5.2.4 Concluding remarks

In this section, we applied the causal approach, using the methodology described in earlier sections, to the study of TCP performance under real-world FTP connections. Although we cannot fully validate the predictions made in this real-case scenario (since we do not have directly access to the parameters of intervention), the results we obtain are in complete agreement with our understanding of TCP and with the literature. Our study shows that, provided the right tools are used appropriately to cope with the characteristics of the data (in particular the use of the KCI test and of copulae to cope with the non-normality of the distributions and non-linearity of the dependences), the causal framework can be very useful to predict the effect of interventions in the network.

In telecommunication networks, changes in the architecture, routing policies or content placements represent important costs in term of money and time so that testing interventions is often impossible. Due to the complexity of the system, intervention decisions usually rely on domain knowledge and back-of-the-envelope computations. Our study offers an appealing formal and automatic methodology to estimate much more precisely the expected effect of an intervention on the system's performance without having to test the intervention in practice. These predictions can then be translated in economical gains (see [TJA12] and references therein) and used by an operator to decide on a potential intervention by comparing this gain to the intervention cost that only the operator can quantify.

A limitation of the approach proposed in this study is that the range of interventions that can be predicted is dependent on the dataset, i.e., on the range of values that were observed prior to these interventions (as, e.g., in the case of the intervention on the receiver window, Section 5.2.3). This constitutes a limitation as we are often interested in predicting the effect of setting a parameter to a value that is not observed. This limitation is inherent to the fact of inferring the model solely from the dataset. For this reason, when leading a causal study, "outliers" are often of great importance as they represent points of interest for predictions. A possible solution to perform intervention predictions in regions not observed in the data is to extend the model to these regions using additional assumptions not drawn from the dataset, but the accuracy of the results will likely be very sensitive to the assumptions made. Our attempts to model the parameters with mixtures of known distributions such as Normal distributions, Beta distributions, Gamma distributions and so on, showed that such models can approximate the distribution for the values of the parameters we observe but they become totally inaccurate if used to extrapolate the distributions of the parameters of our model to domains where they were never observed.

5.3 Lessons learned from the FTP study

As in the study of the Dropbox application, Section 4.5, this section presents some of the lessons learned from the two studies presented in this chapter.

5.3.1 The importance of artificial dataset designs and their limitations

The first part of this chapter has been dedicated to the causal study of the emulated network. The necessity to study a system where intervention can be performed has been critical to our work. Our interest in the causal study of telecommunication networks comes from the ability to predict interventions. The accuracy of our methods should only be assessed in terms of intervention prediction accuracy. For this reason, the study presented in Section 5.1 is a key step in our work.

We started our work using solutions already implemented, publicly available and used in other domains, in the study presented in Chapter 4. We found that these methods did not work and could not work. We then adapted these methods to take into account the specificity and constraints of the telecommunication networks. The changes that we had to make implied to use more complex concepts for testing independences in the inference of our graphical causal models and for modeling of parameter distributions. Such changes meant to allocate more resource (time and computation).

The only possibility to justify to these changes and validate our approach, and its parameterization, was to compare our results to a ground truth. This could be done with the use of an emulated network.

As in many other research domains, it is always important to dedicate time in validating step by step the different choices that were made in the design of any solution. As the Dropbox study showed, Chapter 4, it is important to take into considerations the domain being studied and the constraints inherent to the system being studied. Dropbox was a complicated application to study compared to FTP and the choice of the FTP application allowed us to create a simple scenario and validate our methods.

Finally, we used to artificial datasets several times in our work. Such artificial datasets range from the most advanced situation, presented in this Chapter, where we emulate a network, to a very simple case where we generate 4 variables according to pre defined mechanisms, as it is the case to study the parameterization of our method in Section 3.3.2, to compare the Z-Fisher criterion and KCI test in Appendix A.1, to study the KCI performance in the presence of categorical data in Appendix A.2 or to study the impact of data shortage on the quality of our predictions in Appendix B. However, artificial datasets are limited.

We can see the limitation of the emulated network in the dependence found between the bandwidth and the delay in the causal model presented in Figure 5.2 and the causal model of the randomized experiment presented in Figure 5.8. The two models show i) the risk of bias when recreating scenarios artificially ii) the limitation, in terms of complexity and variations, that exists when trying to use simple models to recreate complex mechanisms.

5.3.2 Reality is always more complex

Interestingly, complementary conclusions can be made concerning the study presented in Section 5.2.

In the study of a real world system, we are subject to more possible errors. The misbehavior of a component of a network (electronic failures) or errors in our measurements are among the most probable errors that can happen in the study of real systems. The reason why we want to model a system is to gain a control on its behavior that, by default, we do not have. Fortunately, the different values of a parameter do not only capture the behavior of this parameter but also the external parameters (exogenous variables) influencing this parameter (we ignore latent variables here). Taking as an example the receiver window, its values do not only capture the values present in the TCP headers of packets sent by one entity to the other but also the parameters influencing the auto-tuning mechanism, as highlighted in the causal model presented in Figure 5.10.

The fact that a parameter represents a mechanism ruling the system behavior sheds light on two important points related to domain knowledge:

- First, while we do not need to have a very deep domain knowledge of the system (otherwise modeling such system with a graphical causal model would be less appealing) it is necessary to have, in our model, a parameter for each mechanism influencing the performance of the system.
- Second, while Bayesian networks offer a very intuitive way to see causal dependencies between the different parameters of the system we are studying and can exhibit dependencies we did not know about, it is our domain knowledge that will finally give us the full characterization of such dependence. The best examples are the dependencies between delay and receiver window, which required some research to understand, and the dependence between the RTT and loss that invalidates the PFTK model that had suggested the choice of several parameters of our model.

5.3.3 The importance of using the correct methods

The comparison between the predictions of interventions that we obtain with the naive approach and the causal approach, presented in Figure 5.16 and

Figure 5.17, shows that the results we obtain with the two methods present a relatively small difference. While we explain in Section 5.2.4 why such difference is still very important in the domain of telecommunication networks, such remarks should not reduce the important finding that, if the correct methods are used to infer the causal model, and to model the distributions of the different parameters of our model. The accuracy of our predictions could be measured in the case of the emulated network, where the difference with the results obtained with naive approach is more important. The predictions we made showed to be accurate while requiring a limited amount of data.

This remark shows the importance of dedicating time and resource to correctly design our method and take into account the constraint of the system, in our case the non linearity and non normality. Parametric modeling of order three distributions, in zones where we only have a few tens of samples, using, for example the Rebmix package [NF11], did not succeed. Other methods, such as histograms, or any quantization methods, could not work in cases where the number of dimensions of the distribution we need to estimate is important (like the ones we obtain from the equations inferred with the back-door criterion) and the number of samples is small.

Chapter 6

Study of the DNS service

In the previous chapter we validated our methods in an emulated environment and then presented the causal study of the FTP traffic in real case scenario. In this chapter, we extend our use of causal models and of the causal theory to study counterfactuals that allow us to better understand the impact of the different mechanisms behind the DNS service in the performance of the Content Distribution Network users.

6.1 Problem statement

Each time an Internet user wants to access a resource, he uses an human readable name called Uniform Resource Locator (URL), containing the domain name of the administrative entity hosting this resource. However, a domain name is not routable and needs to be translated into the (IP) address of a server hosting the resource the client wants to access. It is the role of the DNS service to translate the domain name to the corresponding IP address. Many popular services such as Youtube, Itunes, Facebook, Twitter, rely on CDNs where objects are replicated on different servers and in different geographical locations to optimize the performance for their users. When a client wants to access one of these services, its default DNS server contacts the authoritative DNS server for the domain hosting the resource the client is requesting. Based on the location the request is coming from, the authoritative DNS redirects the client to the optimal server. Most of the ISPs provide a DNS service, but it is now common to see customers using the service of a DNS instead [OSRB12] such as Google DNS or openDNS. Clients using the DNS service of their ISP are served by a local DNS server. Local DNS servers are closer to clients, their usage provides a more accurate location information to the CDN compared to the locations represented by the few DNS servers offered by a public DNS service, Figure 6.1. There have been several studies suggesting that public

DNS services do not perform as well as local DNS services provided by ISPs, mainly because of the impossibility of public DNS to correctly communicate the location of the clients originating the request [HMLG11, AMSU10], Figure 6.2.

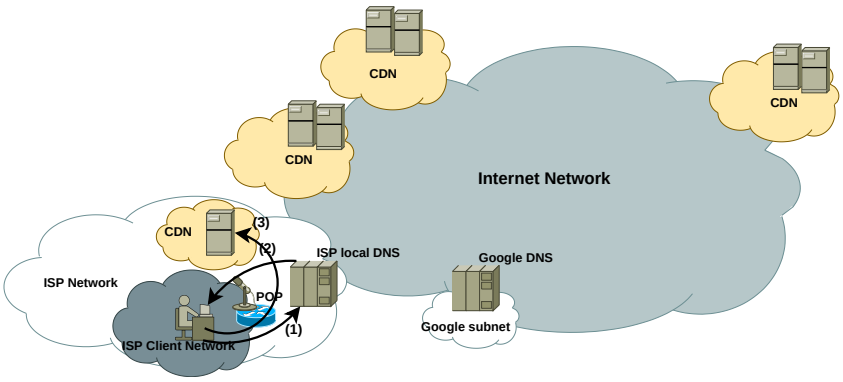


Figure 6.1: Summary of the communication of a client of an ISP DNS service accessing a CDN resource. (1) The client requests its local DNS server the IP address of a machine hosting the resource it wants to access. (2) The ISP DNS contacts one authoritative CDN DNS that provides it the IP address of a close machine hosting the requested resource and the ISP DNS server communicates this address to the client. (3) The client requests the resource to the machine which IP address was giving to him by its local DNS server

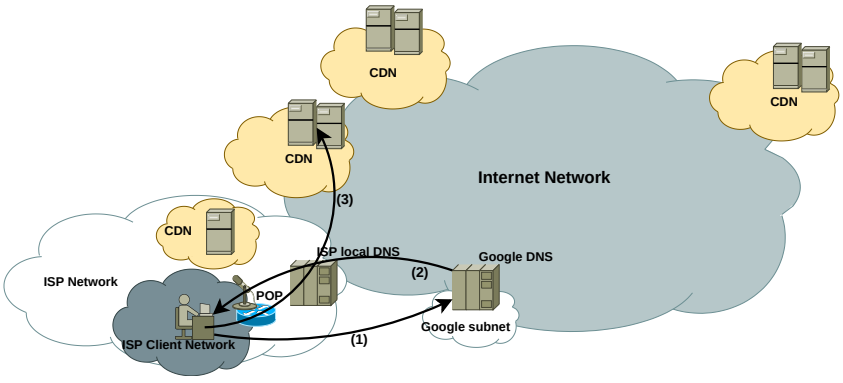


Figure 6.2: Summary of the communication of a client of a Google DNS service accessing a CDN resource. (1) The client requests the closest Google DNS server the IP address of a machine hosting the resource it wants to access. (2) The Google DNS contacts one authoritative CDN DNS that provides it the IP address of a close machine hosting the requested resource and the Google DNS server communicates this address to the client. (3) The client requests the resource to the machine which IP address was giving to him by the Google DNS server

However, studying the performance of the users accessing Internet resources is a complex task. Many parameters influence the end user experience and the relationships between these parameters is not always observable or intuitive.

It is therefore necessary to adopt a simple, yet formal, model that allows us to understand the role of a given parameter and its dependencies with other parameters. In this chapter, we infer the causal model of the impact of the choice of a DNS service on the throughput performance experienced by client accessing resources via the Akamai CDN. We further use this model to predict the effect on throughput performance had the client used another DNS service. This prediction allows us to understand the impact of choosing one DNS service instead of another. From the same model, we are also able to indicate how to improve the performance of the users of the local DNS.

This study differs from previous studies of DNS services in several points:

- We use a causal approach that formally models the structural dependencies of the different parameters influencing the user experience (throughput).
- After observing that the users of the local DNS experience better performance than the Google DNS users, we show that such improvement is made through a redirection of the local DNS users to closer servers and we are able to quantify such performance improvement.
- The causal model of our system shows that the TCP parameterization of the servers accessed by the users of the Google DNS plays a key role in improving their performance. Besides fully explaining the observed performance, this result also indicates a solution to further improve the performance of the users of the local DNS service.

Overall, the main contribution of this study resides in the methodology that is presented, and in its use of counterfactuals to understand the causal dependencies in complex systems.

Section 6.3 presents the environment of our study and the description of the parameters constituting our system. Section 6.4 presents our study of the DNS impact on the throughput. In particular we present the causal model of our system where we can observe the impact of the DNS choice on the throughput. Our approach also allows us to predict the improvement that could be achieved by modifying the parameterization of the servers accessed by the users of the local DNS service. Section 6.5 summarizes the method presented and proposes some directions for exploiting further this method.

6.2 Counterfactuals

The inference of a causal model is made based on statistical tests performed on the observed data, however a causal model encodes structural (or functional)

relationships between the parameters of the system being modeled. Having access to the structural dependencies between the parameters of a system allows us to predict the reaction of a system after an intervention for which we modify the behavior of the components being observed. One vision of causal dependencies comes the notion of ***counterfactuals***, Section 1.3.2.

An example of counterfactual, taken from [Pea09], is the following. In a controlled clinical trial (cf Section 1.3.2) when we find that a treatment, X, has no effect on the distribution of the subject's response, Y, which may stand for either recovery ($Y = 0$) or death ($Y = 1$). Then, if a given subject S has taken the treatment and died, we want to know whether S dies *because* of the treatment or *regardless* of the treatment. Said differently, would S would have died had he not been treated ?

Such consideration, or questions, can be answered if we know the structural dependencies between the different parameters that form the causal model of Y and X.

In this study, we exploit this property of causal models to answer counterfactuals and measure the causal effect of the choice of a DNS service on the performance of the users of the Akamai CDN.

6.3 Experimental set up

6.3.1 Experiment design

To observe the local DNS queries and answers for the clients using their ISP DNS service, we place ourselves at a Point of Presence (PoP) of a large European ISP and observe the traffic directed to or coming from the Akamai CDN, see Figure 6.1 and Figure 6.2. To model the impact of the DNS choice on the ISP client throughput, we make three choices: i) We only focus on the traffic carried by the TCP protocol. ii) To better capture the DNS and network impact on performance, we restrict ourselves to voluminous connections that go beyond the TCP slow start phase and carry at least 2MB. iii) As more than 90% of the observed connections use either Google DNS (*Google DNS (GDNS)*) or the DNS of the local ISP (*Local DNS (LDNS)*) we consider only these two DNS services.

The probe is placed between the client and the server. We can then divide our vision of the network in two. We call internal network, denoted ***isp network***, the part of the network between the client and the probe. On the opposite, we call external network the part of the network between the probe and the server, assimilated to the Internet network and denoted ***inet network***. The traffic is observed on two different days, a Thursday and a Sunday, from 5.30 pm to 9.30 pm (peak time).

The next section presents the parameters that are recorded and the parameter values (minimum, maximum, mean and standard deviation).

6.3.2 Model parameters

We use the Tstat software [FMM+11] to passively measure per-flow statistics. We select a subset of statistics representing the parameters of our system that are known, from domain knowledge, to impact the throughput. We obtain a dataset where each sample represents a single connection and for each connection several parameters are observed. We have about 7000 connections, which represent our sample size.

We consider several parameters in addition to the Tstat statistics [FMM+11]: the DNS service, the number of hops between the client and the server and the server IP address, represented by its corresponding Autonomous System (AS) number.

6.3.3 Observation summary

For each connection, we record 19 parameters. In Table 6.1, we present the average (μ), minimum (min), maximum (max), standard deviation (σ) and coefficient of variation ($CoV = \frac{\sigma}{\mu}$) of each of the 19 parameters over the 7000 connections. As this work focuses on the comparison between the performance of LDNS users and GDNS users, Table 6.2 presents the statistics for the connections where the LDNS is being used and for the connections where the GDNS is used separately.

We use the following notations:

- Parameters with the prefix isp represent $isp\ network$ statistics while the ones with the prefix $inet$ represent the $inet\ network$ statistics.
- The suffix avg represents the average value of a given parameter over a connection
- The suffix std represents the standard deviation of a given parameter over a connection
- The rto parameter represents the presence of at least one packet retransmission due to a time out and $retrscore$ the fraction of retransmitted packets.
- The parameters $rwin*$ and $cwin*$ represent receiver window and congestion window metrics respectively.

- The day of the week and time of the day are captured by the variables *dow* and *tod* respectively.

It should be noticed that the congestion window is a sender parameter. Tstat estimates the congestion window by looking at the amount of data that has not been acknowledged yet, namely the in-flight size. As the sender-to-the-probe path is typically very high speed (∼10Gbps), and the bottleneck is after the probe (e.g., the ADSL link), the in-flight-size is a very good approximation of the sender congestion window.

Destination IP (*dstip*), DNS (*dns*) and days (*dow*) are categorical data for which the average value, standard deviation or coefficient of variation do not exist.

Several observations can be made when looking at the variation of the different parameters in Table 6.1 and will be referred to in the discussion of the causal model of our system.

First, we can observe that the value of the average RTT inside the ISP network (*isprttavg*) is almost three times as high as the average RTT in the "Internet" network (*inetrttavg*). This difference is due to the use of an ADSL link as access link and the latency experienced by the packets entering or exiting the ISP network. When many packets are being sent at the same time they can create some congestion that implies high values in the RTT. Such changes and variations can also be observed in the standard deviation of the RTT that is more than ten times bigger in the internal network than in the external network.

Second, another information that does not appear in Table 6.1, concerns the loss rate . 64% of the connections experience at least one loss but only 27% of them show a retransmission score bigger than 0.5%. This value suggests that, in cases of congestion, packets queue in buffers but are not necessarily dropped.

Concerning the dataset presented in Table 6.2, several observations can be made.

First, we can observe that users of the local DNS experience, on average, a better throughput (*tput* = 3.2 Mbps) than the users of the Google DNS (*tput* = 3 Mbps). For both classes we can observe a Maximum throughput around 30 Mbps that suggests that Fiber access might be used by a small number of ISP clients.

Second, as expected, the external delay experienced by the users of the Google DNS service (*inetrttavg* = 48 ms) is more than twice the one experienced by the users of the local DNS service (*inetrttavg* = 20 ms). However, surprisingly, the opposite can be observed for the internal delay average value (*isprttavg*) and standard deviation (*isprttstd*). Apart from the fact that far more connections are observed for the users of the local DNS service as compared to the users using the Google DNS service, we do not have formal explanation and further research, with other range of observations, should be used.

Table 6.1: Summary of the different parameters

Parameter	μ	min	max	σ	CoV
dstip	N.A.	1300	34000	N.A.	N.A
dns	N.A	1	3	N.A.	N.A.
dow	N.A.	4	7	N.A.	N.A.
tod (s)	7100	52000	78000	4400	0.1
isprttavg (ms)	76	0	19000	460	6.1
isprttstd (ms)	100	0	37000	960	9.2
ispnbhops	1.8	1	3	0.51	0.3
inetrttavg(ms)	26	0.48	660	27	1.0
inetrttstd (ms)	8.2	0	4700	61	7.5
inetnbhops	9.4	2	21	2.8	0.3
rwin0	0.83	0	360	11	13
rwinmin (kB)	31.3	0.004	65	22.5	0.9
rwinmax (kB)	213	17.5	2625	150	0.7
cwinmax (kB)	150	7.3	1625	103	0.7
cwinmin (kB)	0.9	0.001	1.5	0.6	0.7
retrscore	0.005	0	0.19	0.009	1.9
rto (bool)	0.11	0	1	0.32	2.8
nbbytes (MB)	23.8	2.1	3875	138	5.7
tput (Mbps)	3.2	0.006	35	2.6	0.8

Table 6.2: Summary of the different metrics for the two DNS: Local DNS (LD) and Google DNS (GD) (*dow* and *tod* are similar and provide no insights, so they were removed)

Par	μ		min		max		σ	
	LD	GD	LD	GD	LD	GD	LD	GD
isprttavg (ms)	80	61	0	0	19000	15000	470	440
isprttstd (ms)	1100	76	0	0	32000	37000	920	1100
ispnbhops	1.8	1.9	1	1	3	3	0.53	0.4
inetrttavg (ms)	20	48	0.48	11	510	660	20	38
inetrttstd (ms)	8.6	6.5	0	0	4700	1400	65	44
inetnbhops	8.7	12	2	5	17	21	2.4	2.7
rwin0	0.97	0.29	0	0	330	360	12	9.2
rwinmin (kB)	35	12	0.004	0.03	65	65	28	14
rwinmax (kB)	213	213	18	18	2625	2000	150	138
cwinmax (kB)	163	118	7.3	7.8	1625	738	108	72
cwinmin (kB)	0.9	1.2	0.001	0.001	1.5	1.5	0.6	0.5
retrscore	0.005	0.004	0	0	0.19	0.06	0.01	0.01
rto (bool)	0.11	0.11	0	0	1	1	0.32	0.31
nbbytes (MB)	29	7	2.1	2.1	3875	1375	150	44
tput (Mbps)	3.2	3	0.006	0.007	35	29	2.7	2

6.4 Causal study of DNS impact on the throughput

6.4.1 Modeling causal relationships

Using the PC algorithm [SG91] and the kernel based independence test from [GFT+08] we obtain the Bayesian network representing the causal model of our system, presented in Figure 6.3. The response variable (*tput*), the DNS variable (*dns*)

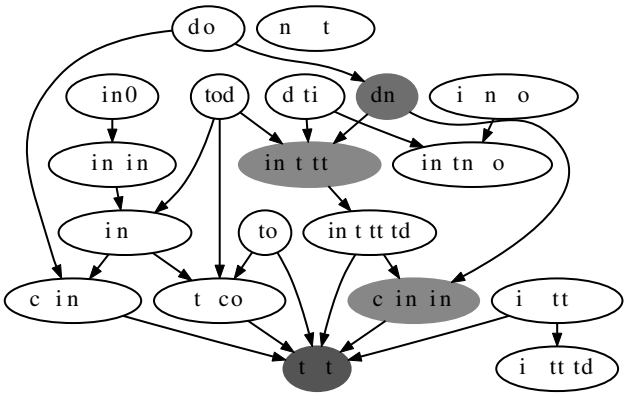


Figure 6.3: Bayesian network representing the causal model of Web performance using two different DNS: the public Google DNS and the DNS of the local ISP

and its two children in the graph (*cwinmin* and *inetrttavg*) appear in a darker color as they represent the variables on which our study will focus. In this section we briefly discuss some of the dependencies exhibited by this model.

We can observe that the day of the week (*dow*) and the time of the day (*tod*) are two parentless nodes, which is not surprising, and that the time of the day (*tod*) influences the RTT between the probe and the server (*inetrttavg*), which captures the *peak hour effect*.

The observation concerning the the two rtt parameters in Section 6.3.3 suggested that the internal rtt would have the stronger impact on the throughput. However, we can observe a direct dependence between the standard deviation of the external RTT (*inetrttstd*) and the throughput (*tput*) but not between the standard deviation of the internal RTT (*isprttstd*) and the throughput. This observation illustrates the ability of causal model to exhibit non intuitive dependencies.

On the other hand, we observe that the day of the week (*dow*) influences the DNS service used by the clients (*dns*). As our observations are made on two days only (a Thursday and a Sunday), our conclusions are limited. However, looking at the data, it appears that on Thursday 72% of the connections use the LDNS service against 28% using the GDNS service, while on Sunday 93% of the connections use the LDNS service against 7% using the GDNS service. It would be interesting to identify the clients using one DNS service and compare their locations with the ones of the clients using the the other DNS service to better understand this dependence. The day of the week may capture the difference in the Internet usage and the devices used at home and at work. However, for privacy reasons, the IP addresses of the clients are obfuscated, which prevents us from investigating this hypothesis.

One of the most interesting dependencies, which motivated this work, is the

one present between the DNS (dns) and the external RTT ($inetrttavg$). We could observe in Tstat logs that the servers accessed by the clients using the LDNS service are often located inside the autonomous system of the ISP. This is not the case for the clients using the GDNS service. As mentioned in the introduction and in [HMLG11], clients using the local DNS service benefit from a redirection to servers closer than the ones of the clients using a public DNS service. We observe an average external RTT of 20 ms for the LDNS service users, while the users of the GDNS service experience an average external RTT of 48 ms (see Table 6.2).

We can also see that congestion window metrics ($cwinmin$, $cwinmax$) have a direct impact on the throughput ($tput$). Additionally, the minimum congestion window ($cwinmin$) has the DNS (dns) as direct parent. Its average value for clients using Google DNS is 1.2kB against 0.9kB for users served by the local DNS, see Table 6.2.

As explained previously, a parameter present in a causal model represents also the mechanisms captured by such parameter. This is the case of the $cwinmin$ that also captures the tuning of the server TCP parameters, such as the initial congestion window with which the minimal congestion window is related.

Clients using the local DNS often access their objects from servers that are located *inside* the ISP network. These servers could have a configuration different from the servers accessed by the users of the GDNS. This hypothesis could also explain the fact that both DNS services result in a similar throughput despite a different RTT. Two other reasons could be the impact of losses on the congestion window or the load of the servers being accessed by the clients.

To capture the server load, we estimate the server processing time defined from the time at which a server sends the acknowledgment of the client HTTP/GET message and the time at which it sends the first data packet. However, the server processing time shows an expected value of 43 ms for the LDNS users against 64 ms for the GDNS users. A higher processing time for the servers accessed by the GDNS users suggests that they are more loaded. On the other hand, the congestion window is impacted by the loss but, as mentioned in Section 6.3.3, few losses actually happen during the period of observation of the system and no significant dependence is found between the loss ($retrscore$) and the DNS service (dns).

It comes with no surprise that the internal RTT ($isprttavg$) is a parent of the throughput. The absence of a dependence between the time of the day (tod) and the internal RTT can be explained by the fact that all the observed users are using the same “internal” path (the path from the users to the probe).

We see that the maximum receiver window advertised by the client ($rwinmax$) has the time of the day as one of its parent (tod). Such fact may be due to the TCP buffer auto tuning mechanism [FSS02] that adjusts the receiver window

according to the quantity and frequency of data received by the client which is influenced by the time of the day.

We can notice the absence of an edge between the DNS (*dns*) and the destination IP address (*dstip*) and the absence of any adjacency with the object size (*nbbbytes*). An explanation of the absence of these edges could be the unequal distribution of the servers accessed by the users of the LDNS service when compared to the distribution of the servers accessed by the users of the GDNS service. A solution to detect weaker dependences is to increase the acceptance rate in the independence tests. Increasing the acceptance rate implies a higher risk in failing to reject weak independences and should be used with caution. The independence of the object size from other parameters influencing the throughput is not necessarily surprising as we consider long connections. However, the absence of a dependence between the DNS (*dns*) and the destination IP (*dstip*) is more difficult to explain as it could also come for a lack of granularity in the representation of IP address by its AS number. This last point does not have an important impact on the results presented in this study.

The loss parameters (*retrscore* and *rto*) and RTT parameters (*inetrttstd* and *isprttavg*) are four of the six direct parents of the throughput, which is in line with our domain knowledge of TCP. The additional parents are the congestion window metrics of the server (*cwinmin* and *cwinmax*).

The fact that none of the receiver window metrics (*rwin**) is a direct parents of the throughput (*tput*) is not surprising. Intrabse [SBUK+05] diagnoses that the connections are never limited by the receiver. We also present in Figure 6.4, the comparison, for each connection, of the throughput obtained for a client with the minimum and average quantity of information that the same user is willing to accept (estimate as $\frac{rwin*}{rtt}$). Tstat does not provide the average receiver window value (over a connection). We use the average between the minimum and maximum receiver window value advertised by a client during a connection as its approximation. We can see that the clients are not limiting the throughput.

6.4.2 Estimation of the impact of a given parameter in the difference of performance experienced by two different users

We have seen that the Bayesian network derived in the previous section reveals a rich set of causal relationships that indicate how the different parameters impact the throughput. We will now use this model to answer *what-if questions* from the existing data without the need to collect more data or perform additional experiments.

In this section, as dealing with probabilities, we compare the expected values of the throughput ($\mathbb{E}[TPUT] = \int f_{TPUT}(tput)\mathrm{d}tput$) instead of its average values (μ_{TPUT}).

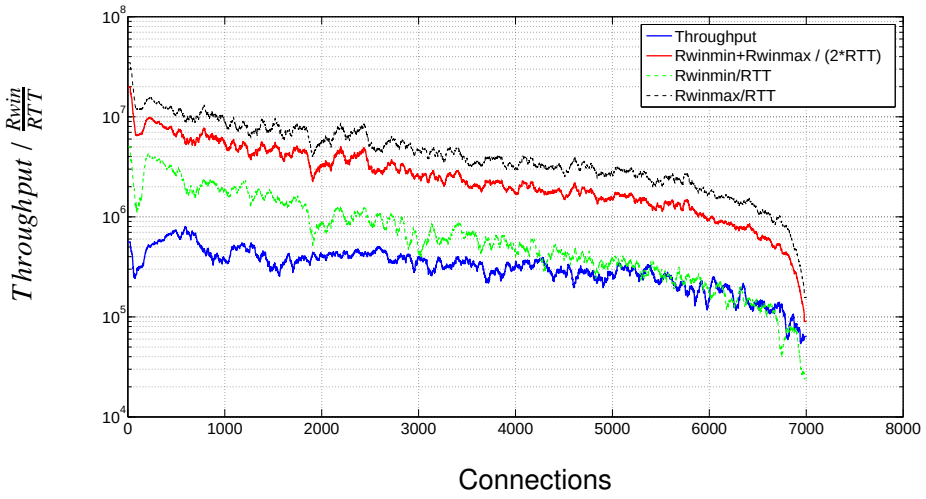


Figure 6.4: Comparison of the throughput experienced by the different users and the limitation imposed by the users memory resources

Distance and delay

In this section we use our causal model to answer the question: "*What is the gain, in terms of performance, induced by choosing the local DNS service and being redirected to servers closer to the client?*".

This question can be reformulated as: "*What would have been the performance of a user served by the local DNS if it would have been redirected to a server whose inetrtt corresponds to the one the Google DNS service would have redirected him to ?*".

In our case, this question is equivalent to predict the effect of an intervention where we modify the external delay (RTT) experienced by clients served by the LDNS and give this delay the distribution of the delay experienced by clients served by the GDNS; the distribution of the rest of the parameters being kept identical for the LDNS service users.

More formally, if we denote by RTT the *inetrttavg* parameter, by LD the local DNS and by GD the Google DNS, we want to estimate the following distribution:

$$f\left(TPUT = tput | DNS = LD, do(RTT \sim f_{RTT|do(DNS)}(\cdot, GD))\right). \tag{6.1}$$

We can observe from the causal graph Figure 6.3 (cf the explanation of d-separation in Section 1.3.4) that $(RTT \perp\!\!\!\perp DNS)_{G_{\underline{DNS}}}$ which implies (Rule 2 from Theorem 2):

$$f_{RTT|do(DNS)}(rtt, GD) = f_{RTT|DNS}(rtt, GD). \tag{6.2}$$

From [Pea09, Section 4.2], if we want to predict how an intervention on X affects Y, where the intervention on X is enforced with the probability $f^*(X|Z)$, we obtain:

$$f(y)_{|f^*(x|z)} = \int_{D_X} \int_{D_Z} f_{Y|do(X),Z}(y, x, z) f^*(x|z) f(z) \mathrm{d}x \mathrm{d}z. \tag{6.3}$$

On the other hand, we can read from the causal graph in Figure 6.3 (cf the explanation of d-separation in Section 1.3.4) that $(RTT \perp\!\!\!\perp TPUT|DNS, TOD)_{G_{\underline{RTT}}}$. It follows, from Rule 2 of Theorem 2 that

$$\begin{aligned} &f_{TPUT|do(RTT),TOD,DNS}(tput, rtt, tod, dns) = \\ &\quad f_{TPUT|RTT,TOD,DNS}(tput, rtt, tod, dns). \end{aligned} \tag{6.4}$$

As a consequence we can rewrite Equation (6.1) as:

$$\begin{aligned} &f(tput|LD)_{f(rtt|do(GD))} = \\ &\int_{D_{RTT}} \int_{D_{TOD}} f(tput|do(rtt), LD, tod) f(tod) f(rtt|do(GD)) P(GD) = \\ &\int_{D_{RTT}} \int_{D_{TOD}} f(tput|rtt, LD, tod) f(tod) f(rtt|GD) P(GD) \end{aligned} \tag{6.5}$$

using Equation (6.2) and Equation (6.4).

Limitations: Figure 6.5 represents the distribution of the external RTT for GDNS users and LDNS users. The prediction represented by the Equation (6.1) is possible as, roughly, the range of the external RTT values observed for GDNS users represents a subset of the values observed for the LDNS users. For the opposite intervention, where GDNS service users would be given access to servers placed at the locations the LDNS service users are redirected to, the prediction presents an important challenge because we do not know $f(tput|rtt, GD, tod)$ for some of the RTT values for which $f(RTT|LDNS) > 0$. This point is a limitation that is common to many machine learning problems, where the amount of available information limits the range of predictions we can make. We are currently working on the extension of the method to the case where less information is available. This extension would require a parametric model to extrapolate the different pdfs out of the domains where the variables of our system are observed. For the purpose of this paper we do not need such extrapolations and manage to obtain the predictions we want directly from our observations of the system.

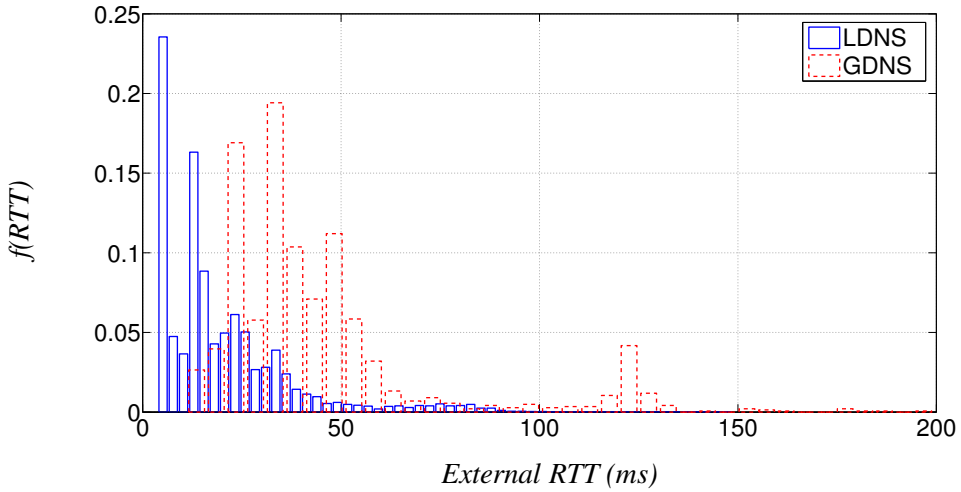


Figure 6.5: Histogram of the external RTT for the local DNS (LDNS) and Google DNS (GDNS)

Results: The result of the intervention is presented in Figure 6.6 with cumulative distribution functions (CDFs). The CDF of the throughput for the LDNS before intervention is plotted (blue solid line) with the CDF of the throughput for the LDNS service users after an intervention setting their external delays distribution to the delay distribution seen by the GDNS users (red dotted line). We can observe the distribution being clearly shifted towards lower values of the throughput.

The expected throughput for clients using the local DNS service prior to intervention is **3.5 Mbps** and **3.0 Mbps** after intervention (14% decrease). This result shows the gain in performance that the redirection to closer CDN servers, provided by the use of the local DNS service, represents. We can assign the 14% decrease in the throughput to the higher values of RTT given to LDNS users in the intervention. These higher values correspond to the RTT of the GDNS users and reflect the suboptimal redirections of user demands to server hosting the requested resources.

Interpretations: The results we obtain cannot be verified in practice as this manipulation would require moving the servers hosting the resources that the local DNS users want to access to locations the Google DNS service redirects his users to and force the local DNS to redirect its users the way Google DNS would. In fact, this is precisely the benefit of the causal approach we adopted: our model offers the possibility to predict the effect of interventions that are difficult to perform experimentally. From the prediction of such interventions we can derive and quantify the impact of the DNS service choice on user performance (throughput).

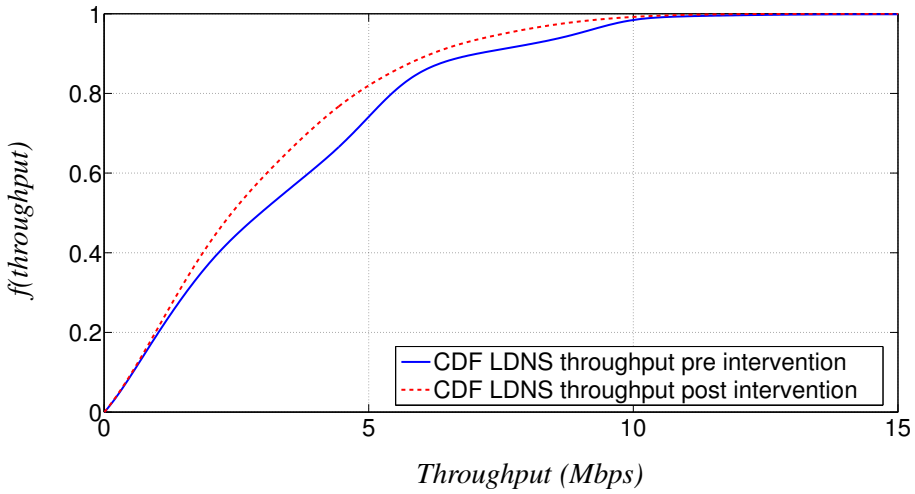


Figure 6.6: Evolution of the throughput distribution before and after intervening on the external delay experienced by Local DNS (LDNS) clients

The results show that, the local DNS obtains a gain in performance by decreasing the RTT delay experienced by its users. This difference in the external delay is due to the non-optimal redirection to CDN servers by the Google DNS service. The gain was quantified by comparing the original performance to the one the users of the local DNS would have experienced if a different strategy was used.

The previous results did not consider the impact of the servers themselves, captured by the minimum congestion window in our problem, or other parameters such as the loss (*retrscore*) that are different between the two DNS services and could explain the throughput experienced, in the original dataset, by the users of the GDNS service that is only 7% smaller. As we can see from the causal model, Figure 6.3, the loss parameters (*retrscore* and *rto*), internal delay parameters (*isprttavg* and *isprttstd*) or maximum congestion window (*cwinmax*) are not influenced by the choice of the DNS service (*dns*). Therefore, the next section focuses on the impact of the minimum congestion window on the performance. Note that *cwinmin* is a direct parent of the throughput (*tput*) and is influenced by the DNS service choice (*dns*).

Minimum congestion window

As mentioned previously, the minimum congestion window (*cwinmin*) is a direct parent of the throughput (*tput*), see Figure 6.3. Its average value is higher for the clients using the GDNS service than for the clients using the LDNS service (1.2kB and 0.9kB respectively). The difference in the expected value of the throughput of LDNS users (3.5 Mbps) and GDNS users (3.3 Mbps) is 6%,

smaller than the gain for the LDNS users being redirected to closer server, that is estimated to be 14%. We make the hypothesis that the minimum congestion window represents a difference in the configuration of the servers accessed by the LDNS users and the configuration of the servers accessed by the GDNS users. To study this hypothesis we estimate the causal effect of the minimum congestion window on the throughput, mediated by the choice of the DNS service.

In this section we are interested in estimating the influence of the DNS service on the throughput via its impact on the minimum congestion window. It is equivalent to ask the question: "*What would be the throughput for the clients using the local DNS if the servers they are redirected to would present the same minimum congestion window as the ones Google DNS users are redirected to ?*".

Because the method is the same for the minimum congestion window (*cwinmin*) as it was for the external delay (*inetrttavg*), we only write down the final equation corresponding to our problem.

We observe, from the causal graph of Figure 6.3 (cf the explanation of d-separation in Section 1.3.4):

- $(CWINMIN \perp\!\!\!\perp DNS)_{G_{\underline{DNS}}}$
- $(CWINMIN \perp\!\!\!\perp TPUT|DNS, INETRTTSTD)_{G_{\underline{CWINMIN}}}$

For the sake of brevity, we denote *cmin* the minimum congestion window (*cwinmin* in our model) and σ_{rtt} the standard deviation of the external rtt (*inetrttstd* in our model). We still use LD when referring to the local DNS and GD for the Google DNS. Therefore, we obtain the following equation:

$$
\begin{aligned}
&f(tput \mid LD)_{f(cmin|do(GD))} = \\
&\int_{D_{CMIN}} \int_{D_{\sigma_{RTT}}} f(tput|do(cmin), LD, ts) f(\sigma_{rtt}) f(cmin|do(GD)) P(GD) = \\
&\int_{D_{CMIN}} \int_{D_{\sigma_{RTT}}} f(tput|cmin, LD, \sigma_{rtt}) f(\sigma_{rtt}) f(cmin|GD) P(GD) \qquad (6.6)
\end{aligned}
$$

Results: From Equation (6.6) we can predict the distribution of the throughput for the LDNS users after an intervention where we give the minimum congestion window the distribution seen by GDNS users. The CDFs of the pre-intervention throughput (solid line) and post-intervention throughput (dotted line) are presented in Figure 6.7 where we can see the gain in throughput corresponding to the intervention on the LDNS server minimum congestion windows. The expected throughput for LDNS service users after the intervention is **4.6 Mbps** (against **3.5 Mbps** prior to intervention) and represents a throughput gain superior to 32%. The gain observed for this intervention is due to the fact that

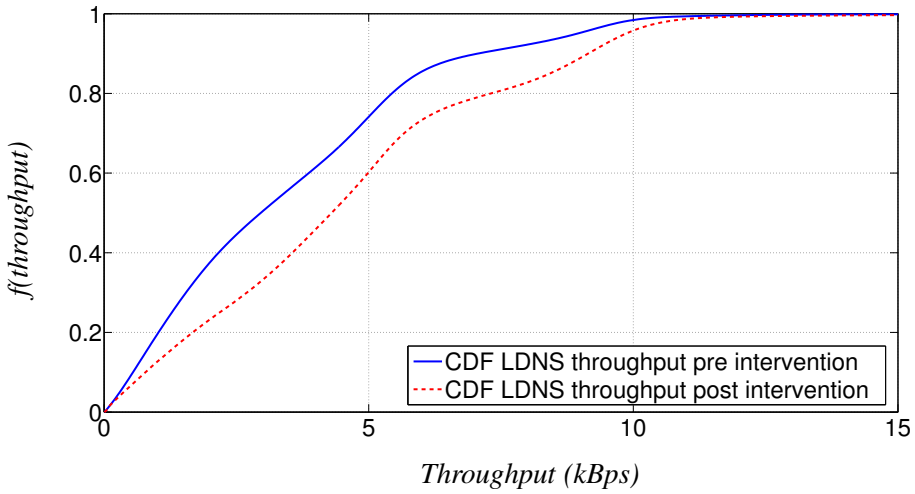


Figure 6.7: Evolution of the throughput distribution before and after intervening on the minimum congestion window of servers the local DNS redirects its clients to

the servers GDNS service users are redirected to use higher values for their minimum congestion window.

Remarks: The study of the opposite intervention, where GDNS service users are redirected to servers with a minimum congestion window following the distribution of the minimum congestion window seen by the LDNS service users, in the original dataset, is more complex. The reasons are the same as the ones mentioned in Section 6.4.2. If we observe the distribution of the minimum congestion windows for LDNS service users and GDNS service users, Figure 6.8, we can notice the absence of $cmin$ values for GDNS users to estimate $f(tput|cmin, GD, \sigma_{rtt})$ for values of $cmin$ where $f(cmin|LD) > 0$.

The results obtained for the intervention on the external RTT were supported by the observations of the system that showed that the local DNS redirects its users to closer servers and, by doing so, improves their throughput (relatively to the situation where the local DNS would not do so). These section findings verify a hypothesis that was made when looking both, at the data and the causal graph. It appears that, while the proximity of the server has an important impact on the throughput, the configuration of the server hosting the content a client wants to access also has an important impact. The impact of a server configuration on its client throughput can be noticed from two different observations. First, from the data, the GDNS service users obtain an expected throughput value (3.3 Mbps) similar to the throughput of the LDNS service users (3.5 Mbps) while experiencing a bigger external delay (48 ms

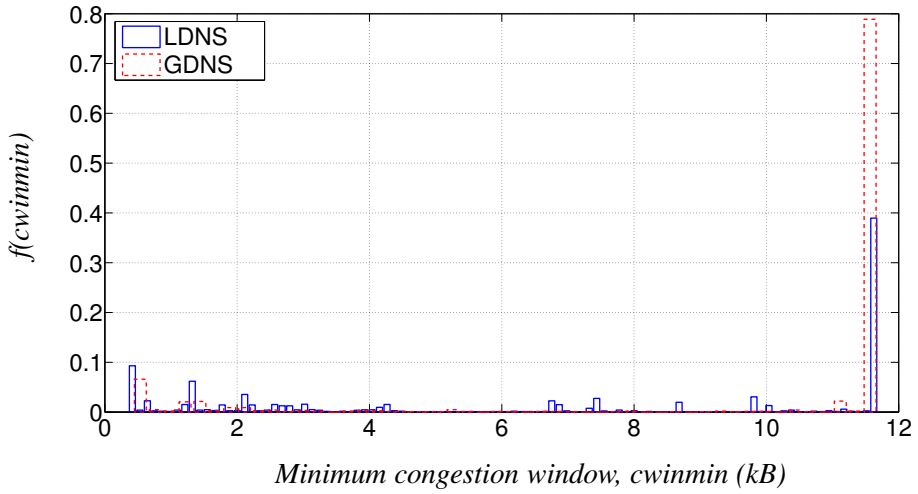


Figure 6.8: Histogram of the minimum congestion window for the Local DNS (LDNS) and Google DNS (GDNS)

vs 20 ms)[1]. Second, by comparing the expected gain in performance corresponding to the two interventions we studied in our causal study. On the one hand, we could observe that the users of the LDNS service are redirected to closer servers, this redirection was estimated to represent a 14% gain of performance. On the other hand, the expected gain in performance when we modify the minimum congestion window of the servers accessed by the LDNS users was estimated to a 30% additional gain in performance. Comparing these two gains allows us to explain the throughput experienced by the GDNS service users and indicates how to improve the throughput of the users of the LDNS service.

The impact of the server TCP stack parameter tuning is studied in [DRC+10]. This work mainly focuses on short connections but shows the improvements that tuning the initial congestion window will have on performance. It is important to notice that, in causal models such as the one presented in Figure 6.3, a given node named according to a parameter X also represents the influence that external factors impacting *only this parameter* have on the rest of the system. Having said so, we can extrapolate our hypothesis on the minimum congestion to the TCP stack tuning parameters (such as the additive increase value for each acknowledged packet) and suppose that the servers that the Google DNS redirects its clients to have a more aggressive TCP parameterization that allows the servers to reach faster a higher value of their ***sending rate*** (number of packets sent at each TCP round).

[1] Note that we are referring to expected value, not arithmetic average as in Table 6.2

6.5 Concluding remarks

In this study, we present a new approach to study the effect of the DNS service on throughput performance. Using a causal approach, supported by the inference of causal model represented by a Bayesian graph, we are able to study the causal effect of a DNS service on the TCP throughput. We compare the performance of clients using their ISP local DNS service to the performance of clients using the Google DNS service. The causal model we obtain allows to unveil dependencies that would be very difficult to extract otherwise from data such as RTT, loss frequency, DNS service, throughput, and so on. This study only focuses on clients downloading files bigger than 2 MB from Akamai servers. However, it appears that the choice of the DNS server has a strong impact on the location of the servers to which the clients are redirected to, which impacts both the distance from clients to servers as well as the servers configuration. These two facts are captured by the dependence between the DNS and the RTT as well as the dependence between the DNS and the server minimum congestion window in the inferred causal model.

A very interesting property of causal models is their "*stability under intervention*". The model inferred from data following a given distribution is still valid when we want to predict the effect of modifying this distribution. We predict, in Section 6.4.2, the performance a client using the local DNS service would have experienced if redirected to servers the Google DNS service would have redirected it to. In this way, we can estimate the impact of the local DNS service redirection strategy on the client performance and quantify the corresponding performance difference.

When comparing the performance of the local DNS and the Google DNS users, we can observe that Google DNS users experience a throughput whose difference with the one of the local DNS service users cannot simply be explained by the redirection of Google DNS users to more distant servers. Based on the causal graph obtained in Section 6.4.1, we can formulate the hypothesis that the configurations of such servers allow the Google DNS users to eventually experience a performance similar to the one of the local DNS service users. This hypothesis is verified by our prediction consisting in giving to servers serving the local DNS users a minimum congestion window equivalent to the one of the servers serving Google DNS users. We estimate the gain in throughput corresponding to this intervention to be 32%. By comparison, the gain in terms of throughput corresponding to the better redirection of the local DNS users is estimated to 14%.

Compared to the studies presented in Chapter 4 and Chapter 5, we demonstrated the real potential of adopting a causal approach. Our approach highly inspired by counterfactuals (see [Pea09] for a formal approach to counterfactuals) are one of the possible way to approach Causality and this work follows

this line. We evaluate the effect of a parameter on the system performance by predicting the effect that changing its parent would have with the rest of the system parameters left unchanged. We manage to answer questions such as "How would the system behave under the condition $C1$ if one of his parameter was behaving as it would have under the condition $C2$, knowing that these two conditions are exclusive ?". The ability to predict such scenarios is a very powerful usage of the inherent mechanisms underlying the development of Causality. Counterfactuals are relatively complex to study, explain and even more to predict. However, with the usage of Bayesian networks as representation of the causal model of our system, we make the prediction of counterfactuals simpler to understand and to estimate.

Complex interventions where many parameters are modified simultaneously require important resources in terms of data and computational power. The results presented in this study represent the first successful attempt to perform such a study. Based on this work, we are confident in the fact that the underlying tools and methods can be improved to reduce the required resources and increase both, the accuracy of such predictions and the range and complexity of the interventions that one can consider. In particular, the extension of prediction of such counterfactuals in cases where the two conditional probabilities have only partial overlap is a limitation of the presented approach. A study of the impact of resource shortage and the impact the parameterization of the method used in this study, is presented in Appendix B. As mentioned in the previous studies, to extend the range of interventions that we can predict we would need a parametric model (Normal, Gamma, Beta, and so on) of the parameter distributions. However, since we deal with multi modal and irregular distributions, such modeling is complex and our preliminary studies trying to fit mixture of know distribution, using the Rebmix package [NF11], did not show encouraging results.

6.6 Lessons learned from the DNS service study

The study presented in this chapter unveils the potential of a causal approach. In Chapter 4, we focused on the different problematics inherent to the study of telecommunication networks. In Chapter 5, we used the lessons learned from the study presented in Chapter 4 to set up and validate the different methods to solve these problematics. In this chapter, we use the own expertise obtained in the FTP study, presented in Chapter 5, to focus exclusively on the benefits of a causal study and to extend its use to understand more complex mechanisms.

6.6.1 Artificial dataset

In this section we try to insist, once more, on the importance of validating the different methods, tools or assumptions that our approach uses.

As we did in the previous chapter, an artificial dataset was generated to validate the use of the KCI test with the presence of a categorical variable, see Appendix A.2. This study showed that the KCI performs well in the presence of categorical data. An important lesson learned in our work is the necessity to verify the compatibility of the methods used with the constraints of the problem being addressed. The oversight of methodological constraints, especially in the use of the independence test implemented in the PC algorithm, could lead to errors that would easily cascade to errors at every step of our causal study and would make the interpretation of such study a complex task.

In addition, we generated a series of datasets and made several studies that show the importance of the parameterization choices on the accuracy of our approach, see Appendix B.

6.6.2 Data acquisition and pre processing

In the study of the DNS impact on CDN user performance, we did not perform the observations of the network traffic ourselves. The data used in this chapter belongs to Politecnico di Torino. It is now common to use data obtained from measurements made by third parties. However, while this does not necessarily appear in the study, this fact has an important impact on the causal study. An important risk to take into account when designing a causal study is the possible presence of latent variables or selection variables. Selection variables are hidden variables that condition the fact that a sample is present in our dataset (for example we never have in our dataset connections that failed).

In the case of the DNS study, it was very important to understand the data acquisition process: the location of the probe and the traffic that was recorded. While such considerations might seem obvious or irrelevant to the party providing the data, it can have a very important impact in the inference of the causal model as parameter dependencies are not always easy to interpret and it is necessary to know how each parameter was estimated to understand the mechanisms that could create a dependence between two parameters.

An example is the IP address of the clients contacting the DNS servers. In a first study we translated the IP addresses of the clients the same way that was used for the destination IP. However, the causal model we obtain was very disappointing, exhibiting many incoherent dependencies that we could not explain. Finally, when looking into the TCP logs (from Tstat) we could see that the client IP addresses indicated origins from many different places. We eventually learned that, for security reasons, the IP addresses of the clients were

obfuscated.[2]. This also highlights the importance of data pre processing. In Figure 6.9, we present the locations of the different destination IP addresses. We can clearly see that clients that use the DNS service of their ISP service are redirected too servers closer than the ones contacted by the clients using the Google DNS service. As suggested by other studies [HMLG11, AMSU10], it is important to spend an important part of a causal study understanding what is present in the observations. These observations are the basis on which the causal model will be inferred and the interventions predicted.

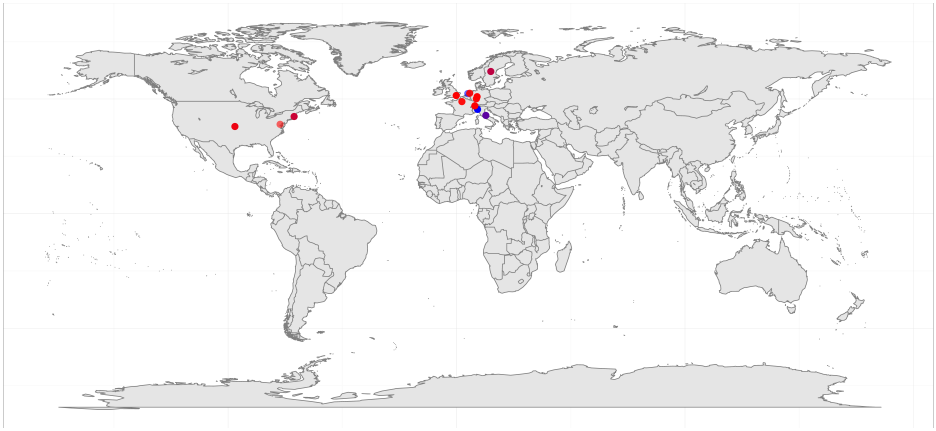


Figure 6.9: Repartition of the different destination IP for the two different DNS services, LDNS (blue) and GDNS (red)

6.6.3 Validating the method and then the data

To conclude, we would like to insist on the importance of the validation of the methodology and the data validation, and the importance to proceed in that order. Even if it may seem obvious, the reason why it is important to validate the methodology first is to discard this source of potential error and be able to focus on the data itself. If we had to work on both points in parallel, we would very likely enter in an infinite loop where we try to solve methodological problems based on incorrect assumptions made on the data and the way it was generated or measured. Or, we would try to understand wrongly detected dependencies in our causal model by looking deeper into the observations.

[2]Notice that, as all the clients should belong to a given ISP, we should have noticed the problem before, but we use this example as it easily shows the importance of understanding the data acquisition process.

Chapter 7

Conclusions

In this chapter, we summarize the different problems that we had to face in our work and the solutions that have been designed in response to these problems. We insist on the process that led us to the different methods used in the different studies presented in Chapter 4 - 6.

In our work, we dedicated an important amount of time and resources to understand the different mechanisms supporting a causal study and their implementations in a domain different from the fields where a causal approach is usually used. The main objective of our work has been to show that performing a causal study is a challenging task but that the results we obtain are worth the effort requested to adapt a causal approach to a new domain. Our work should provide a fertile basis for future work and some directions are given in Chapter 8.

7.1 Objectives and achievements

In this section, we first review the previous chapters, summarizing the different steps that we went through to build the framework that is used in the FTP studies, Chapter 5, and in the study of the DNS service impact on CDN throughput performance, in Chapter 6. In a second part, we try to summarize the different lessons that have been learned during the development of our different solutions and the three studies that have been presented in this thesis.

7.1.1 Progress

The objective of our work was to study telecommunication networks. Telecommunication networks are dynamic, heterogeneous, distributed systems where many actors collaborate to define the user performance. The vision we have of

telecommunication networks is often limited as we might not have access to all the parameters responsible for the performance that can be observed at a given point of a network. As the number of parameters increases, understanding the relationships between them becomes rapidly impossible to handle, unless a formal approach is used.

A peculiarity of telecommunication networks comes from the lack of assumptions that can be made on their behaviors. The parameters capturing the mechanisms responsible of their performance are a mixture of continuous, discrete and categorical data. Their distributions are usually presenting irregular modes that prevent any approximation using know distributions, in particular normal distributions. The dependencies between the different parameters are often unknown and cannot be approximated by a known functional model, in particular linear dependencies.

The interest in adopting a causal approach comes from the possibility to predict the effect of intervening on a system using passive observations. In the case of telecommunication networks, we are dealing with complex systems where experiments are usually not feasible. Active experiments where we would isolate a part of a network and intervene on its components to test the effect of changes that we would like to apply more globally is difficult. The complexity comes from both, the difficulty to recreate conditions that are representative of the Internet network and the economical impact of deteriorating the performance that the users of a given network are expecting to (or paying for) have. As we could see in the study of the FTP traffic performance in an emulated network, Section 5.1, emulations suffer from the same limitations. As it was mentioned in the introduction, the Internet network is too big and too complex to obtain a single model that could be used to predict intervention through active experiments.

The first step in our work has been to build the necessary knowledge of the causal theory, and its software tools, that could be used in our work. At the end of this first step, we focused on the causal model inference part of a causal study, presented in Chapter 4 where a simple approach is adopted. We use the PC algorithm and kPC algorithm to study the performance of the Dropbox application. While the PC algorithm assumes linearity and normality, the Weakly Additive Noise model, on which the kPC relies, was shown to be verified in almost every scenario where the dependencies are not linear and the parameters are normally distributed [TGS09]. Hence the choice of these two algorithms. After several unsuccessful attempts to capture the mechanisms of the Dropbox application, it appeared quite clearly that our approach could not succeed and needed to be revised. However, this study allowed us to understand many considerations that needed to be taken into account in the design of a causal study. The necessity to validate the different elements of our approach and to find a scenario where such validation could be done.

Our next step has been to redefine our problematic and list the different constraints that make the study of our system different from the ones found in the literature. It first appeared that the Z-Fisher criterion could not be used in our work and that the independence test present in the kPC algorithm implementation suffered from poor performance. The first objective, then, had been to find an independence criterion that would correctly capture the dependencies of telecommunication networks parameters. The Kernel Conditional Independence test, presented in Section 3.2.2, seemed to perfectly fit our needs but suffered from two drawbacks. The first one being the use of Cholesky factorization in the implementation of the KCI test that often makes the test fail. The second was the completion time that, compared to the Z-Fisher criterion, was very high. We solved these two issues by coupling the KCI to a bootstrap method that circumvented the numerical issue related to the Cholesky factorization and that allowed simple parallelizations to decrease the completion time of the algorithm of causal model inference. The KCI+bootstrap method implied introducing several parameters that needed to be tuned and validated. The study presented in Section 3.3 showed that the KCI+bootstrap was performing as well as the KCI run in its native form and was able to solve the two issues it was designed to overcome.

A second challenge, which does not appear so clearly in the literature at first, is the difficulty to estimate the, possibly highly dimensional, distributions of the parameters of the system being studied. In the case where the parameters are discrete, as it is often the case in the examples given in [Pea09, SGS01], the problem is greatly simplifies. However, due to the curse of dimensionality, the number of intervals to quantize our multivariate distributions would be too important to model efficiently the different distributions that we are using in our approach therefore, another approach was adopted. Copulae turn out to be a simple and accurate solution to model highly dimensional PDFs.

After putting the different pieces of our method together, it was still necessary to test its accuracy and verify that it could be used in practice. The study of the FTP performance in the emulated network, Section 5.1, allowed us to validate our method as we could perform the intervention that our method allowed us to predict. The results we obtained were very concluding and allowed us to perform the study of the FTP application performance in the Internet network and exhibit interesting dependencies, that domain knowledge and intuitions alone might not have identified.

Finally, the two FTP studies were successful and showed the benefits of adopting a causal approach to the study of telecommunication network based application performance. However, the two studies were mainly focusing on the implementation challenges that need to be overcome in order to exploit the the benefits of a causal approach. In a third study, presented in Chapter 6, we could abstract the underlying implementation constraints to focus on the methodology. Our study of the DNS service impact on CDN user throughput

performance showed that the causal model we inferred is not only able to capture mechanisms ruling the performance of new telecommunication network technologies (CDN) but can also discover dependencies that were not seen before and allowed to propose solutions to improve performance of, already optimized, systems.

7.1.2 Contributions

In this section, we present the main contributions of our work. As the topic of our work has been the adaptation of existing methods to the causal study of telecommunication networks, our contributions are mainly focused on the methodology.

Methodology

In the different studies presented in the previous chapters we could show how to avoid the main pitfalls inherent to a causal study by keeping in mind a series of considerations when designing a causal study:

Scenarios Not all scenarios are eligible to a causal study. One should somehow foresee the model that he could obtain and the use that he would make of this model. An common mistake is to expect that by giving messy data to a machine learning algorithm a clean model will be inferred. Unfortunately, this is not the case and time and efforts will be spared if a special care is taken in the design of the causal study.

Level of details of a study Assuming that we decide to perform the causal study of a system that, we believe, would benefit from a causal approach. The level at which the system will be represented is an important decision. At a too high level, the model would not highlight any interesting dependencies, at too low level we take the risk of modeling mechanisms requiring an important number of parameters to be modeled, and could suffer from the presence of latent variables. Additionally, if we try to model mechanisms that rely on hard-coded protocols we do not want to try to model deterministic mechanisms that would violate the faithfulness assumption necessary for our algorithm to infer a system causal model.

Level of abstraction A subtle difference with the previous point is the choice of the parameters that will form our system model. Each parameter captures, in its variations, the influence that all the mechanisms and external parameters have on this parameter. The choice of the parameters of our model should be dictated by the level of abstraction that the system model is aiming at. This choice should also consider the possible latent

variables present in the intended model. Coming back on the example of the *windshield wipers* and *car accidents*, if we want to study the probability of *car accidents* and we do not see beforehand that the *rain* is a latent variable that influences both, the usage of *windshield wipers* and the probability of *car accidents*, introducing only the *windshield wipers* variable will introduce the presence of a latent variable that will make the detection of the dependence between the two variables *windshield wipers* and *car accident* more difficult. The problematic of feature selection is a very complex problem, which we briefly addressed in Section 1.3.4.

Data preprocessing As highlighted by the study of the impact of the DNS service on CDN user throughput in Chapter 6, understanding the input will help to understand the causal dependencies inferred by the causal model inference algorithm to detect errors in the measurement process or parameter estimations. Outliers are important for the prediction of interventions but the opposite is often true for the inference of causal dependencies. In our work, we often restrict our input data to the samples falling in the $[\mathcal{P}_5, \mathcal{P}_{95}]$ during the stage of the causal model inference. Where $\mathcal{P}_X$ represents the X^{th} percentile of a given parameter observed values.

Domain knowledge Domain knowledge is necessary at every step of a causal study: in the choice of the parameters, in the interpretation of the causal dependencies, and in the interpretation of the predictions.

Our work underlined the limitations of the application of a causal study to real world applications. Once the assumptions of normality, linearity and discreteness removed, most of the existing independence tests become inaccurate and the inference of a system causal model becomes more complex. Identically, a causal model allows to infer the equations that will define the distributions of the parameters post intervention from their distributions pre intervention. However, the estimation of multi dimensional distributions where no assumption can be made, is a complex problem that prevents the use of the causal theory in many scenarios. Finally, from the do-calculus one can read that the absence of certain observations in the input dataset prevents the prediction of many interventions and little importance has been given to this consideration in the literature that formed the basis of our work. Working with incomplete data is an important and complex issue [DKCDS12, GM04].

However, in our work, we showed that such challenges could be overcome by adopting and adapting the correct methods to the constraints of our problem. With the use of the KCI+bootstrap method for testing independences and the use of copulae to model conditional probability density functions, we manage to show that the causal theory can be applied to system more complex than the ones where we usually see it applied.

Results

Our work shows, through examples, that causal studies can, as claimed, be applied to a range of problems and, while special care has to be taken when the systems are more complex, the tools we present offer a good accuracy and exploitable results. The models inferred in our work uncover not only the dependencies suggested by the literature but also new dependencies that, while totally supported by the mechanisms behind telecommunication networks performance, would have been difficult to detect otherwise.

In our study of the FTP server performance, in Section 5.2, the causal model presents the receiver window, the loss frequency and the RTT as direct causes of the throughput, the parameters known to influence the throughput. But the model also detects a dependence between loss events and RTT, that was not present in previous models of TCP performance. It also presents a dependence between the RTT and the receiver window, that turned out to be a hidden auto tuning mechanism used by clients to adapt the resources dedicated to a given TCP connection to the delay between them and the server. The presence of the empirical parameters (the loss, the delay and receiver window) as direct causes of the throughput comforts us on the pertinence of the model, and the presence of dependencies that were not suggested before highlights the potential of a causal approach and its benefits when compared to previous empirical approaches such as the one of the PFTK model.

In the study of the DNS service impact on CDN user throughput, Chapter 6, our model shows that the DNS service impacts the CDN servers that are contacted by the clients. Again, this dependence had been noticed before and the fact that our model exhibits this dependence supports the validity of our model. In addition, the model detects a dependence between the DNS service and the speed at which the data is sent from the servers contacted by DNS service users. This dependence is somehow unexpected and allows us to propose a direction to improve the performance of clients using the DNS service of their ISP.

Finally, through the different studies that have been presented we could see that the causal approach is very valuable to understand complex systems. Once adopting a formal approach that takes into account the different constraints of the systems being studied, we are able to predict complex scenarios that help us understand the role of the different parameters of a system in its global performance. While other systems, such as WISE, have been designed with the same objective, it is our formal approach that allows us to perform complex predictions while using limited resources. To our knowledge, resorting to and estimating counterfactuals to understand and estimate the causal impact of a parameter (DNS service) on a system performance (throughput) was never done in telecommunication networks before.

7.2 Lessons learned

The previous section presented the different steps that constitute our work. During each of these steps we could draw conclusions and take away important lessons that helped us to design or improve our approach to the causal study of a given system. In this section, we present the different lessons that have been learned during this thesis.

This section briefly summarizes the different lessons that have been learned during the different causal studies of Chapters 4, 5 and 6, presented in Section 4.5 - 6.6.

One notion that might seem obvious and appears at every step of our work is the important difference between theory and practice. The causal theory is not limited to any system, even the study of direct effects through the single-door criterion, that requires linear dependencies, can be extended to non linear dependencies as long as the correct precautions are taken and the correct methods are used. However, to put this in practice requires an important amount of work both, to find the correct methods to apply the theory and to implement these methods for real data.

Towards this end, the different studies and the design of our solution to study telecommunication networks taught us a series of necessary steps that we see as requirement to perform a correct causal study.

The first one being to always know the data: the underlying assumptions or constraints, the way such data was obtained, what is the meaning of each parameter and what scenario they are modeling.

The second one is to always validate the methods of the solution that we adopt. Once we have found or have designed the methods that meet the constraints exhibited by the data, it is important to perform studies in a controlled environment where we are able to test, validate and calibrate the different methods that will be used in the different studies they were designed for.

Finally, the last point that is inherent to any data mining related problem is the data itself. It is the data that will in the end dictate the accuracy, pertinence, and quality of a study. The amount of information captured by the data is what matters and will condition all the steps of a causal study. Therefore, one should always try to obtain as much data as possible to observe as many situations as possible and be able to capture more mechanisms.

Summarizing the two previous steps, one should collect its data with the objective of maximizing the chances to predict several interventions. This choice requires taking into account the different implementation choices that were taken and the different limitations that such choices have, to optimize their utilization and the accuracy of a given study.

Chapter 8

Future work

In this chapter, we present some directions for continuing our work and extend it to further usages or applications. In a first section, we try to provide some insights on the improvement of the method presented and used in the previous chapters. In a second section, we present some domains where the approach presented in this thesis could also be beneficially used.

8.1 Method

8.1.1 Extension and improvement

Parameter distributions

In the studies presented in Chapter 5 - 6 we had to face a limitation inherent to any data driven problem: data shortage. Data shortage limits both the accuracy of our predictions and the interventions that could be predicted.

Our use of copulae is very simple and, while not part of our work, a deeper understanding of the inference and the use of Copulae to model multi dimensional distribution, would indubitably improve the accuracy of our results. However, the use of Copulae to model the dependencies between the marginals estimated using normal kernels is highly dependent on the input data. Copulae and kernels are highly impacted by missing data.

As mentioned several times in our studies, one solution would be to model the distribution of a parameter with a mixture of known distributions (Normal, Log normal, Weibull or Gamma):

$$f(x) = \prod_i \omega_i \Psi_i(\Theta_i, x) \tag{8.1}$$

If we obtain functional models of the marginals, we can still use Copulae to model their dependencies but extend our knowledge of parameter distributions to domains where they were not observed. An extension would be very beneficial for our predictions that would not be limited by our observations any more.

However, as mentioned previously, most of the parameters present in our studies have irregular multi modal distributions. We might be able to approximate their distribution in the domain of observation with models as shown in Equation (8.1) but such models become rapidly inaccurate if they are used to predict the values of parameter distributions out of the domains where they were initially observed. The problem of modeling a parameter distribution from its incomplete observation is briefly addressed in the study presented in the Appendix C.

Direct effects

One reasonable expectation when modeling the causal dependencies of parameters with a Bayesian network is to obtain the weights of the edges of the graph representing the causal model of a system. As an edge represents the direct effect of one parameter on another, the computation of the edge weights is equivalent to compute the direct effect between each pair of parameters whose corresponding nodes are adjacent. In the case where the functional dependencies are not linear, [Pea13] proposes a solution to estimate the direct effect of a parameter on another. However, this solution requires an important amount of work that one should undertake only if he is really interested in the exact value of the direct effect of one parameter on another.

Another possibility, conferred by our modification of the KCI test, would be to observe the different p-values of the sub tests and, based on their variations, deduce the confidence that we can have in a given dependence. However, for an edge between X and Y ($X \rightarrow Y$) this would require to consider all the p values of all the conditional tests of $X \perp\!\!\!\perp Y|Z$ and to verify with different studies of artificial datasets that the results we obtain are consistent.

The direct effect of one parameter, X, on another, Y, can indicate that a change in X would have a important impact on Y and can, therefore, orient the choice of the intervention that could lead to the best improvement of the system performance. However, as the dependencies cannot be modeled by linear forms, we first need to measure the "strength" of the direct effect. In addition, let us suppose that X_1 has a direct effect on Y that can be measured and is found to be θ_1; and that the parameter X_2 has a direct on Y measured as θ_2. If X_1 has a standard deviation σ_1 and X_2 a standard deviation σ_2, then, even if $\theta_1 > \theta_2$, the fact that $\sigma_1 \ll \sigma_2$ could indicate that bigger changes on X_2 values are possible and then favor the intervention on X_2 instead of X_1, despite the opposite suggestion given by the $\theta_.$ values. For example even if the direct effect of the retransmission score on the throughput is measured as 10 and the one of the

RTT on the throughput is measured as 2, if the variation of the retransmission score is in the order of 10^{-3}, while the one of the RTT are in the order of 10^{2}, one might be more inclined in intervening on the RTT.

Therefore, the study of the directs can be an interesting extension of the graphical causal models we used in our studies, under the condition of defining the usage these direct effects.

Confidence criterion

Another aspect of creating the model of given system, not specific to our use of Bayesian networks, is the confidence that we can have in our model. In the study presented in Section 5.1, it is the accuracy of our predictions that defined the accuracy of the underlying model. In Tetrad software [SGS01], the Bayesian information criterion is proposed as a criterion for measuring the accuracy of the model. As we cannot assume any functional dependencies and distributions for the parameters of the systems we are studying„ the problem is more complex in our studies.

One solution would be the use of the values of the sub test p-values. Very low and stable p-values for the different sub tests of the different tests that were made to test the independence between two parameters whose corresponding nodes finally appear adjacent in the final model, would mean that we have a strong confidence in the presence of a dependence between the two parameters. However, this approach would mean, again, to consider all the p-values of all the tests of $X \perp\!\!\!\perp Y|Z$ for all the X and Y which nodes are adjacent in the final model. As we are dealing with combinatorial problems, the correct design and automation would need be well thought to decrease the resource consumption of this approach. Note that, this would give us the confidence for all detected dependencies but the same approach should be used for the absence of a dependence between two parameters, trying to obtain high and stable p-values instead.

8.1.2 Usage

One drawback of our method comes from its implementation that, while benefiting from the parallelization of the different tests, still requires an important amount of time to infer a causal model from the observation of the different parameters of a system. Similarly, to predict intervention no GUI or automation was designed at the time of this writing. The implementation of our solution to predict interventions relies on the estimation of conditional probability density functions and no automatic way to derive back-door adjustment equation from the adjacency matrix representing our causal model has been implemented

yet. Our work mainly aims at proving the feasibility of a causal study and its benefits. Towards this end, the ergonomic aspect of our solution to study the causal dependencies of complex systems was not part of our work. However, to generalize the use of a causal approach to study complex systems, some work could be dedicated to the design of an easy-to-use solution that would make the study of complex systems from a causal perspective easier to non experts and more attractive to broader range of systems.

Two interesting extensions of work would be

1. The design of a solution adapted to a highly distributed and parallel computational facility (cloud computing) for the inference of the causal model that would significantly decrease the completion time of our algorithm. A scenario in which we could obtain the causal model corresponding to an input dataset in real could even be considered. This solution would also allow to add, remove features and samples to better study the different effects of the parameters of the system being studied and the quality of our input data.

2. The implementation of a solution that requires less input from the user of the algorithm and can directly answer what if questions. Ideally, to vulgarize the use of a causal approach for the understanding of complex system we would need to design a solution that makes its use very easy for non expert. A ergonomic graphical user interface where the user does not need to know the do-calculus and the d-separation criterion to predict intervention would be very interesting. A software where the user only needs to provide a given dataset and can obtain the corresponding causal model and asks what-if questions, as in the study of the effect the DNS service on the throughput of the users of the Akamai CDN, Chapter 6, could be considered, even if out of the topic of our work.

These two extensions would make our solution very similar to the WISE system [TZV+08], that relies on a Map/Reduce framework [DG08] and developed a basic grammar to enable the asking of what if questions. The important difference being our approach to causal inference, the accuracy of our predictions and also the range of intervention that our system could predict as mentioned in Section 6.5. For a deeper comparison between our solution and the WISE system, we refer the reader to the Section 2.3.

8.2 Domain of application

At the time of this writing, we are carrying out a study on how to explain the impact of the different parameters of a cluster on the performance of an appli-

cation running on Spark [HKZ+15]. Spark is a framework for cluster computing that, unlike MapReduce paradigm, allows querying several times the data loaded in cluster's memory and therefore Spark significantly improves performance in many applications where the data has to be queried several times. In particular, Spark is well suited to address machine learning and data mining related problematic. The use of a causal approach is very interesting in a domain where enough knowledge is present to know the parameters that impact the system performance but where too many factors need to be taken into account if a naive approach was adopted.

As mentioned in Section 1.2.6, the access to the Internet network via mobile broadband access has seen its importance drastically increase in the past years. With the ease with which observations can be made and the important number of parameters to take into account when modeling the performance of a given user browsing the Web with his smart-phone, the adoption of a causal approach could also be very beneficial to study mobile broadband access performance. In particular, Google phone and Google store offer an important number of tools to measure the different parameters that impact the performance of a given application related to the device features as well as the different parameters capturing the network behavior. For example, an application developed at the TUM (Munich, Germany), "*measrdroid*"[1], collects different parameters measures from different users under different conditions. Such a scenario represents a very good opportunity to perform a causal study and predict how changes in different parameters will impact the network throughput of mobile users.

Eventually, causality can be of great use for studying security [DGK+15]. We have very recently started a work related to security where we try to predict the likelihood of a given infection based on different features learned from the behavior of certain users. While still in an early stage, we believe that obtaining a causal model of the impact of an user behavior on its risk of being infected could be of real interest to domain such as risk mitigation, security and the field of cyber insurance.

[1] http://www.droid.net.in.tum.de/

Appendices

In the following chapters we present the different studies that were made to validate, or discard, a given solution, or approach, to the different problematics that were faced in our work, namely, testing independences, coping with data shortage and estimating the distribution of our model parameters with a mixture of known parametric distributions.

In Appendix A, we focus on the issue of testing independence between parameters that follow different distributions. We first compare the test that was used in our work, the KCI test, with the Z-Fisher criterion present in the different implementations of algorithms of causal model inference that we tested initially. We then validate the use of the KCI test in the presence of categorical data.

In Appendix B, we highlight the different constraints mentioned in the study of the impact of the DNS service on the throughput performance of the users of the Akamai CDN, Chapter 6. We exhibit the limitations inherent to working with a limited amount of data and we present the different tests that were made to measure the impact of working with incomplete data.

In Appendix C, we present a preliminary work that tries to fit parametric models to the distributions of some of the parameters present in our work. This study relies on the Rebmix package [NF11] that was used to model the distributions of parameters such as the throughput of the FTP study, Section 5.2, with a mixture of distributions from the families of Log normal, Normal, Beta, or Gamma.

Appendix A

Additional results and studies related to the independence tests

In this chapter, we first present the different tests that were made to compare the performance of the commonly used Z-Fisher criterion and the Kernel Conditional Independence test (KCI) that was used in our work. These studies show clearly the superiority of the KCI and the limitations of the Z-Fisher criterion. In a second section, we present the study that was made to validate the usage of the KCI test in the presence of categorical variables.

A.1 Comparison of the Z-Fisher criterion and KCI test

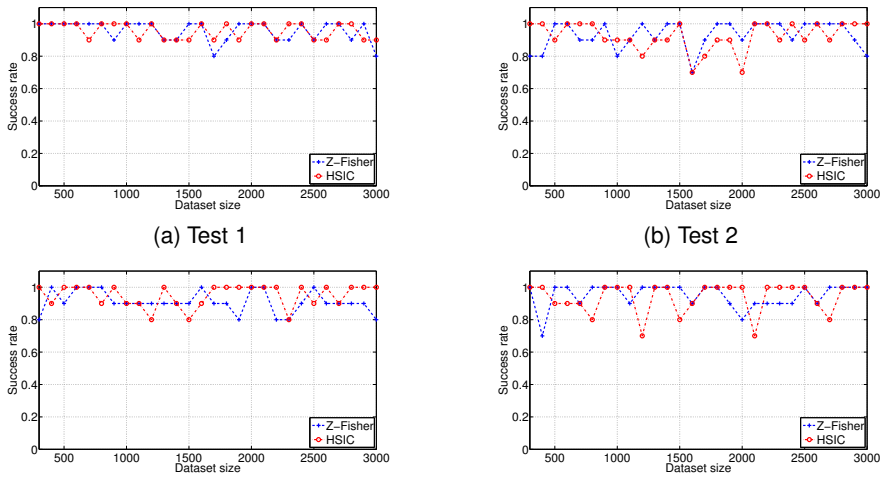
In this appendix we present the different tests that were executed to compare the performance of the KCI and Z-Fisher criteria. We generate artificial data with different functions and under different conditions.

Unconditional tests

In these experiments we generate independently two parameters, X and Y, and test, for different scenarios, the independences with the Z-fisher criterion and the KCI test. The results are summarized in Figure A.1, where the different scenarios are tested 10 times for different sample sizes and the percentage of correct answers are presented. The variables are generated as follows:

- **Test 1** [Normal distribution, same variance]: The values for both parameters are drawn independently from two normal distributions $\mathcal{N}(0, 1)$

- **Test 2** [Normal distribution, different variance]: The values for both parameters are drawn independently from two different normal distributions, $X \sim \mathcal{N}(100, 5)$, $Y \sim \mathcal{N}(5, 2)$
- **Test 3** [Real parameter distribution, same variance]: We randomly draw samples from the RTT and Throughput data values of Table 5.3, and normalize them[1]
- **Test 4** [Real parameter normal distribution, different variance]: We randomly draw samples from the RTT and Throughput data values of Table 5.3



(a) Test 1 (b) Test 2

(c) Test 3 (d) Test 4

Figure A.1: Success rate of unconditional independence tests for the Z-Fisher criterion and KCI test, for the different cases and data set sizes: Test 1 : Normally distributed and same variance, Test 2: Normally distributed and different variance, Test 3: Not normally distributed and same variance, Test 4: Not normally distributed and different variance

These results do not show any preference for one criteria over the other in any of the situations. They also do not show a clear improvement when the sample size increases.

Conditional tests

In this section we compare the independence criteria (Z-Fisher and Hilbert Schmidt Independence Criterion) in the case of conditional independence tests. We restrict ourselves to a conditional set of size 1. We have two possible configurations with three variables **X**, **Y** and **Z**. In the first configuration we have

[1] $X_{normalized} = \frac{X-\mu_X}{\sigma_X}$

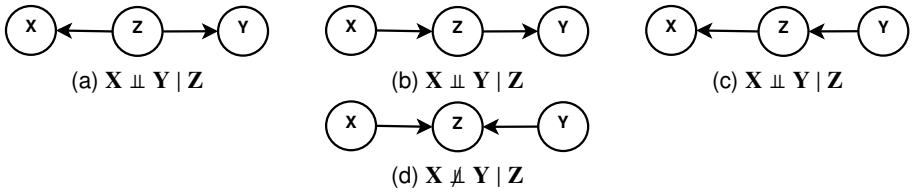


Figure A.2: Different graphical configurations for orienting V-structure

$\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$ and in the second one we have $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$ and $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}$. For each configuration, several graphical representations, corresponding to the mechanisms generating these independences, exist. Some are presented Figure A.2. However, the situations presented in Figure A.2b and Figure A.2c are similar, so we will not differentiate them (inverting the role of **X** and **Y** will not give more information on the performance of a criteria). We will only study the cases represented by Figure A.2a, Figure A.2b and Figure A.2d.

These structures of three variables are very important in causal model inference as they represent the V-structures allowing the PC algorithm to start orienting some edges which, in turn, will induce other orientations.

For each case there several possibilities:
•**Distribution:** normal / not normal
•**Dependences:** linear / not linear
•**Variance:** proportional / disproportional
•**Error terms:** presence / absence

These parameters can have an influence on the outcome of the independence test and will correspond to one series of tests for each case. For each of the three configurations we run 16 tests and, for each test, different input sizes are used. The different tests are presented in Tables A.1-A.3 and their corresponding results in Tables A.4-A.6.

For the normally distributed case, we draw samples from a normal distribution, for the non normal case we draw samples from the observations of the RTT and throughput from the real case scenario dataset, Table 5.3. As previously, for the normally distributed case, we select mean and variance according to the case we are testing. For the non normal cases, if we want similar variance we normalize the samples and if we want different variance we do not normalize the samples from RTT and throughput.

The first observation is that, when the dependences are deterministic, both criteria, rightfully, fail. The second observation is that the KCI has a success rate close to 80% or more over all the scenarios. On the other hand, the Z-Fisher criterion correctly detects only the dependence in the cases {2,4,10,12}, where

the relationship between **X** and **Z** and **Y** and **Z** is linear.

As a conclusion, the KCI test always correctly detects the independences and dependencies, independent of the distribution of the variables and the nature of their dependencies. While Fisher does not seem to difficulties with data that are not normally distributed, the absence of linearity makes it fail on every test of conditional independence. This observation led us to conclusion that the Fisher test is not appropriate for our data.

Test	**Z**	**Functions**	$\sigma_{\mathbf{Z}}$	Error Terms
1	$Z \sim \mathcal{N}(0,1)$	$X = 5 \cdot Z$ $Y = -3 \cdot Z$	proportional	None
2	$Z \sim \mathcal{N}(0,1)$	$X = 5 \cdot Z$ $Y = -3 \cdot Z$	proportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$
3	$Z \sim \mathcal{N}(0,1)$	$X = 2 \cdot Z$ $Y = -300 \cdot Z$	disproportional	None
4	$Z \sim \mathcal{N}(0,1)$	$X = 2 \cdot Z$ $Y = -300 \cdot Z$	disproportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$
5	$Z \sim \mathcal{N}(0,1)$	$X = \sqrt{5 \cdot Z^{\circ}}$ $Y = -3 \cdot \sqrt{Z^{\circ}}$	proportional	None
6	$Z \sim \mathcal{N}(0,1)$	$X = \sqrt{5 \cdot Z^{\circ}}$ $Y = -3 \cdot \sqrt{Z^{\circ}}$	proportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$
7	$Z \sim \mathcal{N}(0,1)$	$X = 3 \cdot \sqrt{500 \cdot Z^{\circ}}$ $Y = -2 \cdot \sqrt{3 \cdot Z^{\circ}}$	disproportional	None
8	$Z \sim \mathcal{N}(0,1)$	$X = 3 \cdot \sqrt{500 \cdot Z^{\circ}}$ $Y = -2 \cdot \sqrt{3 \cdot Z^{\circ}}$	disproportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$
9	Z = not normal	$X = 5 \cdot Z$ $Y = -3 \cdot Z$	proportional	None
10	Z = not normal	$X = 5 \cdot Z$ $Y = -3 \cdot Z$	proportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$
11	Z = not normal	$X = 2 \cdot Z$ $Y = -300 \cdot Z$	disproportional	None
12	Z = not normal	$X = 2 \cdot Z$ $Y = -300 \cdot Z$	disproportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$
13	Z = not normal	$X = \sqrt{5 \cdot Z^{\circ}}$ $Y = -3 \cdot \sqrt{Z^{\circ}}$	proportional	None
14	Z = not normal	$X = \sqrt{5 \cdot Z^{\circ}}$ $Y = -3 \cdot \sqrt{Z^{\circ}}$	proportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$
15	Z = not normal	$X = 3 \cdot \sqrt{500 \cdot Z^{\circ}}$ $Y = -2 \cdot \sqrt{3 \cdot Z^{\circ}}$	disproportional	None
16	Z = not normal	$X = 3 \cdot \sqrt{500 \cdot Z^{\circ}}$ $Y = -2 \cdot \sqrt{3 \cdot Z^{\circ}}$	disproportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$

Table A.1: 16 scenarios for testing conditional independences with the Z-Fisher criterion and KCI test, for the case illustrated in Figure A.2a, $Z^{\circ} = Z - min(Z) + 1$.

Test	Z	Functions	$\sigma_{\mathbf{Z}}$	Error Terms
1	$X \sim \mathcal{N}(0,1)$	$Y = 5 \cdot X$ $Z = -3 \cdot Y$	proportional	None
2	$X \sim \mathcal{N}(0,1)$	$Y = 5 \cdot X$ $Z = -3 \cdot Y$	proportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$
3	$X \sim \mathcal{N}(0,1)$	$Y = 2 \cdot X$ $Z = -300 \cdot Y$	disproportional	None
4	$X \sim \mathcal{N}(0,1)$	$Y = 2 \cdot X$ $Z = -300 \cdot Y$	disproportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$
5	$X \sim \mathcal{N}(0,1)$	$Y = \sqrt{5 \cdot X^{\circ}}$ $Z = -3 \cot \sqrt{Y^{\circ}}$	proportional	None
6	$X \sim \mathcal{N}(0,1)$	$Y = \sqrt{5 \cdot X^{\circ}}$ $Z = -3 \cot \sqrt{Y^{\circ}}$	proportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$
7	$X \sim \mathcal{N}(0,1)$	$Y = 3 \cot \sqrt{500 \cdot X^{\circ}}$ $Z = -2 \cdot \sqrt{3 \cdot Y^{\circ}}$	disproportional	None
8	$X \sim \mathcal{N}(0,1)$	$Y = 3 \cot \sqrt{500 \cdot X^{\circ *}}$ $Z = -2 \cdot \sqrt{3 \cdot Y^{\circ}}$	disproportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$
9	X = not normal	$Y = 5 \cdot X$ $Z = -3 \cdot Y$	proportional	None
10	X = not normal	$Y = 5 \cdot X$ $Z = -3 \cdot Y$	proportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$
11	X = not normal	$Y = 2 \cdot X$ $Z = -300 \cdot Y$	disproportional	None
12	X = not normal	$Y = 2 \cdot X$ $Z = -300 \cdot Y$	disproportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$
13	X = not normal	$Y = \sqrt{5 \cdot X^{\circ}}$ $Z = -3 \cot \sqrt{Y^{\circ}}$	proportional	None
14	X = not normal	$Y = \sqrt{5 \cdot X^{\circ}}$ $Z = -3 \cot \sqrt{Y^{\circ}}$	proportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$
15	X = not normal	$Y = 3 \cot \sqrt{500 \cdot X^{\circ}}$ $Z = -2 \cdot \sqrt{3 \cdot Y^{\circ}}$	disproportional	None
16	X = not normal	$Y = 3 \cot \sqrt{500 \cdot X^{\circ}}$ $Z = -2 \cdot \sqrt{3 \cdot Y^{\circ}}$	disproportional	$\varepsilon_{X/Y} \sim \mathcal{N}(0, 0.1)$

Table A.2: 16 scenarios for testing conditional independences with the Z-Fisher criterion and KCI test, for the case illustrated in Figure A.2b, $X^{\circ} = X - min(X) + 1$, $Y^{\circ} = Y - min(Y) + 1$.

A.2 Testing the KCI algorithm in the presence of categorical data

In Chapter 6 one of the parameter of our system, the DNS, is categorical. In the KCI [ZPJS12], few mentions to the presence of a categorical variable in the independence tests are given.

We designed a simple scenario to test the performance of the KCI (+bootstrap)

Test	Z	Functions	$\sigma_{\mathbf{Z}}$	Error Terms
1	$X \sim \mathcal{N}(0,1)$ $Y \sim \mathcal{N}(0,1)$	$Z = 5 * X - 3 * Y$	proportional	None
2	$X \sim \mathcal{N}(0,1)$ $Y \sim \mathcal{N}(0,1)$	$Z = 5 * X - 3 * Y$	proportional	$\varepsilon_Z \sim \mathcal{N}(0, 0.1)$
3	$X \sim \mathcal{N}(5,10)$ $Y \sim \mathcal{N}(20,100)$	$Z = 5 * X - 3 * Y$	disproportional	None
4	$X \sim \mathcal{N}(5,10)$ $Y \sim \mathcal{N}(20,100)$	$Z = 5 * X - 3 * Y$	disproportional	$\varepsilon_Z \sim \mathcal{N}(0, 0.1)$
5	$X \sim \mathcal{N}(0,1)$ $Y \sim \mathcal{N}(0,1)$	$Z = -3 * \sqrt{(X+Y)^\circ}$	proportional	None
6	$X \sim \mathcal{N}(0,1)$ $Y \sim \mathcal{N}(0,1)$	$Z = -3 * \sqrt{(X+Y)^\circ}$	proportional	$\varepsilon_Z \sim \mathcal{N}(0, 0.1)$
7	$X \sim \mathcal{N}(5,10)$ $Y \sim \mathcal{N}(20,100)$	$Z = -3 * \sqrt{(X+Y)^\circ}$	disproportional	None
8	$X \sim \mathcal{N}(5,10)$ $Y \sim \mathcal{N}(20,100)$	$Z = -3 * \sqrt{(X+Y)^\circ}$	disproportional	$\varepsilon_Z \sim \mathcal{N}(0, 0.1)$
9	X = not normal Y = not normal	$Z = 5 * X - 3 * Y$	proportional	None
10	X = not normal Y = not normal	$Z = 5 * X - 3 * Y$	proportional	$\varepsilon_Z \sim \mathcal{N}(0, 0.1)$
11	X = not normal Y = not normal	$Z = -5 * X - 300 * Y$	disproportional	None
12	X = not normal Y = not normal	$Z = 5 * X - 300 * Y$	disproportional	$\varepsilon_Z \sim \mathcal{N}(0, 0.1)$
13	X = not normal Y = not normal	$Z = -3 * \sqrt{(X+Y)^\circ}$	proportional	None
14	X = not normal Y = not normal	$Z = -3 * \sqrt{(X+Y)^\circ}$	proportional	$\varepsilon_Z \sim \mathcal{N}(0, 0.1)$
15	X = not normal Y = not normal	$Z = -2 * \sqrt{3 * (5 * X + 300 * Y)^\circ}$	disproportional	None
16	X = not normal Y = not normal	$Z = -2 * \sqrt{3 * (5 * X + 300 * Y)^\circ}$	disproportional	$\varepsilon_Z \sim \mathcal{N}(0, 0.1)$

Table A.3: 16 scenarios for testing conditional independences with the Z-Fisher criterion and KCI test, for the case illustrated in Fig.A.2d, $X^\circ = X - min(X) + 1$.

in the presence of categorical variables. As in the study of the parameterization of the KCI+bootstrap procedure, in Section 3.3.2, we generate an artificial dataset. We generate 4 parameters whose dependencies have a corresponding causal model presented in Figure A.3 and test different independences that should be entailed by such system.

We generate two different datasets. One for which only one parameter, X_1 in the model of Figure A.3, is categorical and another for which two parameters, X_1 and X_3 in the model of Figure A.3, are categorical.

The two dataset of 1000 samples are generated as follows:

Test	500				1000				1500				2000				2500				3000			
	$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$	
	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H
1	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0.1	0
2	0	0	0.8	1	0	0	1	1	0	0	0.9	0.9	0	0	1	0.8	0	0	0.8	0.9	0	0	0.9	0.8
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	1	1	0	0	0.8	0.9	0	0	1	1	0	0	1	1	0	0	0.9	1	0	0	1	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	1	0	0	0	1	0	0	0	0.9	0	0	0	0.9	0	0	0	0.9	0	0	0	1
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0.9	0	0	0	1	0	0	0	0.8	0	0	0	0.8	0	0	0	0.9	0	0	0	1
9	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	1	1	0	0	1	0.9	0	0	1	0.8	0	0	0.9	0.8	0	0	0.9	1	0	0	1	1
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0.9	0.9	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0.8	1	0	0	0.9	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0.7	0	0	0	0.8	0	0	0	0.6	0	0	0	0.7	0	0	0	0.9	0	0	0	0.9
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	1	0	0	0	0.8	0	0	0	1	0	0	0	0.9	0	0	0	1	0	0	0	1

Table A.4: Results of the 16 conditional independence scenarios for the model in Figure A.2a, tested using the Z-Fisher criterion and KCI test for 6 different data set sizes, averaged on 10 trials. $\mathcal{I}_1 = X \perp\!\!\!\perp Y$, $\mathcal{I}_2 = X \perp\!\!\!\perp Y \mid Z$.

Test	500				1000				1500				2000				2500				3000			
	$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$	
	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H
1	0	0	0	0	0	0	0.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0.9	0	0	1	1	0	0	0.9	0.9	0	0	1	0.9	0	0	0.9	1	0	0	1	0.9
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	1	1	0	0	0.8	0.8	0	0	0.8	1	0	0	0.8	1	0	0	0.9	0.9	0	0	1	1
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	1	0	0	0	1	0	0	0.1	1	0	0	0	1	0	0	0	1	0	0	0	1
7	0	0	0	0.01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	1	0	0	0	0.9	0	0	0	1	0	0	0	0.9	0	0	0	1	0	0	0	0.9
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0.8	1	0	0	1	1	0	0	1	0.9
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	1	1	0	0	0.9	0.9	0	0	1	1	0	0	1	0.9	0	0	1	1	0	0	1	0.9
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0.9	0	0	0	1	0	0	0	1	0	0	0	0.9	0	0	0	1	0	0	0	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0.9	0	0	0	1	0	0	0	1	0	0	0	0.9	0	0	0	0.9	0	0	0	1

Table A.5: Results of the 16 conditional independence scenarios for the model in Figure A.2b for the Z-Fisher criterion and KCI test, for 6 different data set sizes, averaged on 10 trials. $\mathcal{I}_1 = X \perp\!\!\!\perp Y$, $\mathcal{I}_2 = X \perp\!\!\!\perp Y \mid Z$.

Test	500				1000				1500				2000				2500				3000			
	$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$		$\mathcal{I}_1$		$\mathcal{I}_2$	
	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H	F	H
1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	1	0	0
2	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
3	1	1	0	0	1	1	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
4	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
5	1	1	0	0	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
6	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
7	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
8	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
9	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
10	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
11	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
12	1	1	0	0	1	1	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
13	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	0	0	0
14	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	1	1	0	0	1	1	0	0	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0

Table A.6: Result of the 16 tests for the model Fig.A.2d for 6 different data set sizes using the Fisher criterion and KCI test $\mathcal{I}_1 = X \perp\!\!\!\perp Y$, $\mathcal{I}_2 = X \perp\!\!\!\perp Y/Z$.

First dataset : ds1

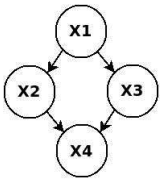


Figure A.3: Causal model of 4 variables
Causal model corresponding to the causal dependencies of the 4 parameters of our artificial dataset to test for KCI performance in the presence of categorical data

- X_1 categorical: randomly drawn, with replacement, from a pool of 4 values
- X_2 continuous: discrete function of X_1 (mapping)
- X_3 continuous: discrete function of X_1 (mapping)
- X_4 continuous: square root function of X_2 and X_3

Second dataset : ds2

- X_1 categorical: randomly drawn, with replacement, from a pool of 4 values
- X_2 continuous: mapping according to X_1 values
- X_3 categorical: drawn from a pool of 4 values, the probability of each category depending on X_1 value
- X_4 continuous: square root function of X_2 plus coefficient depending on X_3

We also add a noise term,$\varepsilon_{X.}$, to all variable, $X.$, with a standard deviation equals to 20% of the variable ($\sigma_{\varepsilon_{X.}} = 0.2.\sigma_{X.}$)

The results of the different independence tests are presented in Table A.7. We denote by **p-value ds1** the median of the p-values obtained in the sub-tests of the KCI+bootstrap procedure, Section 3.2.2, for the tests made on the samples of the first dataset. Accordingly, we denote by **p-value ds2** the median of the p-values obtained in the sub-tests of the KCI+bootstrap procedure for the tests made on the samples of the second dataset.

We can observe that the results that we obtain are consistent with the ground truth. Suggesting that the KCI+bootstrap procedure performs correctly in the presence of Categorical variable. Additionally, the results show that the KCI+bootstrap performs well in the case where one of the two parameters between which the independence is tested is categorical ($X_1 \perp\!\!\!\perp X_4|\{X_2, X_3\}$ for the first dataset). It also shows that it is still the case when one member of the conditioning set is categorical ($X_2 \perp\!\!\!\perp X_3|\{X_1\}$ for the first dataset). Eventually, with the second

dataset, we can observe that the KCI+bootstrap procedure still performs correctly in a mixed case ($X_1 \perp\!\!\!\perp X_4 | \{X_2, X_3\}$ for the second dataset).

Table A.7: Results of the different independence tests made with the KCI+bootstrap for the two artificial datasets

X	Y	Z	p-value ds1	p-value ds2
1	2		0	0
1	3		0	0
1	4		0	0
2	3		0	0
2	4		0	0
3	4		0	0
1	4	2	0	0
1	4	3	0	0
2	3	1	0.5	0.9
1	4	{2,3}	0.3	0.9
2	3	{1,4}	0	0

Appendix B

Different methods to estimate the distribution of the throughput post intervention in the DNS study

In the study of the DNS impact on the throughput, Chapter 6, we could observe that the distribution of the intervention variable, X, conditionally to the GDNS (Google DNS) allowed to predict the distribution of the throughput for LDNS (Local DNS) users. We could also observe that the opposite, predicting the GDNS users throughput for an intervention where X follows the distribution seen by the LDNS users, was not possible. In this document we artificially generate datasets with parameters and dependencies simulating the situation we had to deal with in the DNS performance study. We vary the conditional distributions of the intervention parameter to minimize the number of values observed for both distributions ($f(X|LDNS)$ and $f(X|GDNS)$) and we study its impact on the prediction accuracy. We also expose, with more details, the method developed to predict counterfactuals.

B.1 Problematic

The Figure B.1 represents the causal model we inferred in the study of the DNS impact on user throughput in Chapter 6. We used the PC algorithm [SG91] and the kernel based independence criterion from [GFT+08] to infer the Bayesian graph representing the causal model of our system.

The DNS is a categorical variable that can take only two values: $Local\ DNS$ ($LDNS$) or $Google\ DNS$ ($GDNS$).

The aim of our study was to estimate the impact of the DNS service on the user performance, measured by the throughput.

From the causal model, presented in Figure B.1 we can try to predict the effect of the DNS choice on two parameters: the minimum congestion window (*cwinmin*) and the average external delay (*inetrttavg*). To evaluate this impact we estimate the distribution of the throughput (*tput*) for the users of one DNS service when intervening on the minimum congestion window or the external delay and give to these parameters the distributions of clients using the other DNS.

For the external delay for GDNS users, if given the distribution of LDNS users, we obtain:

$$
\begin{aligned}
&f(tput \mid GDNS)_{f(rtt|do(LDNS))} = \\
&\iint f(tput \mid do(rtt), GDNS, ts) f(ts) f(rtt \mid do(LDNS)) P(LDNS) = \\
&\iint f(tput \mid rtt, GDNS, ts) f(ts) f(rtt \mid LDNS) P(LDNS)
\end{aligned} \tag{B.1}
$$

with *rtt* representing the average external rtt (*inetrttavg*).

For the minimum congestion window for LDNS service users, if given the distribution of GDNS users, we obtain:

$$
\begin{aligned}
&f(tput \mid LDNS)_{f(cwin|do(GDNS))} = \\
&\iint f(tput \mid do(cmin), LDNS, ts) f(\sigma_{rtt}) f(cmin \mid do(GDNS)) P(GDNS) = \\
&\iint f(tput \mid cmin, LDNS, \sigma_{rtt}) f(\sigma_{rtt}) f(cmin \mid GDNS) P(GDNS)
\end{aligned} \tag{B.2}
$$

with σ_{rtt} representing the standard deviation of the external rtt (*inetrttstd*) and cmin standing the minimum congestion window (*cwinmin*).

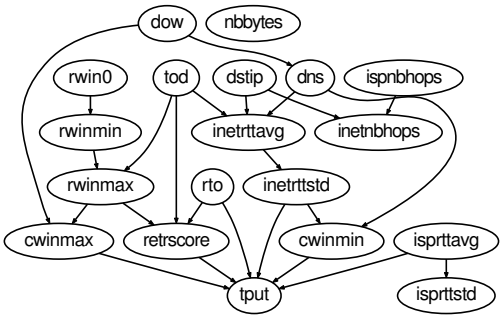


Figure B.1: Causal model of DNS traffic performance

B.2 Implementation

B.2.1 Overview

We can see from Equation (B.1) and Equation (B.2) that the PDF of the throughput of users of the DNS d_1 if we fix X distribution to follow the distribution of X for the users of DNS d_2, is a weighted sum of the PDF of throughput for DNS = d_1 after an atomic intervention on X ($do(X{=}x)$). In this case the weights can be seen as the probability of observing $X = x$ under DNS = d_2.

These considerations implies that

1. We must find enough samples where d_1 and $X = x$ is observed to estimate the PDF of $f(Tput|DNS = d_1, X = x, Z = z)$, where Z verifies the back-door criterion for $X \rightarrow Y$.
2. We must find X values observed for both DNS, as for the values of X, conditionally to dns = d1, we need to multiply the post atomic PDF by the quantity $f(X|DNS = d2)$

Figure B.1 and Figure B.3 represent the distribution of the average external rtt ($inetrttavg$) and the minimum congestion window ($cwinmin$) for the two DNS (denoted d_1 and d_2).

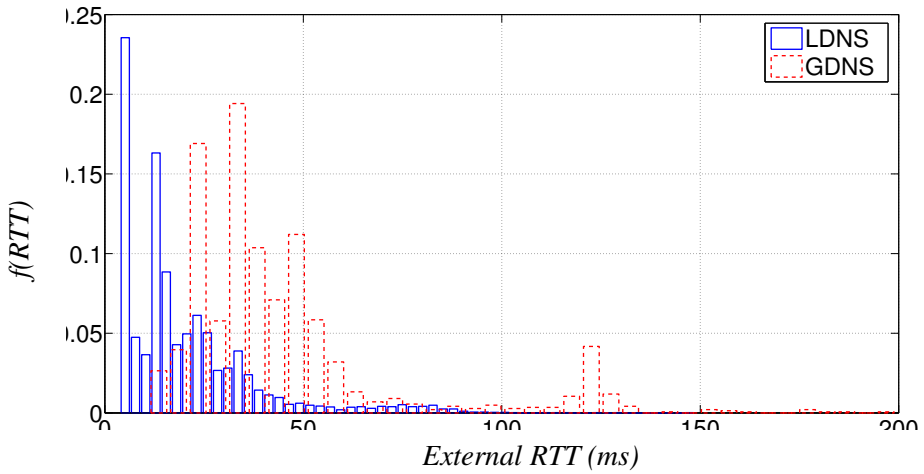


Figure B.2: Histogram of the external RTT for the LDNS and GDNS

B.3 Practical issues

It should be noticed that, as X is a continuous variable, the probability of observing a given value is 0. Therefore, instead of selecting samples for which X

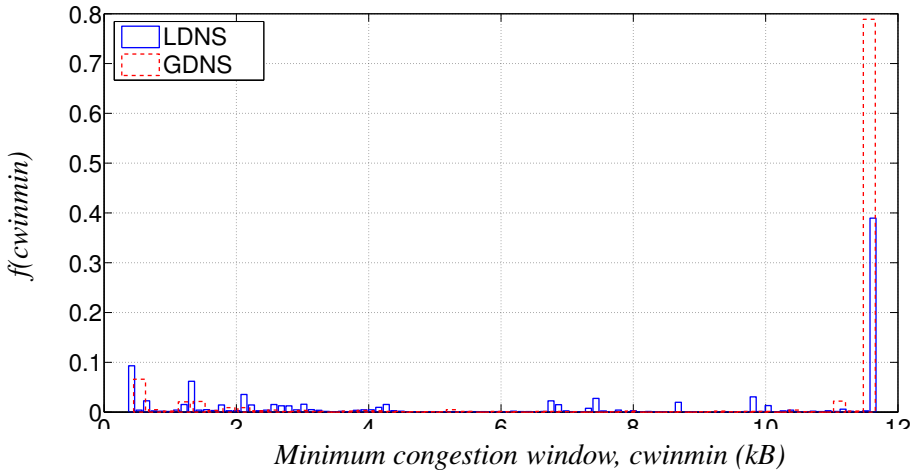


Figure B.3: Histogram of the minimum congestion window for the LDNS and GDNS

$= x$ is observed, we define an interval I_x corresponding to $[x - \delta_X; x + \delta_X]$ and assume that the samples for which the X parameter value falls into this interval, the value is approximated by x.

From these samples, we estimate the cumulative distribution function (CDF) of the throughput and Z (for the *inetrttavg* parameter $Z = \{DNS, ts\}$ and for the *cwinmin* parameter $Z = \{DNS, inetrttstd\}$, as we first condition on a given DNS we have $Z_{inetrttavg} = \{ts\}$ and $Z_{cwinmin} = \{inetrttstd\}$). After selecting the connections (= samples) where the DNS service we want to condition on is present, we use normal kernels to estimate the CDFs of *tput* and Z.

Using a Gaussian copula or a T-copula we estimate $f(Tput|X = x, Z = z, DNS = dns)$ using the equation:

$$f(Tput|X = x, Z = z, DNS = dns) = \\ c(F(Tput|X = x, DNS = dns), F(Z|X = x, DNS = dns))f(Tput|X = x, DNS = dns) \quad \text{(B.3)}$$

Integrating Equation (B.3) on Z we obtain the PDF of

$$f(Tput|do(X = x), DNS = dns) \quad \text{(B.4)}$$

We then select the samples for which $DNS \neq dns$ and use normal kernels to estimate $f(X|DNS \neq dns)$ and frequencies to estimate $P(DNS \neq dns)$.

After these steps, we have all the factors present in Equation (B.1) or Equation (B.2) and we can integrate over X to obtain the probability distribution function of the throughput post intervention.

However, some practical issues remain:

1. How to define the intervals I_X ?
2. As we are working with continuous variables, the two PDFs conditionally to different DNS do not share the same domain. How do we define the conditional probability so that we have common value to integrate on ?

The last point is solved by always defining the PDFs domains as equally spaced points between the minimum and maximum observed value of the corresponding parameter in the whole dataset. We use normal kernel to estimate the different PDFs.

The first point, however, is more complicate as many possibilities exist and there is the constraint of finding enough samples in each interval to estimated the CDFs from which the copula parameters will be estimated and the conditional PDFs derived.

Several solutions have been tested

- Variable bin width histogram
- Fixed bin width and interpolation
- Fixed bin width, filtering, rescaling.

B.3.1 Variable bin width histograms

The first method consists in fixing an objective number of samples and, starting from a fixed width bin histogram, merging adjacent bins until obtaining bins of different size but with a minimum number of samples:

Pros This method ensures the maximum number of atomic predictions being successful

Cons As many of the parameters we observe have a distribution with a long tail, and it is often in the tail of the parameter distributions that we find the values for interesting predictions, we obtain very large bins for the extreme values of the intervention parameter. The approximation of this bin representing a single value is then too important. Additionally, when multiplying the the value of the PDF after an atomic intervention fixing X to the value x by the value of the PDF conditionally to the other DNS for $X = x$ (Equation (B.1) and Equation (B.2)) we need to take into account the actual range of X that the atomic intervention represents to have a consistent approximation. Said differently, if the bin corresponding to X = x under DNS = d1 is covering a big interval, because of data shortage, multiplying the probability of $f(Y|do(X = x), DNS = d1)$ with $f(X|DNS = d2)$ requires that $f(X|DNS = d2)$ is computed from the same bin interval, if such is possible, or weighted according to the bin width.

B.3.2 Fixed bin width

The second method is going in the opposite direction. We use histograms with fixed width bins and try to make predictions of $f(Y|do(x))$ with the number of samples we found in a given bin corresponding to a X value. If the estimation of the prediction fails, then we use interpolation of the $f(Y|do(x)$ for the X values where the post intervention PDF of Y could be estimated.

Pros This method solves the issues of approximation inconsistency of the variable bin width method. We fix the bin width and decide on the approximation of assimilating an interval to a given value.

Cons It can often happen that we manage to predict $f(Y|do(x)$ even when there are few samples in the interval I_x corresponding to the x value. However, with very few samples, it is very likely that this estimation is not accurate. This lack of accuracy impacts in a very negative way the overall post intervention PDF of the throughput accuracy, as the PDF of Y post intervention that could be computed with few values will be used in the interpolation to recover the PDF Y for x values where the estimation of the post intervention PDF failed.

B.3.3 Fixed bin width and high pass filter

The last method, the one eventually adopted, is based on the fact that, if there are very few samples in a given area of the distribution of $X|DNS = dns_1$ then this value is very unlikely to be observed and, as the previous method will impact negatively the overall results by including these samples, we consider that the PDF of $X|DNS = dns_1$ in the domains with few samples is null.

This method uses a fixed bin histogram and selects only the bins where a minimum number of samples is observed to predict atomic interventions. After predicting the distribution of the throughput after intervention we rescale it based on the following observation:

$$\int_{tput} f(tput|do(X = x), DNS = dns_1)dtput = 1 \tag{B.5}$$

This method solves the previous issues. There is a risk of not having any prediction for values in the tail of the distribution of $X|DNS = dns_1$, that can represent the zone of overlap of the two distributions $X|DNS = dns_1$ and $X|DNS = dns_2$. However, limitations due to the lack of samples cannot be overcome by simple approximations as seen in the second method. The only way to solve this issue is to use parametric PDF. The studies that we have made so far show that the approximation of the distributions of the parameters of our

systems for values that are actually observed offers an acceptable accuracy but that these models become inefficient as soon as we try to use them to estimate the PDF of a parameter for values that have not been observed.

B.3.4 Open questions

Eventually, one can ask the following question:

1. In which proportion the absence of values observed for both conditional distributions impacts the prediction accuracy and how to improve the accuracy in this case ?
2. How to choose the bin width ?
3. Should we favor the number of atomic predictions from which the final PDF is estimated or the quality of the atomic intervention predictions by increasing the number of samples from which these atomic predictions are computed ?

To study these aspects we need a ground truth to estimate the accuracy of our prediction for different strategies and parameterization. Therefore, we generate a set of artificial datasets where the intersection between the two conditional PDF (DNS = d_1 and DNS = d_2) domains is modified and its impact on the accuracy of our method is studied.

In practice, instead of varying the bin width and the threshold defining when a bin is given a 0 probability, we do the following: We select an objective number of samples, Δ_S, under which the atomic intervention $f(Y|X_2 = 0, do(X = x))$ is considered as null, and search for the optimal quantization leading to the maximum bins with a number of samples $\geq \Delta_S$. To do so we use this very simple algorithm:

B.4 Artificial dataset

B.4.1 Simulated dependencies

To simulate the same situation as the one met in the study of the DNS service impact on the throughput, we randomly generate 4 parameters, X_1, X_2, X, Y, with dependencies illustrated by the graph presented in Figure B.4. To be closer to the situation observed in our DNS study we generate X_1 by randomly re-sampling the throughput observed in the causal study of the DNS impact and we generate X_2 by randomly re-sampling the observed DNS from the same

Data: Vector X
Result: Set of bins (X_{final} with fixed width (δ) defining the intervals around the values on which atomic interventions will be predicted:
initialization;
nB = 10;
nV_{old} = 0;
nV_{cur} = 1;
X_{old} =[];
X_{curr} = [];
H_{old} = [];
H_{prev} = [];
while $nv_{cur} > nv_{old}$ **do**
  $H_t, E_t = hist(X, nB)$;
  $nV_{old} = nV_{cur}$;
  $nV_{cur} = nvalues(H_t > \Delta_S)$;
  $X_{old} = X_{cur}$;
  $X_{cur} = E_t$;
  $H_{old} = H_{cur}$;
  $H_{cur} = H_t$;
  $nB = 2 * nb$;
end
$nV_{final} = nV_{old}$;
$X_{final} = X_{old}$;
$H_{final} = H_{old}$;
$\delta = X_{final}(2) - X_{final}(1)$;

Algorithm 1: Dynamic quantization

study. We eventually convert X_2 to binary value (0 or 1). The presence of a categorical data (the DNS in the corresponding study) is an important aspect that we want to keep in this study.

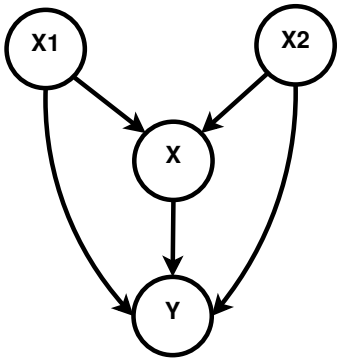


Figure B.4: Artificial dataset dependencies

B.4.2 Intervention prediction

First, from the causal model, G, represented by the Bayesian network of Figure B.4, we can use the d-separation criterion to deduce the following independences:

- $(X \perp\!\!\!\perp Y | X_1, X_2)_{G_{\underline{X}}}$
- $(X_2 \perp\!\!\!\perp X)_{G_{\underline{X_2}}}$

From the do-calculus rules from [Pea09] we can deduce that the distribution of Y under the condition $X_2 = 0$ intervening on X to fix its distribution to $X \sim X|do(X_2 = 1)$ is given by

$$f(Y|X2 = 0, do(X \sim X|do(X2 = 1))) = \int_X \int_{X_1} f_{Y|X,X_1,X_2}(x, x_1, 0) f_{X_1}(x_1) f_{X|X_2}(x, 1) Pr(X_2 = 1) dx_1 dx \quad \text{(B.6)}$$

B.5 Ideal scenario

In this case we generate the dataset as follows:

$\mathbf{X_1}$ = random_re-sampling(DNS)

$\mathbf{X_2}$ = random_re-sampling(Throughput)

$$\mathbf{X} \sim \begin{cases} \Gamma(k_1, \theta_1) + \sqrt{X_1} * \frac{\mu_1}{2} + \varepsilon_X : X_2 = 0 \\ \Gamma(k_2, \theta_2) + \sqrt{X_1} * \frac{\mu_2}{2} + \varepsilon_X : X_2 \neq 0 \end{cases}$$

$$\mathbf{Y} \sim \begin{cases} 10. \sqrt{5.X + 10.X_1} + \varepsilon_Y : X_2 = 0 \\ 25. \sqrt{3.X + 6.X_1} + \varepsilon_Y : X_2 \neq 0 \end{cases}$$

with $\varepsilon_\cdot$ representing an error terms.

In this first case we chose the following values $\{k_1 = 5, \theta_1 = 1.0, k_2 = 2.0, \theta_2 = 2.0, \mu_1 = 5, \mu_2 = 8\}$. The resulting distributions are presented in the Figure B.5. To make sure that the parameters are correctly generated, we infer the corresponding causal model using the PC algorithm [SG91] with the independence test from [ZPJS12].

Notice that, for testing our method under the same main constraint we limit our sample size to 10000 (against 7500 in the real case scenario) and we keep the same proportion between the number of samples where $X_2 = 0$ is observed and the number of samples where $X_2 = 1$ is observed, as it was the case in the real case scenario for the number of connections where the LDNS service was observed (80%) compared to the number of connections where the GDNS service was observed (20%).

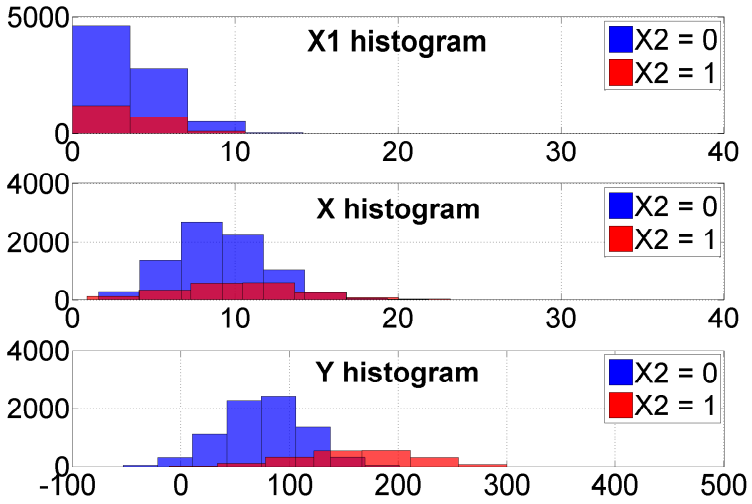


Figure B.5: Distribution of the different parameters for both values of X_2

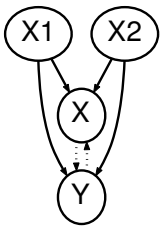


Figure B.6: Causal Model of the first dataset

B.5.1 Prediction of Y given X2 = 0 after intervention on X giving it the distribution of X given X2 = 1

Using the Equation B.6 we are able to compute the PDF $f(Y|X_2 = 0, do(X \sim X|X_2 = 1))$ and obtain the expected value $\mathbb{E}[Y|X_2 = 0, do(X \sim X|X_2 = 1)]$.

As mentioned in Section B.3.4, regarding the method exposed in Section B.3.3, the choice of the number of bins and the threshold to decide or not whether to estimate the post atomic intervention PDF, should have an impact on the prediction accuracy. Consequently, in this ideal scenario where the two distributions of $X|X_2 = 0$ and $X|X_2 = 1$ have very similar domains, we vary these two parameters and study their impact on the prediction accuracy.

To obtain the distribution of $f_{Y|X_2=0,do(X\sim X|X_2=1)}$ we generate X and Y as following:

- $\mathbf{X} \sim \Gamma(k_2, \theta_2) + \sqrt{X_1} * \frac{\mu_2}{2} + \varepsilon : X_2 = 0$
- $\mathbf{Y} \sim 10.\sqrt{5.X + 10.X_1} + \varepsilon$

We obtain an expected value of $\mathbf{E[Y|X_2 = 0, do(X \sim X|X_2 = 1)]}$ of 101.5.

We summarize the different results we obtained in Table B.1.

We can see that the use of a T-copula ($T\text{-}cop$) in the modeling of the multi dimensional PDF gives slightly better results, in terms of accuracy, than the modeling of multidimensional PDF with a Gaussian copula ($G\text{-}cop$). However, very likely due to the functions used for generating the artificial dataset and the use of a Gamma distribution, the T-copula is not always able to model the PDFs $f_{Y|X,X_1,X_2}(y, x, x_1, 0)$ that are necessary to compute the post intervention PDF of Y in Equation (B.6).

From this table, we can notice two additional things:

1. The accuracy of the prediction depends on the distribution of X (already noticed when comparing the two settings in the previous study, one where X is drawn from normal distribution and one from Gamma distribution). We manage to obtain better accuracy (3%) when using normal distribution instead of Gamma distributions (not presented).

2. In the results presented in Table B.1, there is no clear preference for smaller value of Δ_S

It is important to notice that the usage of a Gaussian copula was motivated by the prediction of an intervention in the opposite case ($f_{Y|do(X\ X|X_2=0),X_2=1}(y)$), see next section. In addition, we generated the artificial dataset in order to have the same domains for $X|X_2 = 1$ and $X|X_2 = 1$ and do not observe the same tail dependence as we would have for the throughput in the real case. From this perspective, the usage of T-copula is not fully justified and the usage of a Gaussian copula, for this particular dataset, could be more appropriate.

Table B.1: Effect of varying Δ_S on the prediction accuracy for the first artificial dataset for two different multi dimensional PDFs modeling, using a T-copula ($T\text{-}cop$) and Gaussian copula ($G\text{-}cop$). *#AI* stands for the number of Atomic Interventions

Δ_S	$\hat{\mathbf{E}}\mathbf{[Y|X_2 = 0, do(X \sim X|X_2 = 1)]}$		Error		#A.I.	#Failures	
	G-cop	T-cop	G-cop	T-cop		G-cop	T-cop
20	93.80	96.22	7.8%	5.5%	195	0 (0%)	5 (2.6%)
30	93.28	95.90	8.6%	6.0%	118	0 (0%)	3 (2.5%)
50	95.40	96.79	6.5%	5.2%	76	0 (0%)	2 (2.6%)
70	94.41	96.94	7.5%	5.0%	54	0 (0%)	1 (1.9%))
100	94.24	96.74	7.6%	5.2%	39	0 (0%)	0 (0 %)

B.5.2 Prediction of Y given X2 = 1 after intervention on X giving it the distribution of X given X2 = 0

Without repeating the explanations given in the previous section, we use Equation (B.6) to predict the expected value of $\mathbf{E}[Y|X_2 = 1, do(X \sim X|X_2 = 0)]$.

The results are presented in Table B.2.

Several important remarks can be made from the results presented:

- The usage of a T-copula for modeling the PDF of $f_{Y|X,X_1,X_2}(y, x, x_1, 0)$ for different values of X fails more than 50% of the times.
- The utilization of wider bins, gathering more data to approximate $X = x$, does not improve the the success rate of the prediction of the PDFs post atomic intervention.
- The usage of G-copula gives better results (in terms of prediction accuracy) than the previous prediction of intervention.

Table B.2: Effect of varying Δ_S on the prediction accuracy for the first artificial dataset for two different multi dimensional PDF modeling, using a T-copula (*T-cop*) and Gaussian copula (*G-cop*). *#AI* stands for the number of Atomic Interventions

Δ_S	$\hat{\mathbf{E}}[Y\|X_2 = 1, do(X \sim X\|X_2 = 0)]$		Error		#A.I.	#Failures	
	G-cop	T-cop	G-cop	T-cop		G-cop	T-cop
20	173.9	N.A.	2.5%	N.A.	43	0 (0%)	23 (53%)
30	173.9	N.A.	2.5%	N.A.	32	0 (0%)	20 (63%)
50	175.7	N.A.	1.4%	N.A.	17	0 (0%)	13 (76%)
70	175.7	N.A.	1.6%	N.A.	13	0 (0%)	10 (77%)
100	174.0	N.A.	2.3%	N.A.	9	0 (0%)	8 (89%)

Remarks It should be noticed that, despite the apparent symmetry between the prediction of Y conditionally to X_2 = 1 if we perform an intervention on X where we fix its distribution to the one of $X|X_2 = 0$ and the one the prediction of the value of Y conditionally to X_2 = 0 if we perform an intervention on X where we fix its distribution to the one of $X|X_2 = 1$, the problematic is different. From an external point of view, looking at the completion time and success rate, Gaussian copulae are less data demanding. We generated X_2 by randomly re sampling the DNS parameters from our real dataset. Doing so, we have 80% of the samples where $X_2 = 0$ is observed against 20% where $X_2 = 1$ is observed.

This second scenario shows the sensitivity of our approach to resource limitation. Even in this "optimal scenario" where both conditional PDF of $X|X_2 = 0$ and $X|X_2 = 1$ have the same domains, the shortage of data for the second conditional PDF prevents us from using a model which should be more accurate

(T-copula). This can be seen by comparing the prediction made in this section to the opposite intervention where more data is available and the usage of T-copulae gave more accurate predictions. Eventually, as mentioned previously, we should keep in mind that the generation of an artificial dataset requires many choices (for the distribution of the different parameters, functional dependencies, noise proportion) whose effects cannot be foreseen.

B.5.3 Concluding remarks

In this section we presented the optimal case where we have both conditional PDFs having almost perfectly overlapping domains. We tried to predict interventions where we condition on one value of the categorical data, X_2, and intervene on another parameter, X, and fix its distribution to follow the conditional distribution corresponding to the complementary value of the categorical data. Our results highlight two important things:

1. Our method works and gives accurate prediction (error $< 3\%$) when enough data is present

2. The usage of T-copulae better captures the multi dimensional PDFs dependences but fails when not enough data is present while G-copulae still give accurate predictions.

Again, these conclusions have to be seen in the context of this artificial dataset where we could not faithfully mimic the tail dependencies of the parents of the throughput in our real case scenario nor the variability of the observed values. These choices are inherent to the design of an artificial dataset and should not be seen as a limitation of this study.

B.6 Removing samples in the tail of the distribution

To estimate the impact of the absence of some values in both domains of the conditional PDFs of $X|X_2$, we remove some samples from the dataset where $X_2 = 1$. Figure B.7 represents the original distributions of $X|X_2$ and Figure B.8 represents the resulting distributions of $X|X_2$ after removing samples from the distribution of $X|X_2 = 1$ (we remove the samples for $X_2 = 1$ where $X \in [0, 5] \cup [15, 20]$).

As in the first scenario, we estimate the expected value of Y conditionally to $X_2 = 0$ when we intervene on X and fix its distribution to the one $X \sim X|X_2 = 1$ and the expected value of Y conditionally to $X_2 = 1$ when we intervene on X and fix its distribution to the one $X \sim X|X_2 = 0$

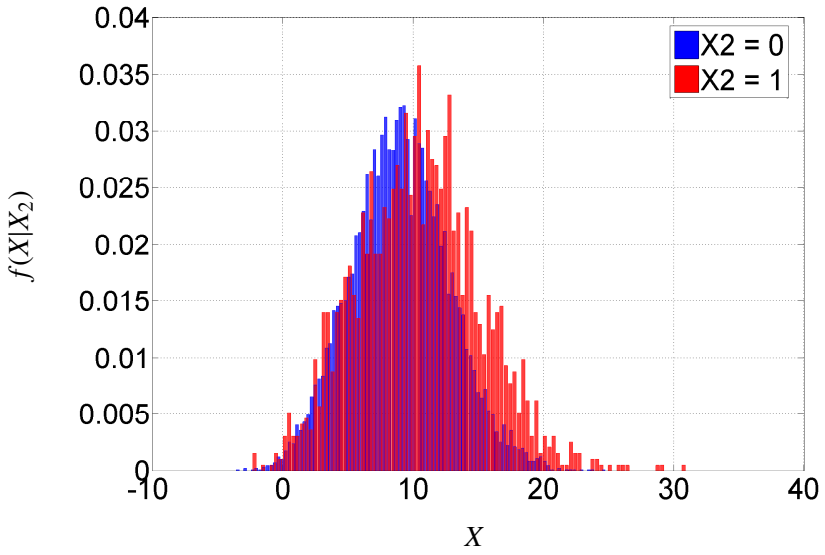


Figure B.7: Conditional probability density function of X conditional on X_2 in the original dataset

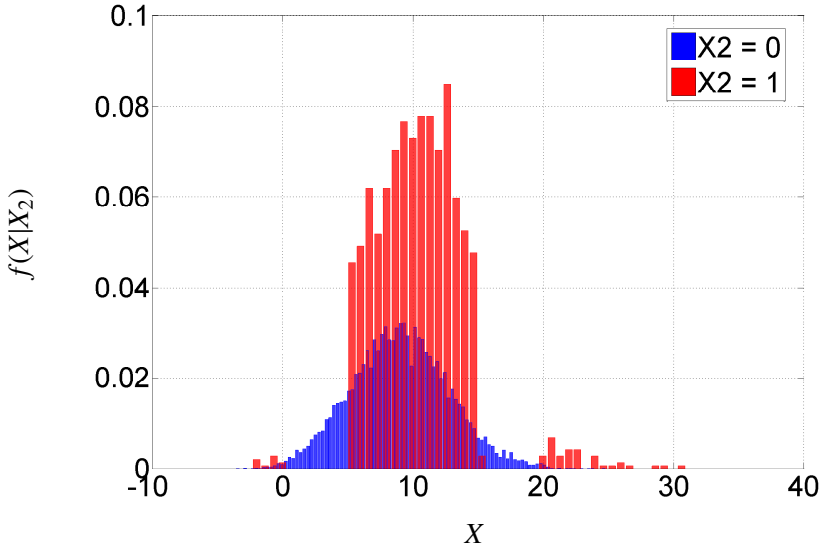


Figure B.8: Conditional probability density function of X conditional on X_2 after removing samples from $X_2 = 1$

We first estimate the effect of intervening on X to fix its distribution to one of $X \sim X|X_2 = 1$. Surprisingly we can see from Table B.3 that the conclusions drawn previously are still valid when we remove samples from the distribution of $X|X_2 = 1$. We can still notice:

- The precision decreases with the increase of Δ_S and the decrease of the number of atomic interventions,
- The number of failures of the multidimensional PDF modeling using a T-copula is sensibly similar.

When looking at the opposite case (conditioning on $X_2 = 1$ and giving to X the distribution of $X \sim X|X_2 = 0$) we can observe that the modeling of the conditional PDFs using a T-copula fails very often and prevent the post intervention PDF estimation, see Table B.4. Surprisingly the estimation of conditional PDF via G-copulae gives very good results.

Table B.3: Effect of varying Δ_S on the prediction accuracy for the first artificial dataset for two different multi dimensional PDF modeling, using a T-copula (*T-cop*) and Gaussian copula (*G-cop*). *#AI* stands for the number of Atomic Interventions

Δ_S	$\hat{\mathbf{E}}[Y\|X_2 = 0, do(X \sim X\|X_2 = 1)]$		Error		#A.I.	#Failures	
	G-cop	T-cop	G-cop	T-cop		G-cop	T-cop
20	96.73	97.12	4.7%	4.3%	195	0 (0%)	5 (2.6%)
40	95.97	96.13	5.5%	5.3%	97	0 (0%)	2 (2.1%)
60	96.39	96.69	5.1%	4.8%	60	0 (0%)	1 (1.7%)
80	95.16	95.87	6.3%	5.6%	48	0 (0%)	1 (2.1%)
100	94.24	95.05	7.2%	6.4%	39	0 (0%)	0 (0.0%)

Table B.4: Effect of varying Δ_S on the prediction accuracy for the first artificial dataset for two different multi dimensional PDF modeling, using a T-copula (*T-cop*) and Gaussian copula (*G-cop*). *#AI* stands for the number of Atomic Interventions

Δ_S	$\hat{\mathbf{E}}[Y\|X_2 = 1, do(X \sim X\|X_2 = 0)]$		Error		#A.I.	#Failures	
	G-cop	T-cop	G-cop	T-cop		G-cop	T-cop
20	176.63	N.A.	0.6%	N.A.	40	0 (0%)	23 (57.5%)
40	176.95	N.A.	0.4%	N.A.	22	0 (0%)	14 (63.6%)
60	178.21	N.A.	0.3%	N.A.	12	0 (0%)	10 (83.3%)
80	179.73	N.A.	1.2%	N.A.	10	0 (0%)	9 (90.0%)
100	177.43	N.A.	0.1%	N.A.	9	0 (0%)	8 (88.9%)

B.6.1 Concluding remarks

For this scenario, where we remove samples from the initial dataset where $X_2 = 1$, we selected samples in regions where the PDF of $f_{X|X_2=0}$ takes small values, to minimize the absence of the observations of X values for both conditional PDFs. The situation is more complex when we want to predict the distribution of $f_{Y|do(X\sim X|X_2=1),X_2=0}$. In this case we predict atomic intervention $f_{Y|do(X=x),X_2=0}$ and assign to each of this PDF a weight taken from the distribution of $X|X_2 = 1$ that,

due to the absence of some of the X values, can take the value 0. However, due to the smooth shapes of the conditional distributions and the usage of generic PDF modeling, our method overcomes these challenges.

The next section presents a more complicated situation where we remove samples in regions where the PDF of $X|X_2 = 0$ takes important values (values of X with high probability under $X_2 = 0$).

B.7 Removing samples in zones away from the tail of the distribution

We generate a new sub dataset with a distribution X to X_2 as represented in Figure B.9. The results for the two interventions are presented below with

- Prediction of expected value of Y after intervention corresponding to PDF of Y: $f_{Y|X_2=0,do(X\sim X|X_2=1)}(y)$ in Table B.5
- Prediction of expected value of Y after intervention corresponding to PDF of Y: $f_{Y|X_2=1,do(X\sim X|X_2=0)}(y)$ in Table B.6

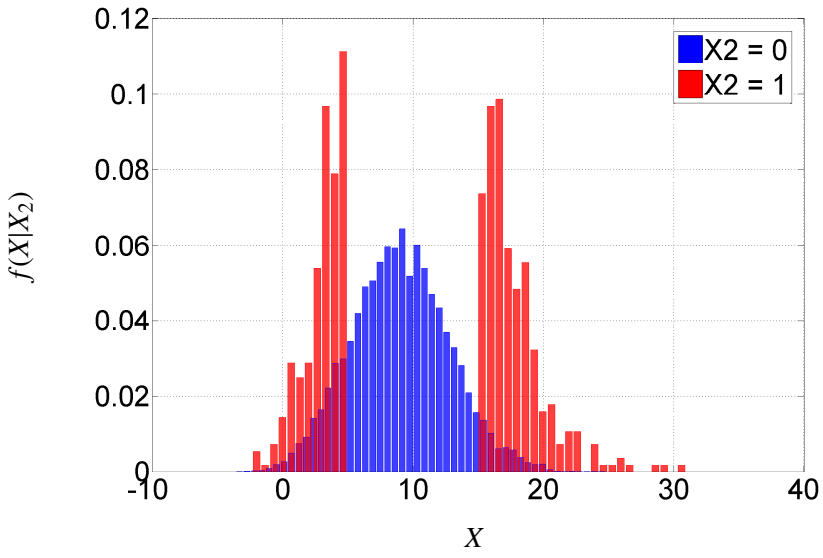


Figure B.9: Conditional probability density function of X conditional on X_2 after removing samples from $X_2 = 1$

Table B.5: Effect of varying Δ_S on the prediction accuracy for the first artificial dataset for two different multi dimensional PDF modeling, using a T-copula (*T-cop*) and Gaussian copula (*G-cop*). *#AI* stands for the number of Atomic Interventions

Δ_S	$\hat{\mathbf{E}}[Y\|X_2 = 0, do(X \sim X\|X_2 = 1)]$		Error		#A.I.	#Failures	
	G-cop	T-cop	G-cop	T-cop		G-cop	T-cop
20	66.79	80.81	34.2%	20.4%	195	0 (0%)	5 (2.6%)
40	76.15	85.80	25.0%	15.5%	97	0 (0%)	2 (2.1%)
60	67.07	82.14	33.9%	19.1%	60	0 (0%)	1 (1.7%)
80	69.55	83.08	31.5%	18.2%	48	0 (0%)	1 (2.1%)
100	N.A.	N.A.	N.A.	N.A.	39	0 (0%)	0 (0.0%)

Table B.6: Effect of varying Δ_S on the prediction accuracy for the first artificial dataset for two different multi dimensional PDF modeling, using a T-copula (*T-cop*) and Gaussian copula (*G-cop*). *#AI* stands for the number of Atomic Interventions

Δ_S	$\hat{\mathbf{E}}[Y\|X_2 = 1, do(X \sim X\|X_2 = 0)]$		Error		#A.I.	#Failures	
	G-cop	T-cop	G-cop	T-cop		G-cop	T-cop
20	153.68	N.A.	13.5%	N.A.	12	0 (0%)	7 (58.3%)
40	154.50	140.18.	13.0%	21.1%	6	0 (0%)	3 (50.0%)
60	157.90	N.A.	11.1%	N.A.	5	0 (0%)	3 (60.0%)
80	163.52	N.A.	8.0%	N.A.	3	0 (0%)	2 (66.7%)
100	157.25	N.A.	11.5%	N.A.	2	0 (0%)	1 (50%)

B.7.1 Concluding remarks

We can observe the accuracy of the prediction of the expected value of Y decreasing drastically. The usage of a T-copula for conditional PDF gives a slightly better accuracy than the prediction based on the usage of a Gaussian copula usage. However with a 20% error we cannot use our method any more.

For the prediction of of the expected value of Y conditionally to $X_2 = 1$ when intervening on X and fix its distribution to $f_{X|X_2=0}(x)$, we penalize the usage of T-copula for modeling conditional PDFs more than the G-copula that seems to be less data demanding.

B.8 Conclusion

Most of the results and conclusions can be found in the concluding remarks sections, Section B.5.3, Section B.6.1 and Section B.7.1. In this section we just recall the objective of this study and summarize the conclusions.

This study was motivated by the study of the impact of the DNS service (ISP DNS vs Google DNS) on user quality of experience (QoE), captured by the

throughput. We made use of the causal theory to predict some counterfactuals that, in turns, allowed us to quantify the causal effect of the DNS choice on user performance. The prediction of counterfactuals was using the estimations of an order 4 conditional probability density function where one of the parameter that was conditioned on, the DNS, was categorical with only two exclusive values. To estimate these PDFs we use different models (histograms, copulaes) with different parameterizations (bin widths) and working under different constraints (intersection of the two conditional PDFs domains). The aim of this study was twofold:

1. Validate the approach in an environment were the intervention we predict can be manually done and our prediction accuracy verified.
2. Study the impact of the different model and parameter choices on the prediction accuracy.

The need to have a ground truth to assess the accuracy of our results led to the generation of an artificial dataset where the parameter dependencies and the parameter distributions meet the same constraints as the ones of the parameters present in the real case scenario, Chapter 6. Namely, having non linear dependencies, a non normal distribution, having one of the parent of the intervention variable being categorical and a second parent that will be used in the application of the back-door criterion to predict atomic interventions.

B.8.1 Results for the ideal scenario

We first generate an artificial dataset where the domains of the two conditional PDFs have an almost perfect intersection (full overlap) and verify that our method gives satisfying results. The results we obtain show that our method perform correctly with a good prediction accuracy ($\leq$ 5% error). We also noticed that the accuracy depends on the distribution of the parameters of our system. This accuracy was shown to be even better (error close or below 2%) when using normal distribution and random re-sampling of throughput for generating X. [1] We could also notice that the use of a T-copula gives slightly better results but requires more data and resources. This constraints was more obvious in the second prediction where, due to the lack of variations / data, the T-copula modeling of the multivariate PDFs for predicting atomic intervention failed in more than 50% of the cases. The use of T-copulae to model conditional PDFs to eventually predict the distribution of Y post intervention failed, while the usage of Gaussian copulae modeling kept giving acceptable results.

[1] The choice of X was dictated by the need to easily remove samples for this study and the question of the unequal influence of $X|Z_2$ on Y compared to the one of Z_2 on Y in the previous study.

These conclusions show also the limitations of generating an artificial dataset where we need to find a trade-off between the control we have on the parameters and the nature and randomness of the parameters that is an very important factor in our method accuracy.

B.8.2 Results for the impact of removing samples in the tail of the PDF

In a second scenario, we remove samples where $X_2 = 1$ is observed and for X values where the PDF $f_{X|X_2=0}$ takes small values. The conclusions from this study were similar to the one for the full dataset. The constraints of working with a limited amount of data was clearer in the usage of T-copulae and we could also see that the prediction accuracy shows a preference towards the number of atomic interventions against the number of samples for predicting these interventions. Gaussian copulae show very good results but the superiority of the Gaussian copulae on T-copulae was assigned to the way we generated our artificial dataset.

B.8.3 Results for the impact of removing samples in the domain away from the tail of the PDF

In a last scenario, we remove samples where $X_2 = 1$ is observed and for X values where the PDF $f_{X|X_2=0}$ takes big values (X values with important probably under $X_2 = 0$). While the conclusions drawn in the previous cases are still valid, we can observe the deterioration of the prediction accuracy for both T-copula and G-copula model of conditional PDF and for all values of Δ_S. This result is not surprising and comfort the remarks made in the causal study of the DNS, in Chapter 6, where we claimed that our method could not be used to predict the distribution of the throughput for the Google DNS service users after an intervention where fix the distribution of the external delay or minimum congestion window to be the one seen by the users of the local DNS service. In the case where no data is available we are not able, unless using parametric model to artificially extend our observation and “create” new observations, to make predictions.

Appendix C

Parametric model of distributions

In this section we try to approximate, with a mixture of distributions from pre defined distribution families (normal, log-normal,Weibull, gamma) the marginals of the FTP parameters, Section 5.2. Our interest in a parametric model comes from its ability to obtain the probability of the value of a given parameter even if this value was never observed. As it can be read from the back-door criterion used in the studies presented in Chapter 5, and from the do-calculus in general, used the DNS study in Chapter 6, we need to estimate the probabilities of the different parameters involved in the equations derived from the different graphical criteria presented in Section 1.3.4 to estimate interventions. Therefore, having access to the probability of all parameters values, we free ourselves from the limitation of data shortage.

This section is organized as follows. First we present Rebmix, the application we tested to estimate the distributions of the parameters present in our study. Second, we test Rebmix software and verify its accuracy in an environment where its accuracy can be measured. Eventually, we study if the estimations we obtain can be use to extrapolate the parameter PDFs outside of the domains where they were observed.

C.1 Rebmix

Rebmix [NF11] is an algorithm to compute finite mixture model for univariate and multi variate distributions. This algorithm is computing the weighted mix of components chosen in one of the following families of distributions: Normal, Log normal, Weibull, Gamma, for continuous densities, and Binomial, Poisson or Dirac for discrete ones.

The algorithm consists in a data preprocessing stage and an optimization stage summarized in the following two sections.

C.1.1 Preprocessing stage

The preprocessing stage consists in the non parametric estimation of marginals with *Histogram* or *Parzen Window* (kernels). In this case the algorithm requires the number of bins or windows to be used to estimate the empirical densities. The k-nearest neighbor approach can also be used.

The width for histogram/Parzen Window is defined by

$$h_i = \frac{y_{imax} - y_{imin}}{k} \tag{C.1}$$

where k is the number of bins.

The origin for histogram is defined by

$$y_{i0} = y_{imin} + \frac{h_i}{2} \tag{C.2}$$

C.1.2 Optimization stage

Let $\mathbf{y_1}, .., \mathbf{y_n}$ be n observations of a d dimensional dataset. Each observation can be modeled by a finite mixture model, represented in Equation (C.3) (bold letters representing vector of dim ≥ 1).

$$f(\mathbf{y}|c, w, \mathbf{\Theta}) = \sum_{l=1}^{c} w_l f(\mathbf{y}|\Theta_l) \tag{C.3}$$

Supposing that every component is independent of the other we have:

$$f(\mathbf{y}|\Theta_l) = \prod_{i=1}^{d} f(y_i|\Theta_{i,l}) \tag{C.4}$$

The aim of Rebmix algorithm is to estimate the parameters c, w_l (summing to 1) and the component parameters $\mathbf{\Theta}_l$

C.2 Testing Rebmix package

C.2.1 First test

In this section we are testing the Rebmix accuracy by randomly generating samples from a mixture of Weibull densities defined as

$$f(x, \omega, \Theta_1, \Theta_2) = \prod_c \omega_c f_{\Theta_{1,c},\Theta_{2,c}}(x), \tag{C.5}$$

where $f_{\Theta_{1,c},\Theta_{2,c}}(x)$ is the probability density function of a Weibull distribution with scale factor $\Theta_{1,c}$ and shape parameter $\Theta_{2,c}$, defined as

$$f_{\Theta_{1,c},\Theta_{2,c}}(x) = \frac{\Theta_{2,c}}{\Theta_{1,c}} \left(\frac{x}{\Theta_{1,c}} \right)^{b-1} e^{-(\frac{x}{\Theta_{1,c}})^{\Theta_{2,c}}} \tag{C.6}$$

The approach is the following

1. Estimate the Weibull mixture fitting the throughput parameter $\Rightarrow \{\omega, \Theta_1, \Theta_2\}$
2. Generate a density function, $f_{original}$ with previously estimated parameters from Equation (C.5)
3. Draw random samples from the distribution of $f_{original} \Rightarrow datasaet_{original}$
4. Try to fit the best mixture of probability density functions from Normal, Log normal, Weibull or Gamma distribution from $dataset_{original} \Rightarrow \{\omega_{est}, \Theta_{1_{est}}, \Theta_{2_{est}}\}$
5. Generate the estimated density function, $f_{estimate}$ from $\{\omega_{est}, \Theta_{1_{est}}, \Theta_{2_{est}}\}$ and Equation (C.5)
6. Generate the estimated density function, f_{kernel}, using a normal kernel
7. Compare $f_{original}$, $f_{estimate}$ and f_{kernel}

The Weibull mixture, $f_{original}$, corresponds to 16 components summarized in Table C.1a.

The best fitting distribution mixture to the data from $dataset_{original}$ is found to be a Weibull mixture of 10 components, summarized in Table C.1b

Figure C.1 plots the three different PDFs $f_{original}$, $f_{estimate}$ and f_{kernel}. The result shows that the model we obtain using Rebmix software offers a very bad accuracy. However the kernel estimate and parametric estimate show an important similarity. It could be due to the parameters in the components of the original mixture leading to a very skewed shape of the original PDF. To verify this hypothesis, in the next section, we repeat the same experience using as original PDF the estimated PDF from this section.

C.	ω_c	$\Theta_{1,c}$	$\Theta_{2,c}$
1	0.065123	14400	10
2	0.03257	1825	50
3	0.36976	200	440
4	0.12487	175	1350
5	0.023471	80	1075
6	0.042934	215	480
7	0.035283	210	570
8	0.056559	190	2150
9	0.088208	1240	380
10	0.0099126	45	4815
11	0.05012	3000	120
12	0.027548	110	4540
13	0.021767	455	595
14	0.0092832	14150	30
15	0.019172	315	1405
16	0.023421	2785	130

(a) Original Weibull mixture components

C.	ω_c	$\Theta_{1,c}$	$\Theta_{2,c}$
1	0.84	1590	17
2	0.036	1595	33
3	0.024	1544	248
4	0.023	1585	227
5	0.030	1491	88
6	0.027	1404	56
7	0.0071	1658	92
8	0.0034	1451	706
9	0.00013	1510	12820
10	0.012	1738	87

(b) Estimated Weibull mixture components

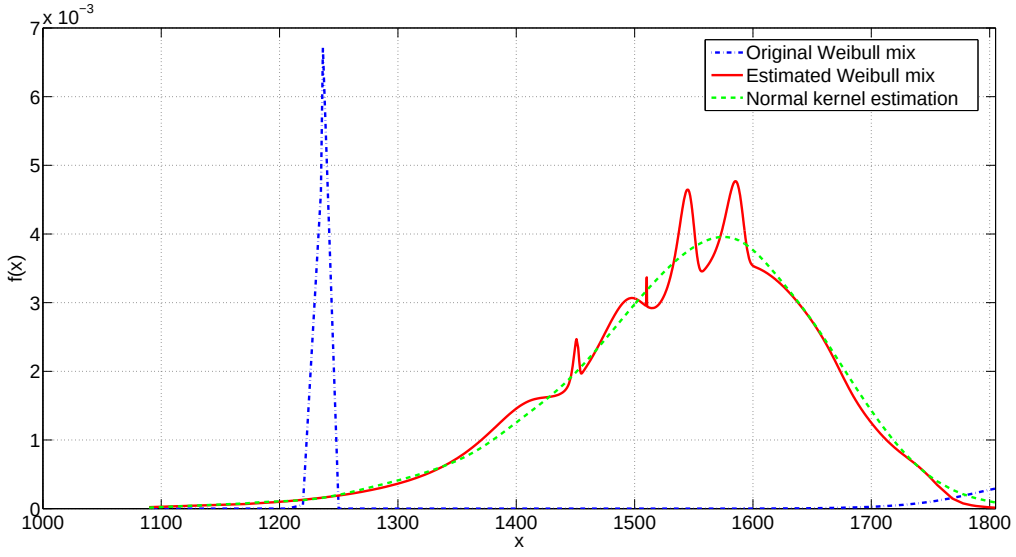


Figure C.1: Comparison of the original and estimated PDFs

C.2.2 Second test

The previous experiment outcome was questioned by the data generation process that was used to test the accuracy of a parametric modeling of a given parameter distribution. To generate random samples from a user defined dis-

tribution we use the function from Adam Nieslony[1] and generate the Weibull mixture with the 10 components summarized in Table C.1b.

Note that Rebmix took more than one day to test the best fitting mixture for the families Normal, Log normal, Weibull and Gamma with a maximum number of components of 20.

Finally, the best fitting mixture is found to be a Weibull mixture. The estimation of the empirical PDF based on histogram leads to the approximation of the original Weibull mixture by only one component with a scale factor of 1586 and a shape factor of 17.

The estimation of the empirical PDF with kernels led to the approximation of the original Weibull mixture by 14 components, described in Table C.1. Figure C.2 plots the different PDFs estimates and compare them to the original PDF. We can see a good fitting of the 14 components mixture. Comparing the first components of the original mix, Table C.1b, with the first components of the two estimates (histogram and kernel based) we can see that:

- The first component in the original mixture is more important than the rest. This observation can explain why an histogram based method only finds one component whose parameters are very close to this component from the original mix, and is correctly detected by both preprocessing methods
- The use of a kernel to estimate the original PDF leads to better results.
- The estimate of the original distribution using kernels finds more components than the number of Weibull components that generated the original one. Such observation should be kept in mind as it could highlight the risk of over fitting the original data when trying to model the distribution it was drawn from using a limited sample size.
- Finally, we can see that the use of parametric mixture to estimate the original data, even if biased by the fact that the original data was generated using a mixture of distributions known by the Rebmix tool, gives a more detailed model of the PDF when compared to the normal kernel estimates. As noticed in the emulated network study, Section 5.1, kernel estimates tend to provide smoother models that can impact the accuracy of the model.

C.2.3 Appraisal

This study measured the consistency in evaluating a given distribution from limited observations. We validated the use of the Rebmix package to model

[1] http://www.mathworks.com/matlabcentral/fileexchange/26003-random-numbers-from-a-user-defined-distribution/content/randpdf.m

C	ω_c	Θ_{1_c}	Θ_{2_c}
1	0.75	1.5e+03	16
2	0.12	1.6e+03	59
3	0.061	1.6e+03	96
4	0.03	1.7e+03	1.1e+02
5	0.013	1.5e+03	2e+02
6	0.012	1.6e+03	2e+02
7	0.003	1.7e+03	8.1e+02
8	0.0037	1.7e+03	5.1e+02
9	0.0031	1.7e+03	5.4e+02
10	0.0035	1.5e+03	3.8e+02
11	0.0031	1.8e+03	4e+02
12	0.0022	1.5e+03	4.6e+02
13	5.5e-05	1.6e+03	1.6e+04
14	0.0055	1.4e+03	63

Table C.1: Rebmix Weibull 14 components for estimating the 10 components Weibull mixture

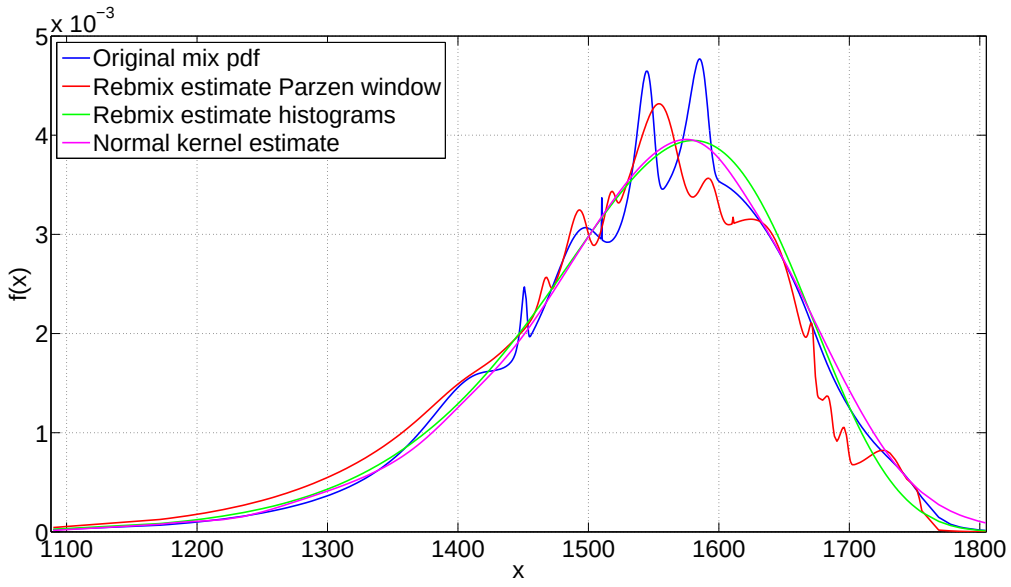


Figure C.2: Comparison of the different estimates

a given distribution using a mixture of distributions drawn from a given family of distribution (Normal, Log normal, Weibull or Gamma). The results we can obtain are limited, as exposed in Section C.2.1, by i) the absence of ground truth in the first study ii) the artificial distribution that we generate in the second study.

C.3 Parametric models and prediction

In this section, we focus on the problematic we are interested in, the modeling of our system parameters distribution to extrapolate the PDFs of the parameters of our models. In this study, we try to model the distribution of the throughput of the CDN clients observed in Chapter 6. The reason why we chose this throughput comes from the fact that the DNS study was the one where we had the bigger number of samples.

C.3.1 Modeling the throughput of the CDN users

In this first study, we just compare the estimation of the throughput distribution using three different methods

- Histograms: Using the Friedman Diaconi rule to choose the bin with, h ($h = 2 * \frac{IQR(x)}{n^{1/3}}$, n being the sample size)
- Normal kernels
- The Rebmix package to model the distribution with a mixture of distribution from the following families: Normal, Log normal, Weibull or Gamma

The results are presented in Figure C.3. To improve the visibility the histogram was rescaled. The model estimated by Rebmix is a mixture of 6 Weibull distributions. We can make the same observations that were made in the previous study. The mixture model and kernel model are showing a high similarity and the kernel model presents a smoother distribution. Here, however, we do not have a ground truth to compare the measure the accuracy of the different models we obtain. We then compare the estimated distributions with the histogram. Despite the very high peak present in the mixture model inferred by the Rebmix package, we can see the estimations obtained using the kernel methods and the mixture model offer acceptable accuracy.

As we are usually interested in the post intervention throughput expected value, we compare the two expected throughput obtained with the kernel model and the mixture model. The expected value estimated by the kernel model is 3.45 Mbps against 3.66 (+6%) when using the mixture model. Here again, the results we obtain show an encouraging similarity.

C.4 Inferring the CDN throughput distribution with missing data

In this second study we remove 10%, 20% and 30% of the samples from the original dataset and use Rebmix to estimate the best fitting mixture model that

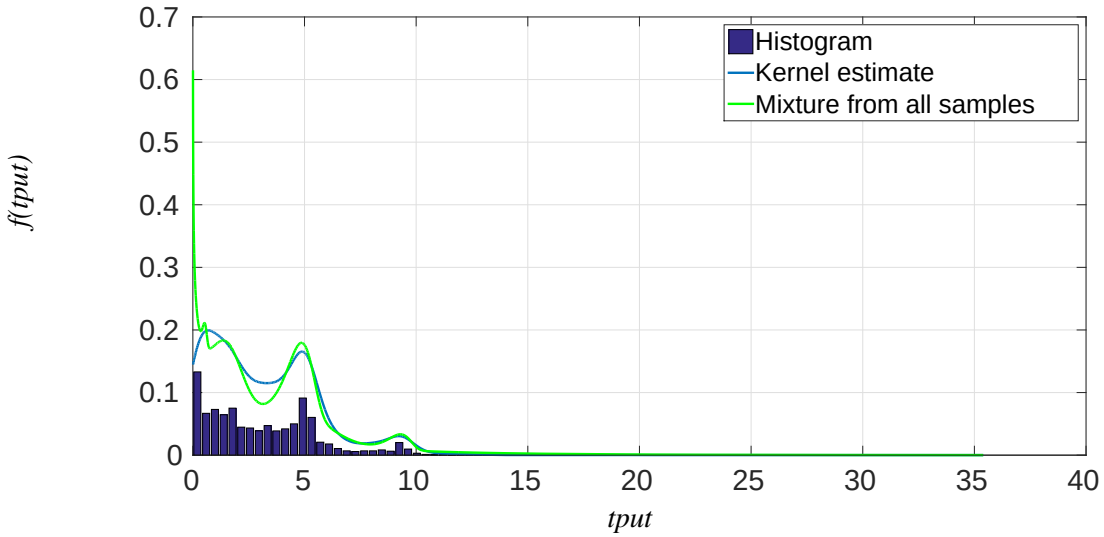


Figure C.3: Estimations of the throughput distribution using three different approaches

can be compared to the one estimated using the same method when all the samples are observed. To remove 10%, 20% and 30% form the sample we select the samples falling in the intervals $[P_5, P_{95}]$, $[P_{10}, P_{90}]$ and $[P_{15}, P_{85}]$ respectively (where P_X represent the X^{th} percentile).

Removing 10%: If we use the 90% of the samples to estimate the throughput distribution, the best fitting model we obtain is a mixture of 9 normal distributions. Figure C.4, we compare the components of the estimations of the throughput distribution with Normal distributions using 100% of the samples vs 90% of the samples. In Figure C.5 compares the components of the estimations of the throughput distribution with Weibull distributions using 100% of the samples vs 90% of the samples. Figure C.6 presents the probability distribution function estimate of the throughput when 90% of the samples are used and Weibull or Normal mixtures are used. If using Normal distribution mixtures evaluated on the 100% of the samples we obtain an expected throughput of 2.79 Mbps (-20%). If we force Rebmix to model the throughput distribution with a Weibull distribution mixture, using 90% of the samples, and estimate the expected throughput, using this model on 100% of the samples, we obtain an expected throughput of 3.18 (-8%).

To clarify the method, we estimated the parameters of the Normal mixture and Weibull mixture using both 100% of the samples and 90% of the samples. We then use these four models to obtain the probability of the 100% of the samples and estimate the expected value. Such approach aims to measure the accuracy of the model inferred with 90% of the samples to estimate the probabilities

of the 10% remaining samples.

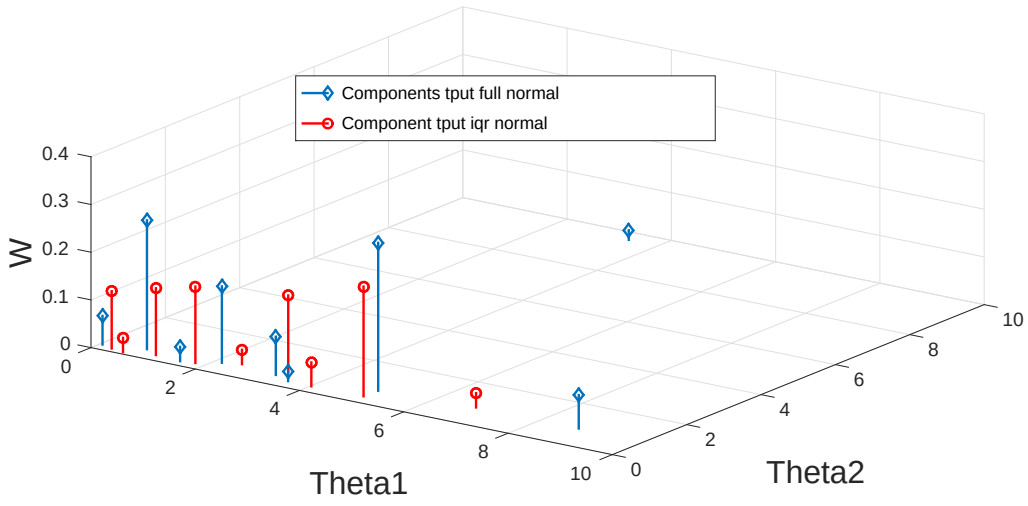


Figure C.4: Comparison of the Normal components estimated from 100% of the samples and the Normal components estimated from 90% of the samples (IQR)

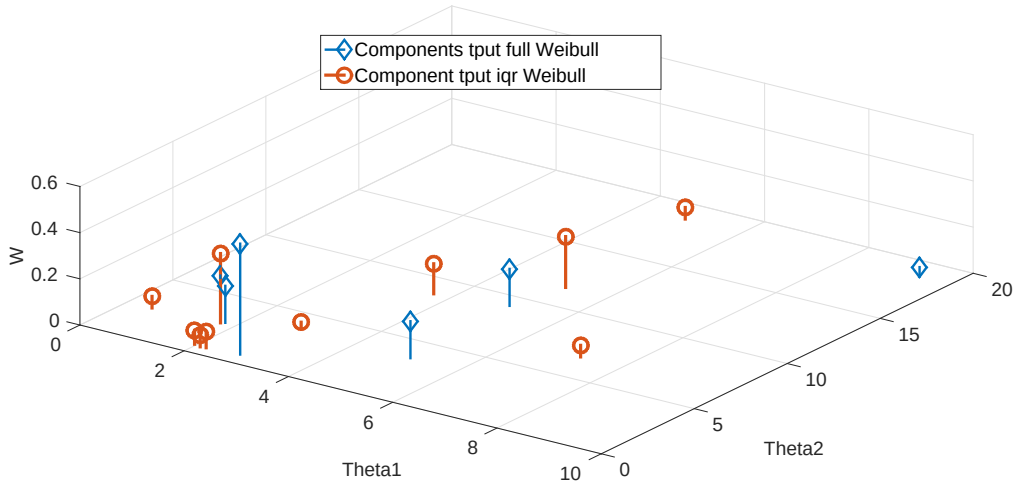


Figure C.5: Comparison of the Weibull components estimated from 100% of the samples and the Weibull components estimated from 90% of the samples (IQR)

Removing 20%: The same operation is repeated removing 20% of the samples. Unexpectedly, Rebmix finds that the best mixture fitting the distribution of the 80% of samples is the product of 6 Weibull distributions. The comparison of the 6 Weibull components estimated from 100% of the samples and the 6

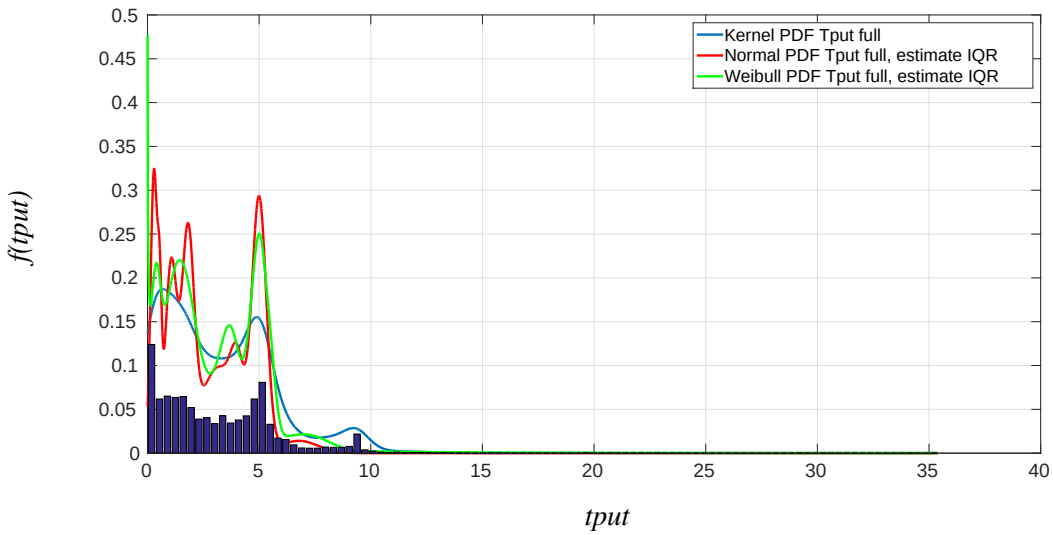


Figure C.6: Comparison of the Normal components estimated from 100% of the samples and the Normal components estimated from 90% of the samples (IQR)

Weibull components estimated from the 80% of the samples is presented in Figure C.7, while the probability density functions, evaluated on 100% of the samples, are presented in Figure C.8. As it was less visible on the previous we did not add the information of the P_5 and P_{95}. On Figure C.8, two vertical lines are placed, that delimit the interval of the samples from which the model with missing data was inferred. We can observe that the estimation made with 80% of the samples is of no use to estimate the probability density of the 20% remaining samples. The expected throughput we obtain is of 2.9 Mbps (-15%).

Removing 30%: The same operation is repeated removing 30% of the samples. Unexpectedly, Rebmix finds that the best mixture fitting the distribution of the 70% of samples is the product of 9 Normal distributions, instead of Weibull distributions as found previously. However, when forcing Rebmix to use Weibull distributions to model the 70% of the throughput samples, it finds 4 components, represented in Figure C.9, and infer a very poor model of the original PDF„ Figure C.10. With such model, we obtain an expected throughput of 7.3 Mbps (+110%).

Instead, when using the product of Normal distributions, the PDF model we obtain seems more accurate, see Figure C.11, while of no use to obtain the probabilities of the remaining 30% of the samples. The expected throughput we obtain, with the Normal mixture, is of 2.9Mbps (-15%) which is better than the results we obtain using Weibull distributions but with an accuracy that is too low to be used in practice.

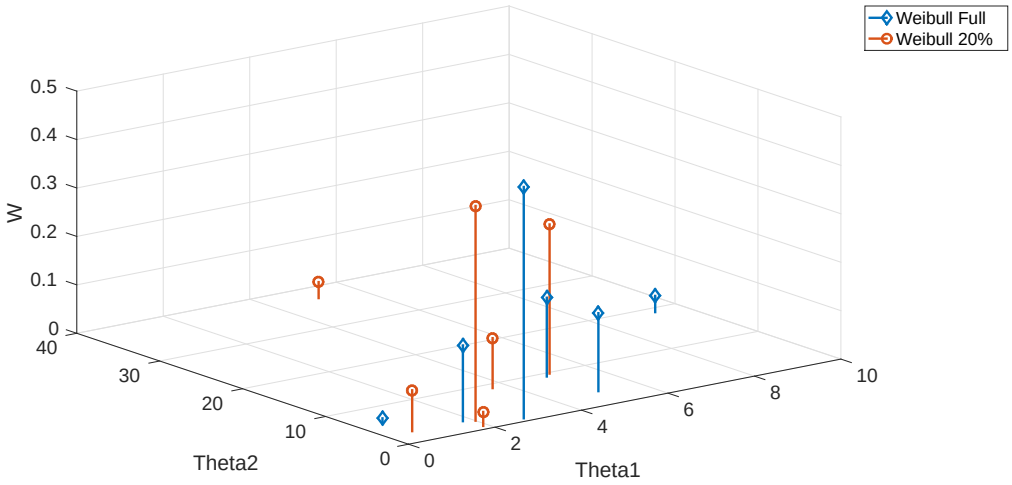


Figure C.7: Comparison of the Weibull components estimated from 100% of the samples and the Weibull components estimated from 80% of the samples (IQR)

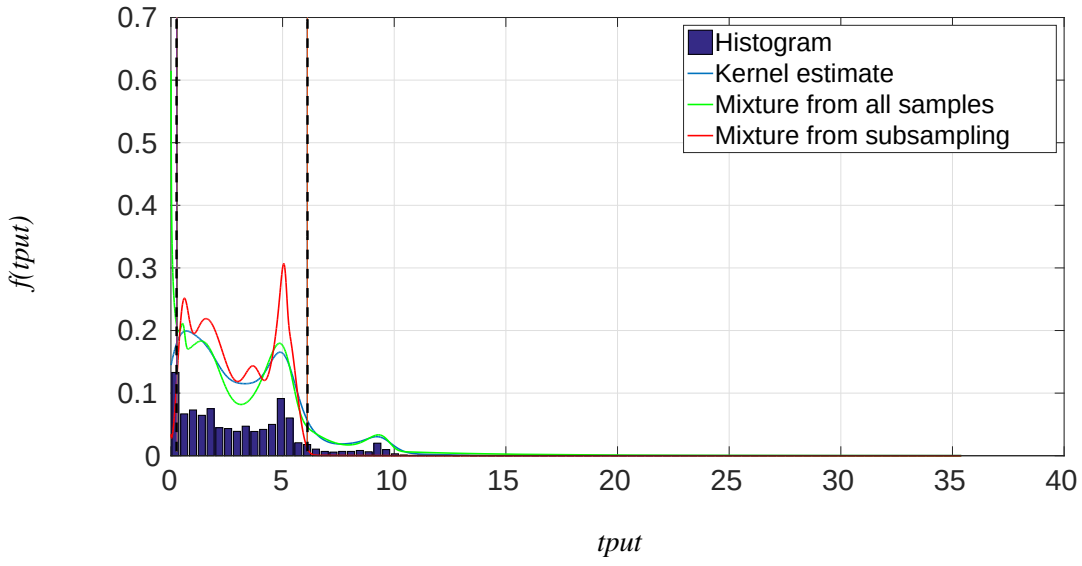


Figure C.8: Comparison of the PDF estimated from 100% of the samples and the PDF estimated from 80% of the samples (IQR)

C.5 Concluding remarks

Rebmix seems to be an interesting tool to obtain a parameterized model of the marginals of our parameters. The point being to be able to use this model in the Back door adjustment formula (Equation (C.7)) and overcome the limitations

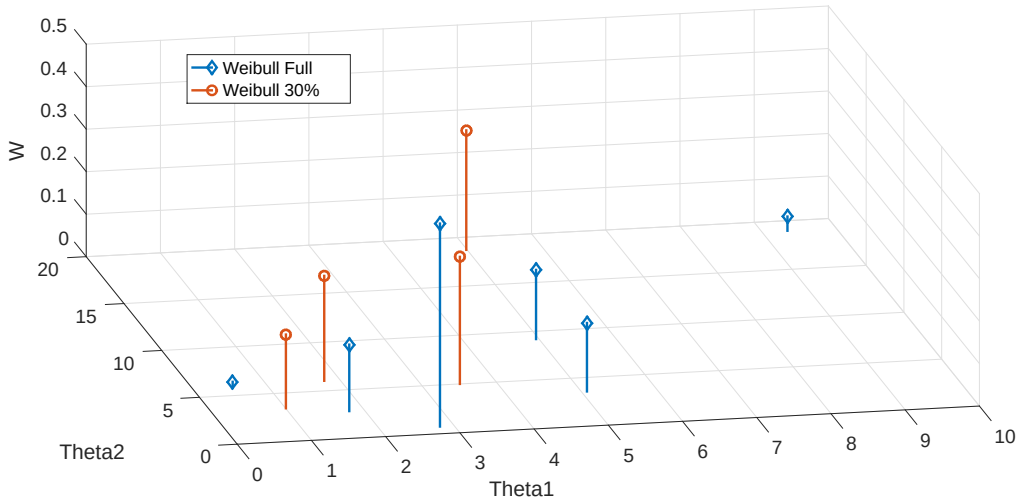


Figure C.9: Comparison of the Weibull components estimated from 100% of the samples and the Weibull components estimated from 70% of the samples (IQR)

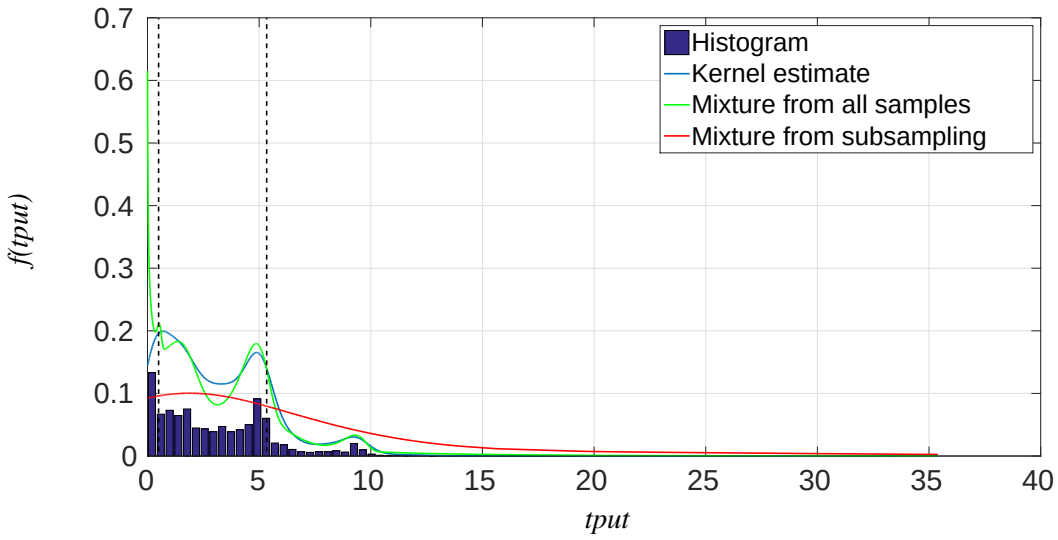


Figure C.10: Comparison of the PDF estimated from 100% of the samples and Weibull components and the PDF estimated from 70% of the samples using Weibull components (IQR)

of the missing data. We could see that, in the case of the *Throughput*, the finite mixture of Weibull components was correctly modeling the marginal in the domain where the throughput was observed but it could not be used as an

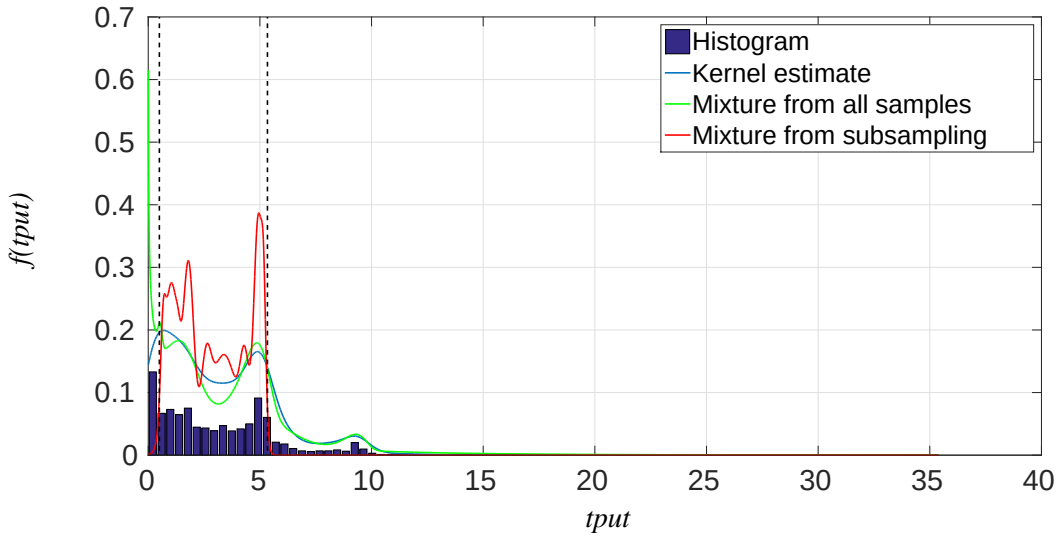


Figure C.11: Comparison of the PDF estimated from 100% of the samples and Weibull components and the PDF estimated from 70% of the samples using Normal components (IQR)

extrapolation to learn the probabilities of unobserved samples.

$$Pr(Y = y|do(X = x)) = \sum_{Z} Pr(Y = y \,|\, X = x,\, Z = z)Pr(Z = z) \tag{C.7}$$

Additionally, we could see that, rapidly, the data shortage impacts greatly the accuracy of the approximation, preventing the use of such models in our work.

The study presented here is a preliminary study. To be more accurate, we should also study the impact of removing values in different zones of the distribution (instead of removing from the minimum and maximum values). In addition, as we remove values from the tails, the comparison of the expected value might under estimate the loss, in terms of accuracy, that the mixture model implies. This scenario is not unrealistic and is similar to the one we had to face in the study of the DNS impact on CDN user performance, when assessing the impact of the minimum congestion window on the throughput, Section 6.4.2.

Some limitations inherent to our approach and the use of the Rebmix package should be noticed. The first one being the limitations of the algorithm which only considers (i) A limited set of distributions (ii) A mixture of components from the same distribution. While the first point might be a limitation of little importance if, with these classes of distributions, we manage to correctly model our marginal, the second point is more important. It could be interesting to have a mixture of components from different families of distributions. To do so a possibility

would be to explain, with the best accuracy possible, the marginal with only one component and *subtract* from the original data the part explained by such component and then repeat the process on the residual. The operation could be repeated as many times as necessary until reaching an acceptable accuracy. In addition, instead of limiting ourselves to given distribution, we could use any parametric function. The risk here being to obtain a model that over fits our data. In which case the use of cross validation is mandatory if the model is to learn the probabilities of unobserved values.

Finally, it should be noticed that Rebmix is computationally demanding. To obtain the best components for modeling 3 parameters, testing with two different criteria and for 1 to 20 components of the 4 families of distributions (*Normal*, *Lognormal*, *Weibull* and *Gamma*), it took several hours (Intel (R) Core i7 CPU Q 720 @1.6GHz, 4GB memory).

Appendix D

Survey on Copulae

In this chapter, we give a very brief overview of the Copula theory and its use in our work. We present the two families of Copulae used in our work, namely the Gaussian copulae et the T-copulae.

D.1 Introduction

A copula is a multivariate probability distribution for which the marginals are uniformly distributed. Copulae are used to model the dependencies between random variables. Copulae are commonly used for modeling high dimensional probability distributions using the Sklar Theorem that states that any multivariate joint distribution can be estimated from its marginals and a copula that captures the dependencies between the different marginals.

D.1.1 Probability integral transform

If $(X_1, X_2, \ldots, X_n)$ is a n-dimensional vector where the marginals X_i are continuous variables and $F_i(X_i)$ are their corresponding cumulative distribution functions (CDFs). Applying the probability integral transform the random vector $(F_1(X_1), F_2(X_2), \ldots, F_n(X_n))$ has uniformly distributed marginals.

The copula of $(X_1, X_2, \ldots, X_n)$ is defined as the joint cumulative distribution function of $(F_1(X_1), F_2(X_2), \ldots, F_n(X_n))$:

$$C(u_1, u_2, \ldots, u_n) = P(F_1(X_1) \leq u_1, F_2(X_2) \leq u_2, \ldots, F_n(X_n) \leq u_n). \quad \text{(D.1)}$$

In Equation D.1, C captures all the dependencies between the components $(X_1, X_2, \ldots, X_n)$ while the marginal cumulative distribution functions, F_i, model the distributions of the marginals.

D.1.2 From modeling marginal dependencies to modeling multivariate distribution

Figure D.1 gives an intuition on how a copula is used to model a bivariate distribution of the vector (X, Y). We first model the dependencies between the components $U = F_X(X)$ and $V = F_Y(Y)$ with a (Gaussian in this example) copula that is parameterized by maximizing the maximum likelihood for a given parameterization of the copula and the input (F_X, F_Y). From the distribution of (U, V) we then deduce the distribution of (X, Y) using the invert cumulative distribution functions,F_X^{-1} and F_Y^{-1}.

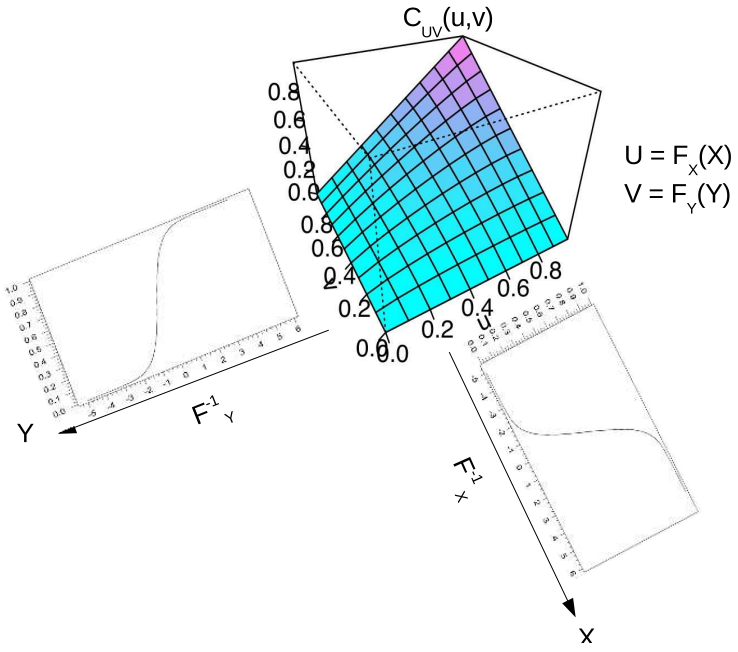


Figure D.1: From copula modeling to multivariate distribution

D.2 Copulae

D.2.1 Definition

A copula is defined as a joint cumulative distribution function of a d-dimensional vector defined on the hypercube $[0, 1]^d$ where its marginals are uniform.

More formally a copula, C is defined as a function

$$\begin{aligned} C : [0,1]^d &\rightarrow [0,1] \\ : U_1, \ldots, U_d &\rightarrow C(U_1, \ldots, U_d), \end{aligned} \tag{D.2}$$

that verifies

- $C(u_1, \ldots, c_{i-1}, 0, c_{i+1}, \ldots, c_d) = 0$, the copula is zero if one of its argument is zero.
- $C(1, \ldots, 1, u, 1, \ldots, 1) = u$, the copula is equal to u if all its arguments are equal to 1 apart from one equals to u.
- C is d-increasing, for each hyper-rectangle $B = \prod_{i=1}^{d} [x_i, y_i] \subseteq [0,1]^d$ the C-volume of B is non negative ($\int_B dC > 0$).

D.2.2 Sklar theorem

Theorem 4. *Sklar Theorem If H is the multivariate cumulative distribution of the vector $(X_1, \ldots, X_d)$ with*

$$H(x_1, \ldots, x_d) = \mathbb{P}(X_1 \leq x_1, \ldots, X_d \leq x_d), \tag{D.3}$$

H can be expressed from its marginals only as:

$$H(x_1, \ldots, x_d) = C(F_1(x_1), \ldots, F_d(x_d)), \tag{D.4}$$

where C is a copula.

In case the multivariate probability density function, h, is available we further have:

$$h(x_1, \ldots, x_d) = c(F_1(x_1), \ldots, F_d(x_d)) \cdot f_1(x_1) \cdot, \ldots \cdot f_d(x_d), \tag{D.5}$$

where c is the density of the copula.

This theorem also states that, if the marginals F_i are continuous then the copula C is unique. However, in our case we try to fit a copula from a given family (T-copulae and Gaussian copulae are the two families used in our work, see Section D.2.3) to better capture and model the dependencies between the marginals of the multivariate distributions we need to estimate.

D.2.3 Copula families

In this section we describe the two families of copulae that we use in our work, namely Gaussian copulae and T-copulae.

Gaussian copula family

The Gaussian copula [XKS00, PCK06] is built from a multivariate normal distribution over $\mathbb{R}^d$ by using the probability integral transform. Based on a correlation matrix R ($R_{ij} = corr(X_i, X_j)$) the Gaussian copula is defined as:

$$C_R^{Gauss}(u) = \Phi_R(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_d)), \tag{D.6}$$

where Φ^{-1} represents the inverse cumulative distribution function of a standard normal distribution and Φ_R is the CDF of a multivariate normal distribution with covariance matrix equal to R.

$$\Phi(x) = \frac{1}{\sqrt{2\Pi}} \int_{-\inf}^{x} e^{-\frac{t^2}{2}} dt \tag{D.7}$$

The density function can be written as

$$C_R^{Gauss}(u) = \frac{1}{\sqrt{detR}} exp\left(\frac{1}{2}\mathbf{\Phi^{-1}}(\boldsymbol{u})^T(R^{-1} - \mathbb{I})\mathbf{\Phi^{-1}}(\boldsymbol{u})\right) \tag{D.8}$$

An example of a bivariate Gaussian copula with a correlation of 0.4 is presented in Figure D.2

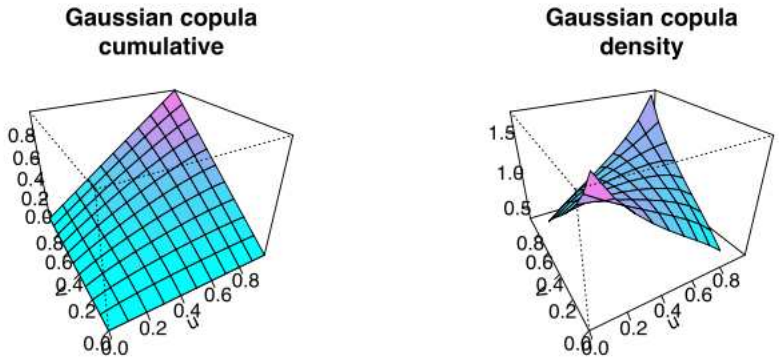


Figure D.2: Example of a bivariate Gaussian copula with $\rho = 0.4$

T-copulae

Please refer to [SA05] for deeper details, in this section we just summarize some of the main ideas from this work.

Multi-variate t-distribution The d-dimensional random vector $\boldsymbol{X} = (X_1, ..., X_d)'$ is said to have a (non-singular) multivariate t-distribution with ν degrees of freedom, mean vector μ and positive-definite dispersion or scatter matrix Σ, denoted $X \sim t_d(\nu, \mu, \Sigma)$, if its density is given by:

$$f(x) = \frac{\Gamma(\frac{\nu+d}{2})}{\Gamma(\frac{\nu}{2})\sqrt{(\Pi\nu)^d|\Sigma|}}\left(1 + \frac{(x-\mu)'\Sigma^{-1}(x-\mu)}{\nu}\right)^{-\frac{\nu+d}{2}} \tag{D.9}$$

Copula It is important to notice that a copula remains invariant under standardization of its marginal distributions. This means that the copula, $t_d(\nu, \mu, \Sigma)$ is identical to that of $t_d(\nu, 0, P)$, where P is the correlation matrix implied by the correlation matrix Σ. The unique copula is then given by:

$$C^t_{\nu,P}(\boldsymbol{u}) = \int_{-\inf}^{t_\nu^{-1}(u_1)} \cdots \int_{-\inf}^{t_\nu^{-1}(u_d)} \frac{\Gamma(\frac{\nu+d}{2})}{\Gamma(\frac{\nu}{2})\sqrt{(\Pi\nu)^d|\Sigma|}}\left(1 + \frac{\boldsymbol{x}'P^{-1}\boldsymbol{x}}{\nu}\right)^{-\frac{\nu+d}{2}} dx, \tag{D.10}$$

where t_ν^{-1} denotes the quantile function of a standard univariate t_ν distribution.

An interesting advantage of the T-copula comes from its ability to capture tail dependencies between the different components of the multivariate distribution.

One result from [SA05] shows that the tail dependency coefficient is given by

$$\lambda = 2t_{\nu+1}(-\sqrt{\nu+1}\sqrt{1-\rho}/\sqrt{1+\rho}), \tag{D.11}$$

where ρ is the off-diagonal element of P.

This result shows that, by tuning the different parameters of the T-copula we can better capture the tail dependencies between the different components of the multi-variate distribution you want to model. Such property is really interesting when modeling telecommunication networks performance, namely the throughput, as it is often the case that we find dependencies in tail of parameter distributions, such as the one of delay or loss, with the throughput. One limitation of the t-copula model comes from its symmetry in the tail dependencies. However, it is possible to extend the model of t-copula to create asymmetry [bar81] and some works have been made to control the tail dependencies between the different component [DSA03] that makes the use of t-copula family very flexible. An illustration of different configurations of the modeling of a bivariate distribution using the family of t-copula is presented in Figure D.3.

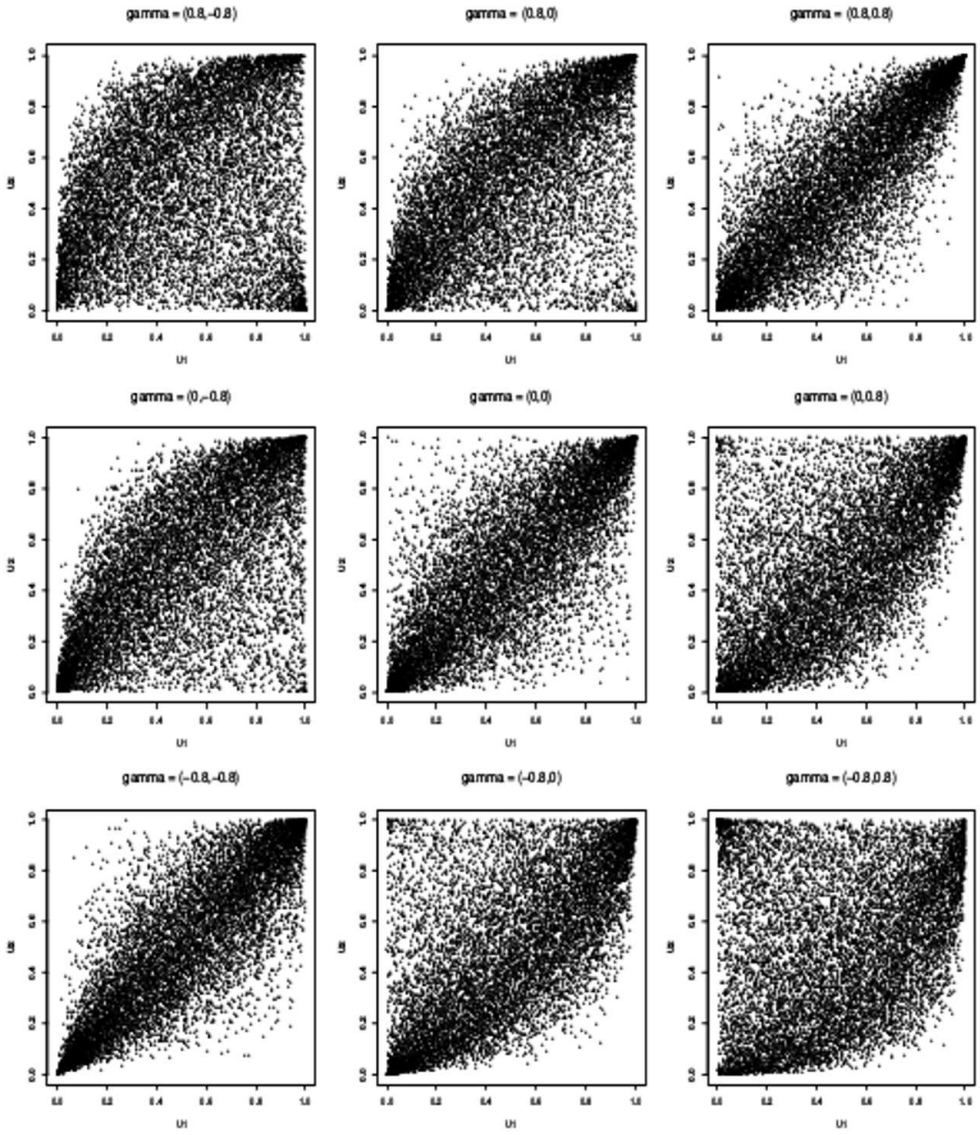


Figure D.3: 10000 simulated points from the bivariate skewed t copula $C^t_{\nu,\rho,\gamma_1,\gamma_2}$ for $\nu = 5$, $\rho = 0.8$ and various values of the parameters (γ_1, γ_2)

Appendix E

Survey on independence tests

In this chapter, we give a very brief overview of independence testing before focusing on the KCI and the HSIC. We then compare these two indpendence tests and expose the reasons why we chose the KCI test in our work.

E.1 Existing works

E.1.1 Estimation of distributions

[SW08] measures the distance between the conditional probability, $p(X|Y,Z)$ and the conditional probability, $p(X|Y)$. [SW07] exploits the characteristic functions of these two distributions. However, estimation of these two distributions is rather complex, which deteriorates the accuracy of the test, above all when Z is of important dimension.

E.1.2 Discretization

[Mar05, Hua10] discretize Z in a set of bins and transforms conditional independence test into unconditional test. Because of the curse of dimensionality this approach cannot scale up with the size of the conditioning set Z.

E.1.3 Find functions separating X and Y

[Son09] proposes a method where one tries to find a function h and a parameter Θ_0 such that X and Y are conditionally independent given a single index function $\lambda_{\Theta_0}(Z)$= $h(Z^T\Theta_0)$ of Z. This is different from testing independence. One counter example given in [ZPJS12] is to take X and Y conditionally dependent on two intersecting but non overlapping subsets of Z.

E.2 HSIC

E.2.1 Description

This section summarizes some of the concepts presented in [GFT+08].

Let P_{xy} be a (Borel) probability measure defined on a domain $\mathcal{X} \times \mathcal{Y}$ and let P_x and P_y be the respective marginal distributions on $\mathcal{X}$ and $\mathcal{Y}$.

Let $\mathcal{F}$ be a Reproducive Kernel Hilbert Space (RKHS) with the continuous feature mapping $\phi(x) \in \mathcal{F}$ from each $x \in \mathcal{X}$, such that the inner product between the features is given by the kernel function $k(x, x') =< \phi(x), \phi(x') >$. Likewise, let $\mathcal{G}$ be a second RKHS on $\mathcal{Y}$ with kernel $l(.,.)$ and feature map $\psi(y)$. From [FBJ04], the cross covariance operator $C_{xy} : \mathcal{G} \to \mathcal{F}$ is defined such that for all $f \in \mathcal{F}$ and $g \in \mathcal{G}$ we have

$$< f, C_{xy} g >_{\mathcal{F}} = \mathbb{E}_{xy} \left([f(x) - \mathbb{E}_x(f(x))] \left[g(y) - \mathbb{E}_y(g(y)) \right] \right). \tag{E.1}$$

The cross-covariance operator itself can then be written:

$$C_{xy} = \mathbb{E}_{xy} \left[(\phi(x) - \mu_x) \otimes \left(\psi(y) - \mu_y \right) \right], \tag{E.2}$$

where $\mu_x := \mathbb{E}_x[\phi(x)]$, $\mu_y := \mathbb{E}_y[\psi(y)]$ and $\otimes$ is the tensor product [GBSS05, Eq. 6], the generalization of the cross-covariance matrix between random vectors. When $\mathcal{F}$ and $\mathcal{G}$ are universal RKHS (that is dense in the space of bounded continuous functions) on the compact domains $\mathcal{X}$ and $\mathcal{Y}$, then ***the largest singular value of this operator is zero if and only if*** $\boldsymbol{X \perp\!\!\!\perp Y}$ [GHS+05]. The maximum singular value gives a criterion to measure independence. However, instead of using the maximum singular value, the squared Hilbert-Schmidt norm can be used instead. Its population expression is given by:

$$HSIC_b = \mathbb{E}_{xx'yy'}[k(x,x')l(y,y')] + \mathbb{E}_{xx'}[k(x,x')]\mathbb{E}_{yy'}[l(y,y')] - 2\mathbb{E}_{xy}[\mathbb{E}_{x'}[k(x,x')]\mathbb{E}_{y'}[l(y,y')]], \tag{E.3}$$

where x' denotes an independent copy of x. We call this criterion the Hilbert Schmidt Independence criterion.

To estimate $HSIC(P_{xy}, \mathcal{F}, \mathcal{G})$ from the sample $\boldsymbol{Z} = (\boldsymbol{X}, \boldsymbol{Y}) = \{(x_1, y_1), \ldots, (x_m, y_m)\})$, a (biased) empirical estimate is given by

$$HSIC_b(\mathbf{Z}) = \frac{1}{m^2} \sum_{i,j}^{m} k_{ij} l_{ij} + \frac{1}{m^4} \sum_{i,j,q,r}^{m} k_{ij} l_{qr} - 2 \frac{1}{m^3} \sum_{i,j,q}^{m} k_{ij} l_{iq} = \frac{1}{m^2} Tr(\boldsymbol{KHLH}), \tag{E.4}$$

where $k_{ij} = k(x_i, x_j)$ and $l_{ij} = l(y_i, y_j)$ and the indexes denote all the r-tuples drawn with replacement from $\{1, \ldots, m\}$. $\boldsymbol{K}$ is the $m \times m$ matrix with entries k_{ij}, $\boldsymbol{L}$ the matrix with entries l_{ij} and $\boldsymbol{H} = \mathbb{I} - \frac{1}{m} \mathbf{1}\mathbf{1}^T$. When a Gaussian kernel is used we have $k_{ij} = exp(-\sigma^2 \|x_i - x_j\|^2)$.

E.2.2 Test

The test aims at distinguishing between the null hypothesis $\mathcal{H}_0 : X \perp\!\!\!\perp Y$ and the the alternative hypothesis $\mathcal{H}_1 : X \not\perp\!\!\!\perp Y$. To do so, the test compare the test statistics $HSIC_b(\mathbf{Z})$ with a particular threshold, if the statistics is above the threshold, $\mathcal{H}_0$ is rejected, otherwise it is accepted. The acceptance region is defined as a real value below this threshold. To define the threshold, [GFT+08] derives an asymptotic distribution of the empirical estimate $HSIC_b(\mathbf{Z})$ of $HSIC_b(P_{xy}, \mathcal{F}, \mathcal{G})$ under $\mathcal{H}_0$ and use the $1-\alpha$ quantile of this distribution as the test threshold.

Distribution of $HSIC_b(\mathbf{Z})$

Under $\mathcal{H}_1$: With

$$h_{ijqr} = \frac{1}{4!} \sum_{t,u,v,w}^{i,j,q,r} k_{tu} l_{tu} + k_{tu} l_{vw} - 2 k_{tu} k_{tv}, \tag{E.5}$$

where the sum represents all ordered quadruples (t,u,v,w) drawn without replacement from (i,j,q,r), the authors prove that, under $\mathcal{H}_1$:

$$m^{\frac{1}{2}} \left(HSIC_b(\mathbf{Z}) - HSIC(P_{xy}, \mathcal{F}, \mathcal{G}) \right) \xrightarrow{D} \mathcal{N}(0, \sigma_u^2), \tag{E.6}$$

with $\mathcal{N}(0, \sigma^2)$ representing a normal distribution and σ_u^2 defined by:

$$\sigma_u^2 = 16 \left(\mathbb{E}_i \left[\mathbb{E}_{j,q,r} [h_{ijqr}] \right]^2 - HSIC(P_{xy}, \mathcal{F}, \mathcal{G}) \right), \tag{E.7}$$

where $\mathbb{E}_{j,q,r} := \mathbb{E}_{z_j, z_q, z_r}$.

Under $\mathcal{H}_0$:

$$mHSICC_b \xrightarrow{D} \sum_{l=1}^{\infty} \lambda_l z_l^2 \tag{E.8}$$

where $z_l \sim \mathcal{N}(0,1)$ i.i.d. and λ_l are the solution to the eigen-value problem

$$\lambda_l \psi_l(z_j) = \int h_{ijqr} \psi_l(z_i) dF_{i,q,r}, \tag{E.9}$$

where the integral is over the distributions of z_i, z_q and z_r

Approximating the $1-\alpha$ quantile of the null distribution: From the distribution of Equation E.8 we can estimate the $(1-\alpha)$ quantile. The consistency of the test (convergence to 0 of the type II error when $m \to \infty$) is ensured by the decays as m^{-1} of the variance of $HSIC_b$ under $\mathcal{H}_1$. However, the distribution under $\mathcal{H}_0$ is more complex and some approximations are necessary to obtain the quantiles.

To do so the authors approximate the distribution of $HSIC_b$ under $\mathcal{H}_0$ with a two parameter Gamma distribution as follows:

$$mHSIC_b(\mathbf{Z}) \sim \frac{x^{\alpha-1}e^{-x/\beta}}{\beta^{\alpha}\Gamma(\alpha)} \tag{E.10}$$

where $\alpha = \frac{(\mathbb{E}[HSIC_b(\mathbf{Z})])^2}{var(HSIC_b(\mathbf{Z}))}$ and $\beta = \frac{mvar(HSIC_b(\mathbf{Z}))}{\mathbb{E}[HSIC_b(\mathbf{Z})]}$

Such approximations still require the estimation of $\mathbb{E}\left[HSIC_b(\mathbf{Z})\right]$ and $var(HSIC_b(\mathbf{Z}))$. To do so, the authors the following two results, valid under $\mathcal{H}_0$.

For $\mathbb{E}[HSIC_b(\mathbf{Z})]$:

$$\mathbb{E}[HSIC_b(\mathbf{Z})] = \frac{1}{m}Tr(C_{xx})Tr(C_{yy}) = \frac{1}{m}\left(1 + \|\mu_x\|^2\|\mu_y\|^2 - \|\mu_x\|^2 - \|\mu_y\|^2\right), \tag{E.11}$$

where the second equality assumes $k_{ii} = l_{ii} = 1$ and reminding that the covariance operator are defined as:

$$C_{xx} := \mathbb{E}\left[(\phi(x) - \mu_x) \otimes (\phi(x) - \mu_x)\right] \tag{E.12}$$

An empirical estimate of the statistics defined in Equation E.11 is obtained by replacing the norms with $\widehat{\|\mu_x\|^2} = (m)_2^{-1}\sum_{i,j\in i_2^m} k_{ij}$. It is important to notice that this results in a bias of $O(m^{-1})$ in the estimate of $\|\mu_x\|^2\|\mu_y\|^2$. The following notation is used:

- $(m)_n := \frac{m!}{(m-n)!}$
- $\boldsymbol{i}_r^m$ denotes the set of all r-tuples drawn without replacement from the set $\{1,\ldots,m\}$.

For $var(HSIC_b(\mathbf{Z}))$:

$$var(HSIC_b(\mathbf{Z})) = \frac{2(m-4)(m-5)}{(m)_4}\|C_{xx}\|_{HS}^2\|C_{yy}\|_{HS}^2 + O(m^{-3}) \tag{E.13}$$

If we use the symbol $\odot$ the entry-wise matrix product, $\boldsymbol{A}^{\cdot 2}$ the entry wise matrix power, and $\boldsymbol{B} = ((\boldsymbol{HKH}) \odot (\boldsymbol{HLH}))^{\cdot 2}$, we can obtain negligible bias estimate of EquationE.13 replacing the product of covariance norms with $\mathbf{1}^{\boldsymbol{T}}(\boldsymbol{B} - \boldsymbol{diag}(\boldsymbol{B}))\mathbf{1}$ which is more efficient than taking the product of the empirical norms.

E.3 KCI

This section summarizes some of the concepts presented in [ZPJS12].

E.3.1 Notations

X, Y and Z are continuous parameters with domains $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$. By denoting k_x a measurable, definite, positive kernel on $\mathcal{X}$, we denote $\mathcal{H}_X$ the corresponding Reproducive Kernel Hilbert Space (RKHS). The same way k_y, k_z are defined on $\mathcal{Y}$ and $\mathcal{Z}$ with corresponding RKHS $\mathcal{H}_\mathcal{Y}$ and $\mathcal{H}_\mathcal{Z}$. The probability law of X, Y and Z are denoted P_X, P_Y and P_Z respectively and the joint probabilities are denoted P_{XZ}.

We denote the space of square integrable functions on X and (X, Z) as L^2_X, L^2_Z and L^2_{XZ} respectively. $\boldsymbol{x} = \{x_1, \ldots, x_n\}$ denotes the i.i.d. sample of X of size n. $\boldsymbol{K}_X$ is the kernel matrix of the sample $\boldsymbol{x}$ and the corresponding centralized matrix is denoted $\widetilde{\boldsymbol{K}}_X \triangleq \boldsymbol{H}\boldsymbol{K}_X\boldsymbol{H}$, with $\boldsymbol{H} = \mathbb{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T$.

For example the (i,j) entry of $\boldsymbol{K}_X$, when the default Gaussian kernel is used is given by:

$$k(x_i, x_j) = exp(-\frac{\|x_i - x_j\|^2}{2\sigma_x^2}) \tag{E.14}$$

E.3.2 Characterisation of independence

Cross covariance operator

For the random vector (X, Y) on $\mathcal{X} \times \mathcal{Y}$, the cross-covariance operator from $\mathcal{H}_X$ to $\mathcal{H}_Y$ is defined by the relation:

$$< g, \Sigma_{YX} f >= \mathbb{E}_{XY}\left[f(x)g(Y)\right] - \mathbb{E}_X\left[f(x)\right]\mathbb{E}_Y\left[g(Y)\right], \tag{E.15}$$

for all $f \in \mathcal{H}_X$ and all $g \in \mathcal{H}_Y$. The conditional cross-covariance operator of (X, Y) given Z is further defined as

$$\Sigma_{YX|Z} = \Sigma_{YX} - \Sigma_{YZ}\Sigma_{ZZ}^{-1}\Sigma_{ZX} \tag{E.16}$$

If characteristic kernels are used[1], the conditional independence related to the cross-covariance operator as follows:

[1] A kernel k_X is said to *characteristic* if $\mathbb{E}_{X\sim P}[f(x)] = \mathbb{E}_{X\sim Q}[f(X)]$ ($\forall f \in \mathcal{H}$ implies $P = Q$

Theorem 5 (Characterization of CI based on the cross covariance operator [fuk08]). *Denote* $\ddot{X} \triangleq (X, Z)$, $k_{\ddot{\mathcal{X}}} \triangleq k_{\mathcal{X}} k_{\mathcal{Z}}$ *and* $\mathcal{H}_{\ddot{\mathcal{X}}}$ *the RKHS corresponding to* $k_{\ddot{\mathcal{X}}}$. *Assume* $\mathcal{H}_{\ddot{\mathcal{X}}} \subset L^2_X$, $\mathcal{H}_{\ddot{\mathcal{Y}}} \subset L^2_Y$ *and* $\mathcal{H}_{\ddot{\mathcal{Z}}} \subset L^2_Z$. *If we also have that* $k_{\ddot{\mathcal{X}}} k_{\mathcal{X}}$ *is a characteristic kernel on* $(\mathcal{X} \times \mathcal{Y}) \times Z$ *and that the direct sum of two RKHS on* $\mathcal{Z}$ *is dense in* $L^2(P_Z)$. *Then*

$$\Sigma_{\ddot{X}Y|Z} = 0 \iff X \perp\!\!\!\perp Y|Z \tag{E.17}$$

Partial association

Additionally, [J.80] gives the characterization of CI using the absence of correlation between functions in a certain spaces. Considering the following spaces:

$$\begin{aligned} \varepsilon_{XZ} &= \{\widetilde{f} \in L^2_{XZ} | \mathbb{E}(\widetilde{f}|Z) = 0\} \\ \varepsilon_{YZ} &= \{\widetilde{g} \in L^2_{YZ} | \mathbb{E}(\widetilde{g}|Z) = 0\} \\ \varepsilon'_{YZ} &= \{\widetilde{g'} | \widetilde{g'} = g'(\boldsymbol{Y}) - \mathbb{E}(g'|Z), g' \in L^2_Y\} \end{aligned} \tag{E.18}$$

They can be constructed from the corresponding L^2 spaces via nonlinear regression. For instance, for any function $f \in L^2_{XZ}$, the corresponding function $\widetilde{f}$ is given by:

$$\widetilde{f}(\ddot{X}) = f(\ddot{X}) - \mathbb{E}(f|Z) = f(\ddot{X}) - h^*_f(Z), \tag{E.19}$$

where $h^*_f(Z) \in L^2_Z$ is the regression function of $f(\ddot{X})$ on Z

Theorem 6. *Characterization of CI based on partial association [J.80] The following conditions are equivalent to each other. (i)* $X \perp\!\!\!\perp Y|Z$; *(ii)* $\mathbb{E}[\widetilde{f}\widetilde{g}] = 0$, $\forall \widetilde{f} \in \varepsilon_{XZ}$ *and* $\widetilde{g} \in \varepsilon_{YZ}$, *(iii)* $\mathbb{E}[\widetilde{f}f] = 0$, $\forall \widetilde{f} \in \varepsilon_{XZ}$ *and* $g \in L^2_{YZ}$, *(iv)* $\mathbb{E}[\widetilde{f}\widetilde{g'}] = 0$, $\forall \widetilde{f} \in \varepsilon_{XZ}$ *and* $\widetilde{g'} \in \varepsilon'_{YZ}$, *(v)* $\mathbb{E}[\widetilde{f}g'] = 0$, $\forall \widetilde{f} \in \varepsilon_{XZ}$ *and* $g \in L^2_Y$,

The advantage of Theorem 5 over Theorem 6 is that it exploits spaces corresponding to some characteristic kernel instead of all functions in L^2.

E.3.3 Kernel based independence test

Let denote the eigenvalue decomposition of the centralized kernel matrices of $\widetilde{K}_X$ and $\widetilde{K}_Y$ as:

$$\widetilde{\boldsymbol{K}}_X = \boldsymbol{V_x} \boldsymbol{\Lambda_x} \boldsymbol{V_x^T} \tag{E.20}$$

and

$$\widetilde{\boldsymbol{K}}_Y = \boldsymbol{V}_{\boldsymbol{y}}\boldsymbol{\Lambda}_{\boldsymbol{y}}\boldsymbol{V}_{\boldsymbol{y}}^{\boldsymbol{T}}, \tag{E.21}$$

where the matrices $\boldsymbol{\Lambda}_{\boldsymbol{x}}$ and $\boldsymbol{\Lambda}_{\boldsymbol{y}}$ are the diagonal matrices containing the non-negative eigen values $\lambda_{x,i}$ and $\lambda_{y,i}$ respectively. Without loss in generality, we will suppose that the eigenvalues are sorted in descending order.

Let $\boldsymbol{\psi}_X = [\psi_1(\boldsymbol{x}), \ldots, \psi_n(\boldsymbol{x})] \triangleq \boldsymbol{V}_{\boldsymbol{x}}{\boldsymbol{\Lambda}_{\boldsymbol{x}}}^{1/2}4$ and $\boldsymbol{\phi}_Y = [\phi_1(\boldsymbol{y}), \ldots, \phi_n(\boldsymbol{y})] \triangleq \boldsymbol{V}_{\boldsymbol{y}}{\boldsymbol{\Lambda}_{\boldsymbol{y}}}^{1/2}4$. I.e. $\psi_i(\boldsymbol{x}) = \sqrt{\lambda_{x,i}}\boldsymbol{V}_{x,i}$, where $\boldsymbol{V}_{x,i}$ denotes the ith eigen vector of $\widetilde{K}_X$.

On the other hand, we denote $\lambda^*_{X,i}$ and the eigenfunctions $u_{X,i}$ of the kernel $k_{\mathcal{X}}$ with respect to the probability measure with density $p(x)$, i.e. we have:

$$\int k_{\mathcal{X}}(x, x')u_{X,i}(x)p(x)dx = \lambda^*_{X,i}u_{X,i}(x'), \tag{E.22}$$

assuming unit variance of $u_{X,i}$ and also supposing that $\lambda^*_{X,i}$ are sorted in descending order. Using the same notations for Y we can define $\lambda^*_{Y,i}$, $u_{Y,i}$ of $k_{\mathcal{Y}}$.

If we define:

$$S_{ij} = \frac{1}{\sqrt{n}}\psi_i(\boldsymbol{x})^T\phi_j(\boldsymbol{y}) = \frac{\sum_{t=1}^n \psi_i(x_t)\phi_j(y_t)}{\sqrt{n}}, \tag{E.23}$$

with $\psi_i(x_t)$ being the t-th component of the vector $\psi_i(\boldsymbol{x})$, we have the following results:

Theorem 7. *Distribution of eigen value under independence Suppose that we are given arbitrary centered kernels $k_{\mathcal{X}}$ and $k_{\mathcal{Y}}$ with discrete eigenvalues and the corresponding RKHS's $\mathcal{H}_{\mathcal{X}}$ and $\mathcal{H}_{\mathcal{Y}}$ for set of random variable X and Y respectively. We have the following three statement*

1. *Under the condition that $f(X)$ and $g(Y)$ are uncorrelated for all $f \in \mathcal{H}_{\mathcal{X}}$ and $g \in \mathcal{H}_{\mathcal{Y}}$, for any L such that $\lambda^*_{X,L+1} \neq \lambda^*_{X,L}$ and $\lambda^*_{Y,L+1} \neq \lambda^*_{y,L}$, we have*

$$\sum_{i,j=1}^{L} S_{ij}^2 \xrightarrow{d} \sum_{k=1}^{L^2} \mathring{\lambda}^*_k z_k^2, \text{ as } n \to \infty, \tag{E.24}$$

   where z_k are i.i.d. standard Gaussian variables, $\mathring{\lambda}^*_k$ are the eigenvalues of $\mathbb{E}[\boldsymbol{w}\boldsymbol{w}^T]$ and $\boldsymbol{w}$ represents the random vector obtained by stacking the $L \times L$ matrix $\boldsymbol{N}$ whose (i, j) entry is $N_{ij} = \sqrt{\lambda^*_{X,i}\lambda^*_{Y,j}}u_{X,i}(X)u_{Y,j}(Y)$

2. *In particular, if $X \perp\!\!\!\perp Y$ we have*

$$\sum_{i,j=1}^{L} S_{ij}^2 \xrightarrow{d} \sum_{k=1}^{L^2} \lambda^*_{X,i}\lambda^*_{Y,j} \cdot z_{ij}^2, \text{ as } n \to \infty, \tag{E.25}$$

   where z_{ij}^2 are χ_1^2-distributed variables

3. The results of Equation (E.24) and Equation (E.25) also hold if $L = n \to \infty$

Noting that:

$$Tr(\widetilde{\boldsymbol{K}}_X \widetilde{\boldsymbol{K}}_Y) = Tr(\psi_X \psi_X^T \phi_Y \phi_Y^T) = Tr(\psi_X^T \phi_Y \phi_Y^T \psi_X) = n \sum_{i,j=1}^{n} S_{ij}^2 \tag{E.26}$$

we have the asymptotic distribution of $\frac{1}{n} Tr(\widetilde{\boldsymbol{K}}_X \widetilde{\boldsymbol{K}}_Y)$ under the condition that $f(X)$ and $g(Y)$ are always uncorrelated.

Note however that the eigenvalues $\mathring{\lambda}_k^*$ are not known so empirical ones need to be used instead.

E.3.4 Unconditional independence testing

Theorem 8. *Unconditional independence test Under the null hypothesis that X and Y are statistically independent, the statistic:*

$$T_{UI} \triangleq \frac{1}{n} Tr(\widetilde{\boldsymbol{K}}_X \widetilde{\boldsymbol{K}}_Y) \tag{E.27}$$

has the same asymptotic distribution as

$$\check{T}_{UI} \triangleq \frac{1}{n^2} \sum_{i,j=1}^{n} \lambda_{\boldsymbol{x},i} \lambda_{\boldsymbol{y},j} z_{ij}^2 \tag{E.28}$$

i.e. $T_{UI} \overset{d}{=} \check{T}_{UI}$ as $n \to \infty$

Given the samples $\boldsymbol{x}$ and $\boldsymbol{y}$, one first calculates the centralized kernel matrices $\widetilde{\boldsymbol{K}}_X$ and $\widetilde{\boldsymbol{K}}_Y$ and their eigenvalues $\lambda_{\boldsymbol{x},i}$ and $\lambda_{\boldsymbol{y},i}$ and then evaluates the statistic T_{UI} according to Equation (E.27). Next the empirical null distribution of $\check{T}_{UI}$ under the null hypothesis can be simulated using Equation (E.28) and drawing random samples from the χ_1^2 distributed variables z_{ij}^2. The p-value can then be obtained by locating T_{UI} in the null distribution.

E.3.5 Conditional independence testing

For the conditional test, the authors use the condition *(iv)* of Theorem 6, but the considered functional spaces are $f(\ddot{X}) \in \mathcal{H}_{\ddot{X}}$, $g'(Y) \in \mathcal{H}_{\boldsymbol{y}}$, $h_f^*(Z) \in \mathcal{H}_{\mathcal{Z}}$ and $h_g'^*(Z) \in \mathcal{H}_{\mathcal{Z}}$, as defined for Theorem 5. The functions $\widetilde{f}(\ddot{X})$ and $g'(Y,Z)$ appearing in condition *(iv)*, whose spaces are denoted by $\mathcal{H}_{\ddot{X}|\mathcal{Z}}$ and $\mathcal{H}_{\boldsymbol{y}|\mathcal{Z}}$, respectively, can be constructed from the functions f, g', h_f^*, and $h_{g'}^*$.

Supposing that we already have the centralized kernel matrices $\widetilde{\boldsymbol{K}}_X$ (the centralized kernel matrix of $\dot{X} = (X, Z)$), $\widetilde{\boldsymbol{K}}_Y$ and $\widetilde{\boldsymbol{K}}_Z$ on the samples $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$. We use kernel ridge regression to estimate the regression function of $h_f^*(Z)$ of Equation (E.19) we have $\widehat{h_f^*(\boldsymbol{z})} = \widetilde{\boldsymbol{K}}_Z(\widetilde{\boldsymbol{K}}_Z + \varepsilon\mathbb{I})^{-1}.f(\ddot{\boldsymbol{x}})$, where ε is a small positive regularization parameter. Consequently, $\widetilde{f}(\ddot{\boldsymbol{x}})$ can be constructed as $\widetilde{f}(\ddot{\boldsymbol{x}}) = f(\ddot{\boldsymbol{x}}) - \widehat{h_f^*(\boldsymbol{z})} = \boldsymbol{R}_Z$ where:

$$\boldsymbol{R}_Z = \mathbb{I} - \widetilde{\boldsymbol{K}}_Z(\widetilde{\boldsymbol{K}}_Z + \varepsilon\mathbb{I})^{-1} = \varepsilon(\widetilde{\boldsymbol{K}}_Z + \varepsilon\mathbb{I})^{-1} \tag{E.29}$$

Based on the eigenvalue decomposition $\widetilde{\boldsymbol{K}}_{\ddot{X}} = \boldsymbol{V}_{\ddot{\boldsymbol{X}}}^{\boldsymbol{T}}\boldsymbol{\lambda}_{\ddot{X}}\boldsymbol{V}_{\ddot{X}}$, we can construct $\boldsymbol{\psi}_{\ddot{\boldsymbol{x}}} = [\psi_1(\ddot{\boldsymbol{x}}), \ldots, \psi_n(\ddot{\boldsymbol{x}})] \triangleq \boldsymbol{V}_{\ddot{\boldsymbol{X}}}\boldsymbol{\lambda}_{\ddot{\boldsymbol{X}}}{}^{1/2}$, as an empirical kernel map for $\ddot{\boldsymbol{x}}$. Correspondingly, an *empirical* kernel map of the space $\mathcal{H}_{\ddot{X}|\mathcal{Z}}$ is given by $\widetilde{\boldsymbol{\phi}}_{\ddot{\boldsymbol{x}}} = \boldsymbol{R}_Z\boldsymbol{\psi}(\ddot{\boldsymbol{x}})$. Consequently, the centralized kernel matrix corresponding to the functions $\widetilde{f}(\ddot{X})$ is

$$\widetilde{\boldsymbol{K}}_{\ddot{X}|Z} = \widetilde{\boldsymbol{\psi}}_{\ddot{\boldsymbol{x}}}\widetilde{\boldsymbol{\psi}}_{\ddot{\boldsymbol{x}}}{}^T = \boldsymbol{R}_Z\widetilde{\boldsymbol{K}}_{\ddot{X}}\boldsymbol{R}_Z. \tag{E.30}$$

Similarly, that corresponding to $\widetilde{g'}$ is

$$\widetilde{\boldsymbol{K}}_{Y|Z} = \boldsymbol{R}_Z\widetilde{\boldsymbol{K}}_Y\boldsymbol{R}_Z. \tag{E.31}$$

Further we let the EVD decompositions of $\widetilde{\boldsymbol{K}}_{\ddot{X}|Z}$ and $\widetilde{\boldsymbol{K}}_{Y|Z}$ be

$$\widetilde{\boldsymbol{K}}_{\ddot{X}|Z} = \mathbb{E}_{\ddot{\boldsymbol{X}}|\boldsymbol{Z}}\boldsymbol{\lambda}_{\ddot{\boldsymbol{X}}|\boldsymbol{Z}}\mathbb{E}_{\ddot{\boldsymbol{X}}|\boldsymbol{Z}}^{\boldsymbol{T}} \tag{E.32}$$

and

$$\widetilde{\boldsymbol{K}}_{Y|Z} = \mathbb{E}_{\boldsymbol{Y}|\boldsymbol{Z}}\boldsymbol{\lambda}_{\boldsymbol{Y}|\boldsymbol{Z}}\mathbb{E}_{\boldsymbol{Y}|\boldsymbol{Z}}^{\boldsymbol{T}}, \tag{E.33}$$

with $\boldsymbol{\lambda}_{\ddot{\boldsymbol{X}}|\boldsymbol{Z}}$ being the diagonal matrix containing the non-negative eigenvalues $\boldsymbol{\lambda}_{\ddot{\boldsymbol{X}}|\boldsymbol{Z},\boldsymbol{i}}$ (resp. $\boldsymbol{\lambda}_{\boldsymbol{y}|\boldsymbol{z},\boldsymbol{i}}$). Let $\boldsymbol{\psi}_{\ddot{\boldsymbol{x}}|\boldsymbol{z}} = [\psi_{\ddot{\boldsymbol{x}}|\boldsymbol{z},\mathbf{1}}(\ddot{\boldsymbol{x}}), \ldots, \psi_{\ddot{\boldsymbol{x}}|\boldsymbol{z},\boldsymbol{n}}(\ddot{\boldsymbol{x}})] \triangleq \boldsymbol{V}_{\ddot{\boldsymbol{x}}|\boldsymbol{z}}\boldsymbol{\lambda}_{\ddot{\boldsymbol{x}}|\boldsymbol{z}}{}^{1/2}$ (resp. for $\boldsymbol{\phi}_{\boldsymbol{y}|\boldsymbol{z}}$).

Then we have the following result:

Theorem 9. *Conditional Independence Test Under the hypothesis H_0 ($X \perp\!\!\!\perp Y|Z$) we have that the statistic:*

$$T_{CI} \triangleq \frac{1}{n}Tr(\widetilde{\boldsymbol{K}}_{\ddot{X}|Z}\widetilde{\boldsymbol{K}}_{Y|Z} \tag{E.34}$$

has the same asymptotic distribution as:

$$\check{T}_{CI} \triangleq \frac{1}{n}\sum_{k=1}^{n}\mathring{\lambda}_k \cdot z_k^2, \tag{E.35}$$

where $\mathring{\lambda}_k$ are the eigenvalues of $\check{\boldsymbol{w}}\check{\boldsymbol{w}}^T$ and $\check{\boldsymbol{w}} = [\check{\boldsymbol{w}}_1, \ldots, \check{\boldsymbol{w}}_n]$ with the vector $\check{\boldsymbol{w}}_t$ obtained by stacking $\check{\boldsymbol{M}}_t = [\psi_{\ddot{\boldsymbol{x}}|\boldsymbol{z},\mathbf{1}}(\ddot{x}_t), \ldots, \psi_{\ddot{\boldsymbol{x}}|\boldsymbol{z},\boldsymbol{n}}(\ddot{x}_t)]^T[\phi_{\boldsymbol{y}|\boldsymbol{z},\mathbf{1}}(\ddot{y}_t), \ldots, \phi_{\boldsymbol{y}|\boldsymbol{z},\boldsymbol{n}}(\ddot{y}_t)]$

Similarly to the unconditional independence testing, we can do KCI-test by generating the approximate null distribution with Monte Carlo simulation. We first need to calculate $\widetilde{\boldsymbol{K}}_{\check{X}|Z}$

and $\widetilde{\boldsymbol{K}}_{Y|Z}$, and their eigenvalues and eigenvectors. We then evaluate T_{CI} according to Equation (E.34) and simulate the distribution of $\check{T}_{CI}$ given by Equation (E.35) by drawing i.i.d. χ^2_1 samples and summing them up with weight $\mathring{\lambda}_k$. Finally, the p-value is calculated as the probability of $\check{T}_{CI}$ exceeding T_{CI}.

E.3.6 Approximating the null distribution with a Gamma distribution

In addition to the simulation-based method to find the null distribution, as in [GFT+08], the authors provide approximations to the null distributions with a two-parameter Gamma distribution. The two parameters of the Gamma distribution are related to the mean and variance. In particular, under the null hypothesis that X and Y are independent (resp. conditionally independent given Z), the distribution of $\check{T}_{UI}$ given by (E.28) (resp. of $\check{T}_{CI}$ given by (E.35)) can be approximated by the $\Gamma(k, \theta)$ distribution:

$$p(t) = t^{k-1} \frac{e^{-t/\theta}}{\theta^k \Gamma(k)},$$

where $k = \mathbb{E}^2[\check{T}_{UI}]/\mathbb{V}ar(\check{T}_{UI})$ and $\theta = \mathbb{V}ar(\check{T}_{UI})/\mathbb{E}[\check{T}_{UI}]$. (resp $\check{T}_{CI}$). To compute the mean and variance of $\check{T}_{UI}$ (resp. $\check{T}_{CI}$) the authors use the following theorem:

Theorem 10 (Mean and variance of $\check{T}_{UI}$ and $\check{T}_{CI}$)**.** *1. Under the null hypothesis that X and Y are independent on the a given sample $\mathcal{D}$, we have:*

$$\mathbb{E}[\check{T}_{UI}|D] = \frac{1}{n^2} Tr(\widetilde{\boldsymbol{K}}_X) \cdot Tr(\widetilde{\boldsymbol{K}}_Y) \tag{E.36}$$

and

$$\mathbb{V}ar[\check{T}_{UI}|D] = \frac{2}{n^4} Tr(\widetilde{\boldsymbol{K}}_X^2) \cdot Tr(\widetilde{\boldsymbol{K}}_Y^2) \tag{E.37}$$

2. Under the null hypothesis that X and Y are independent conditionally to Z on the a given sample $\mathcal{D}$, we have:

$$\mathbb{E}[\check{T}_{CI}|D] = \frac{1}{n} Tr(\widetilde{\boldsymbol{w}}\widetilde{\boldsymbol{w}}^T) \tag{E.38}$$

and

$$\mathbb{V}ar[\check{T}_{CI}|D] = \frac{2}{n^2} Tr\left((\widetilde{\boldsymbol{w}}\widetilde{\boldsymbol{w}}^T)^2\right) \tag{E.39}$$

E.4 Differences

The unconditional independence testing method defined in [ZPJS12] is closely related to the one based on the Hilbert-Schmidt independence criterion (HSIC) proposed by [GFT+08]. Actually the statistics defined in [ZPJS12] (Equation (E.27) and Equation (E.28)) are the same as $HSIC_b$ defined in [GFT+08] but the asymptotic distributions are in different forms. In [ZPJS12], the asymptotic distribution only involves the eigenvalues of the two regular kernel matrices $\widetilde{\boldsymbol{K}}_X$ and $\widetilde{\boldsymbol{K}}_Y$, while in [GFT+08] it depends on the eigenvalues of an order-four tensor, which are more difficult to calculate.

It is important to notice that the test from [GFT+08] makes uses of many (biased) approximations that may explain the poor results that we obtained we using the (derived) version of the implementation of this test. Indeed, from this work only unconditional test of independence is described. However, this test is used in the kPC algorithm [GPR09] where the authors adapt the HSIC test from [GFT+08] to test conditional independences. It should be noticed that *(i)* the code is not available anymore from the location pointed by the paper; *(ii)* It is not clear from the paper how to compute the threshold for testing conditional independences with the null (approximated) norm of the covariance operator *(iii)* Our experiments using the independence test provided in the implementation of the kPC algorithm (accessed when code was still available) showed both inconsistent and inaccurate results.

E.5 Conclusions

Overall, both tests are very similar. They are actually identical for the unconditional independence test. The availability of the KCI test implementation, its accuracy when testing known independences and the description of the derived statistics for testing the conditional independences are the three reasons why we opted for this test in our work.

Appendix F

Résumé [Français]

F.1 Introduction

Depuis l'adoption des réseaux de télécommunications Internet dans le monde entier, dans les années 90, le domaine des réseaux de télécommunications a évolué très rapidement. Le nombres d'acteurs influençant ses performances et le nombre de services présents ne cessent d'évoluer. Aux vues du nombres de technologies sur lesquelles les réseaux de télécommunications s'appuient et aux vues du nombre de services existants, modéliser et étudier les performances des réseaux Internet est devenu un problème complexe. Dans un tel scénario, modifier l'un des composants ou l'un des mécanismes d'un réseau de télécommunications représente une opération qui est à la fois onéreuse et difficilement réversible. D'autre part, prédire l'effet d'un changement au sein d'un mécanisme influençant les performances d'un réseau de télécommunications devient difficile dû au nombre de paramètres qu'il est nécessaire de prendre en compte et l'impossibilité d'implémenter des expériences contrôlées. Afin de répondre à ces différentes problématiques, notre travail propose une approche causale afin d'étudier les performances d'applications telles que FTP ou de services tels que les CDNs. Sur la base d'observations passives exclusivement (aucune modifications ou actions sur le système étudié), une approche causale permet de modéliser le rôle des différents paramètres responsables des performances d'un réseaux de télécommunication. Une telle approche permet notamment de prédire comment un service réagira à des changements opérés sur les propriétés du réseau sur lequel ce service s'appuie sans qu'aucune action sur le système étudié ou qu'aucune observation supplémentaire ne soit nécessaire. Bien que la théorie de la causalité permet de résoudre les différents challenges intrinsèques à l'étude des performances des réseaux de télécommunications, elle souffre aussi d'un nombre de limitations dans ses implémentations qui empêche son utilisation au sein des réseaux de télécommunications. Notre travail présente une solution pour s'affranchir des limita-

tions qui existent dans l'implémentation de la théorie de la causalité et afin d'implémenter la théorie causale au sein des réseaux de télécommunications. Après avoir validé notre solution à l'aide de simulations, nous présentons les études de plusieurs scénarios mettant en avant la supériorité d'une approche causale sur les approches classiques et démontre les avantages et le potentiel d'une approche causale dans l'étude des performances des réseaux de télécommunications.

F.2 Une approche causale

F.2.1 Motivation

Notre travail s'appuie en grande partie sur les travaux de Judea Pearl [Pea09]. Dans ces travaux, Judea Pearl développe de nombreuses méthodes et théories afin de tirer profit d'une représentation graphique des liens de causalité qui peuvent exister entre les différents paramètres influençant les performances d'un système. L'utilisation de réseaux Bayésiens comme représentations de modèles causaux simplifie grandement l'étude de systèmes aussi complexes que les réseaux de télécommunications.

La raison pour laquelle nous nous intéressons à une approche causale dans notre étude des performances des réseaux de télécommunications vient de la stabilité des modèles causaux lors d'interventions. Une intervention correspond ici à la modification du comportement d'un élément du système étudié en le déconnectant de ses causes directes et indirectes et en fixant ses variations à une valeur (ou distribution) préalablement fixée. Dans cette optique, prédire l'effet d'un changement est rendue difficile par le nombre de dépendances qu'il est nécessaire de prendre en compte afin de ne pas obtenir une estimation biaisée d'une intervention. En effet, le plus grand risque lors de l'étude d'un système où beaucoup de paramètres doivent être pris en compte pour comprendre son fonctionnement vient du nombre d'interdépendances entre ses paramètres et la présence de possible fausses associations. Une fausse association entre deux paramètres d'un système correspond à ces deux paramètres apparaissant, dans le modèle du système correspondant, comme s'influençant l'un l'autre alors qu'il s'agit en réalité d'un mécanisme non représenté par le modèle qui influence ces deux paramètres et les fait apparaître comme associés. La présence de fausses associations au sein d'un modèle entraîne des estimations biaisées de l'action qu'aurait le changement d'un paramètre sur les performances du système étudié.

Par conséquent le but d'une étude causale est (i) d'identifier les dépendances causales entre les différents paramètres constituant le modèle du système que l'on veut étudier; (ii) L'utilisation de ce modèle pour éliminer les fausses associations, isoler l'effet d'un paramètre sur les performances du système afin

de prédire l'effet qu'un changement sur ce paramètre aurait sur les performances du système. L'utilisation des réseaux Bayésiens simplifie la résolution de ces deux problématiques et les deux sections suivantes expliquent comment (i) Construire un modèle causal graphique d'un système donné à partir de l'observation passive de ce système (ii) L'utilisation de ce modèle pour prédire l'effet qu'aurait une intervention sur les performances de ce système.

F.2.2 Inférence d'un modèle causal

La première étape lors de l'inférence du modèle causal, sous la forme d'un réseau Bayésien, d'un système consiste en la définition du dit système. Cette définition correspond au choix des paramètres qui seront présents dans notre modèle et qui devront capturer tous les mécanismes susceptibles d'influencer les performances de ce système. Cette étape est en grande partie dictée par la connaissance du domaine dans lequel le système se place et le problème de "feature selection" n'est pas au centre de notre travail.

Une fois la définition du système achevée, la deuxième étape consiste en l'observation de notre système dans différents scénarios. Cette phase correspond à l'acquisition de données qui constituera la seule et unique base de tout l'étude causale.

L'étape suivante consiste en l'inférence du modèle causal à partir des données (observations) récoltées lors de l'étape précédente. Plusieurs méthodes existent et nous nous cantonnerons ici à la méthode qui a été retenue au sein de notre travail qui correspond à une "contraint-based approach" dans laquelle certaines contraintes sont imposées par les données et le modèle est construit de façon à vérifier ces contraintes. Plus précisément, dans notre travail, nous utilisons l'algorithme PC [SG91] que nous décrivons ci-dessous.

L'algorithme PC prend comme entrée la série d'observations des paramètres de notre système. Pour chacun des paramètres du système il crée un sommet (ou noeud) et connecte ensuite tous les sommets du graphe ensembles (graphe complet). Pour tous les couples de paramètres du système, l'algorithme PC teste si ces deux paramètres sont indépendants. Si c'est le cas, l'arrête reliant les deux sommets correspondant est enlevée, sinon elle est laissée intacte. A la fin de cette étape certain noeuds (= sommets) sont toujours adjacents, le PC teste donc toutes les indépendances conditionnelles entre les paramètres dont les noeuds correspondant sont adjacents. En commençant par tester toutes les indépendances conditionnelles avec une taille de set conditionnel de 1, le PC incrémente graduellement la taille de ce set à chaque fois que toutes les indépendances pour une taille de set conditionnel donnée ont été testées et que le degré du graphe est supérieur à la taille du set conditionnel. Lorsque la taille du set conditionnel a atteint le degré du graphe, il n'est alors plus possible de trouver de nouvelles indépendances et la structure

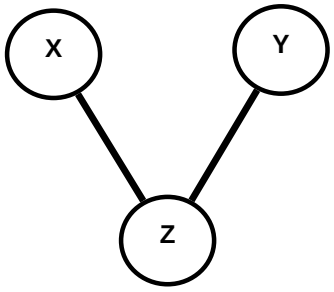


Figure F.1: Exemple de V-structure

du graphe a été obtenue. La dernière étape consiste en l'orientation des arrêtes présentes dans le graphe. En commençant par orienter les V-structures, voir Figure F.1, sur la base des tests d'indépendances effectués lors de l'étape précédente, et puis en orientant autant d'arêtes que possible sans créer de cycle ou de collider (dans $X \rightarrow Z \leftarrow Y$, Z est appelé un collider).

Un point important à retenir de cet présentation de l'algorithme PC est sa forte dépendance et sensibilité à la précision des tests d'indépendances effectués sur les données. En effet, toutes les étapes d'inférence du modèle causal se reposent sur les résultats des tests d'indépendance.

F.2.3 Prédire l'effet d'une intervention

Dans cette section nous supposons que nous avons déjà inféré le graphe représentant les différentes relations de dépendances causales entre les différents paramètres formant le système étudié. Ce modèle peut alors être utilisé pour identifier le set de paramètres qui sera en mesure de bloquer les paramètres qui influencent à la fois les performances du système et le paramètre dont nous souhaitons connaître l'influence causale sur les performances du réseau. [Pea09] définit un ensemble de critères graphiques qui nous permettent d'identifier facilement le set minimal de paramètres sur lesquels il suffit de conditionner pour bloquer les influences des paramètres qui ne doivent pas être pris en compte pour estimer l'effet d'un changement sur l'un des paramètres du système sur les performances du système.

Ce chapitre représentant un résumé en français de notre travail, nous ne présentons ici qu'un seul de ces critères qui est celui qui a été utilisé le plus largement au sein de notre travail le "back-door criterion". Il est cependant nécessaire d'introduire la notion de d-séparation qui permet d'obtenir une représentation graphique de la notion de dépendance.

Si l'on considère deux noeud, X_1 et X_2, au sein du graphe représentant le modèle causal de notre système, l'existence d'un chemin entre ces deux noeuds les rend dépendant et conditionner sur l'un des noeuds, W, présent sur le chemin

rend ces deux noeuds indépendants ($X_1 \perp\!\!\!\perp X_2|W$). Cependant, si l'on considère un chemin non orienté contenant un collider, Z, alors les deux noeuds sont indépendants ($X_1 \perp\!\!\!\perp X_2$) vis-à-vis de ce chemin mais conditionner sur un collider rend ces deux paramètres dépendant ($X_1 \not\perp\!\!\!\perp X_2|Z$).

Sur la base de ces définitions, il est donc possible de définir le "back-door criterion":

Definition 2. ***(Back-door criterion)*** *Un (ensemble de) paramètre(s) Z satisfait le back-door criterion pour paire de variables (X_i,X_j) au sein d'un graphe G si:*

1. *Aucun noeud dans Z est un descendant de X_i; et*
2. *Z bloque tous les chemins entre X_i et X_j qui contiennent une flèche dans X_i.*

En utilisant la définition du back-door criterion, il est alors possible de prédire l'effet qu'aura, sur X_j, une intervention sur X_i en utilisant comme notation $do(X = x)$ pour représenter l'intervention sur le paramètre X qui isole ce paramètre de ces causes directes et indirectes et fixe sa valeur à x.

Theorem 11. ***(Back-door adjustment)*** *Si un ensemble de paramètres Z vérifie le back-door criterion vis-à-vis d'une paire de paramètres (X_i, X_j), alors l'effet qu'aura, sur Y, une intervention sur X est $identifiable$ et donné par la formule suivante:*

$$P(x_j \mid do(X_j = x_j)) = \sum_z P(x_j \mid X_i = x_i, Z = z)P(Z = z). \tag{F.1}$$

Dans ce théorème $identifiable$ signifie que l'effet de cette intervention peut être estimé sans équivoque à partir des observations passives du système. En effet, il est important de noter que la partie droite de l'équation ne contient plus d'opérateurs do et que les quantités présentes peuvent être estimées à partir des données recueillies lors de la phase d'acquisition de données.

On voit ici tout l'intérêt d'adopter une approche causale qui non seulement permet de prédire comment un système réagira à un changement donné mais, de plus, ne requiert ni observations supplémentaires ni expériences.

Il est finalement important de noter que le Théorème 11 ne traite que les interventions appelées atomiques, assez simples, mais que des scénarios plus complexes peuvent être considérés, comme c'est le cas dans notre étude de l'impact du service DNS sur les performances d'un CDN, Section F.6.

F.3 Réseaux de télécommunications

La section précédente présente la théorie causale à la base de notre travail. Cependant il est important de garder à l'esprit que, si cette théorie ne fait aucune hypothèse concernant la nature du système où elle est appliquée il en va différemment de ses implémentations. Il est donc important de prendre en compte les spécificités du système étudié, dans notre cas les réseaux de télécommunications.

F.3.1 Le réseaux Internet

Dans cette section nous donnons une définition très concise des réseaux de télécommunications et de deux protocoles considérés comme centraux au sein de notre travail que sont IP et TCP. Pour une présentation plus étendue le lecteur peut soit se référer au Chapitre 1 soit à l'ouvrage d'excellente qualité [KR09].

Réseaux de télécommunications

D'un point de vue très général, les réseaux de communications correspondent à des systèmes constitués de noeuds (machines) responsable de faire transiter une information depuis une source (l'envoyeur) vers une destination (le récepteur). Les variations qui existent se trouvent dans le type de noeuds formant le réseaux (stations de bases, routeurs, ...) où la nature des liens connectant deux noeuds adjacents (câble, faisceaux hertziens, ...).

Dans notre travail nous nous intéressons à un réseau en particulier qu'est le réseau Internet. Ce réseau n'est pas à proprement parler un réseau mais un réseau de réseaux faisant cohabiter un nombre important et varié de technologies et d'acteurs et qui a pour but de les faire collaborer afin de rendre une communication entre tous leurs utilisateurs possible. De ce point de vue, modéliser les performances d'applications reposant sur Internet est une problématique relativement complexe. D'autant plus qu'il est en général difficile de pouvoir observer tous les paramètres influençant les performances d'une communication ayant lieu entre deux entités traversant de nombreux réseaux, tous ayant leurs spécificités, leur règles et leurs contraintes.

TCP/IP

Afin de permettre la communication entre deux entités distantes, sur la base du réseau Internet, il a été nécessaire de définir un certain nombre de normes permettant à deux machines de se comprendre. Ces règles ont été formalisées

à travers ce qui est connu comme les protocoles de communications. Dans le cas du réseaux Internet, les deux protocoles qui sont les plus importants pour nos études sont d'une part le protocole Internet (Internet Protocol ou IP) et le protocole TCP (Transmission Control Protocol).

IP est connu comme un protocole "best-effort" dans le sens où il n'est responsable que de transmettre une information d'un noeud du réseau au suivant. Il ne se soucie pas de l'occurrence d'erreurs telles que la perte d'information due à des dysfonctionnements soit au sein d'une machine, telle qu'un routeur, soit du système lui même, comme dans les situations de congestion où le trafic donné au réseau est supérieur à sa capacité et des informations sont donc supprimées pour éviter le crash du système. Il est cependant important de noter qu'IP définit l'adressage de toutes les machines connectées au réseau Internet, leur permettant donc de s'identifier et se joindre.

TCP, en contre-partie, assure une transmission fiable des données de bout en bout. Pour ce faire, TCP implémente plusieurs mécanismes afin de réagir à l'occurrence de pertes, d'adapter la vitesse d'envoi des données à l'état du réseaux (*congestion control*). TCP définit aussi un paramètre envoyé par le récepteur à l'envoyeur pour lui communiquer les ressources qu'il peut allouer à leur échange d'information, ce qui permet à l'envoyeur d'adapter sa vitesse d'envoi à la capacité du récepteur.

Responsable de la transmission de la plus grande partie des données sur Internet (80%), TCP est au centre de notre travail. Afin de s'assurer de la bonne transmission des données sur le réseau, les informations que veulent s'échanger l'envoyeur et le récepteur sont divisées en paquets et l'envoi d'un paquet par l'envoyeur est suivi de l'envoi d'un acquittement par le récepteur, témoin de la transmission correcte d'un paquet de l'envoyeur au récepteur.

Afin d'estimer l'état du réseau, l'envoyeur enregistre le temps nécessaire à l'envoi d'un paquet et la réception de son acquittement, le Round Trip Time (RTT). A l'aide des acquittement et de l'utilisation de timers, l'envoyeur est aussi capable de détecter et réagir à l'occurrence de pertes sur le réseau. TCP adapte la vitesse d'envoi à l'état du réseau en augmentant linéairement la quantité de paquets qui peuvent être envoyés sans attendre d'acquittement et divisant cette quantité lors de la détection d'un perte.

Dans le cas de la version TCP Reno, [PFTK98] ont pu inférer un modèle de la vitesse de transmission des données au niveau de l'envoyeur en fonction de l'état de congestion du réseau qui peut se résumer dans l'équation suivante:

$$throughput = C.\frac{MSS}{RTT \cdot \sqrt{p}}, \tag{F.2}$$

où C est une constante, MSS est la Maximum Segment Size (la plus grosse taille de paquet qui peut être envoyé de l'envoyeur au récepteur), RTT le round trip time et p la probabilité de perte.

Contraintes

Une remarque importante, en observant l'équation F.2, dans notre étude des performances des réseaux de télécommunications les paramètres présents ne présentent pas, à priori, de dépendances linéaires. Ceci pose un problème important lors de l'application de la théorie causale à l'étude des réseaux de télécommunications. En effet, comme souligné lors de présentation de l'algorithme PC, la précision de cet algorithme est fortement liée à la précision des tests d'indépendances. Malheureusement, dû à la faible utilisation d'une approche causale dans le milieu de l'ingénierie, les implémentations existantes d'algorithmes d'inférence de modèle causaux reposent sur des hypothèses de linéarité et normalité. C'est par exemple le cas du Bayes Net toolbox [Mur01], Tetrad [SGS01] ou encore le paquet R pcalg [SGS01].

La deuxième contrainte qui se présente vient donc de la distribution des paramètres qui doivent être considérés lors de l'étude des performances des réseaux de télécommunications. En effet, Figure F.2 présente les distributions de certains paramètres présents lors d'études des réseaux de télécommunications. Il est donc important de noter que

- Les paramètres à considérer représentent un mélange de paramètres continus et discrets
- Leurs distributions ne suit pas de distributions connues, en particulier Gaussienne
- La plupart des paramètres ont une distribution multimodale et irrégulière ce qui rend très difficile leur approximation à l'aide de familles de distributions connues (Gaussiennes, Beta, Gamma, . . .)

Cette deuxième contrainte entraîne deux challenges. Le premier, similaire à l'observation faite précédemment vis-à-vis de la nature des dépendances entre les paramètres à considérer, se rapporte au test d'indépendance à utiliser lors de l'inférence des modèles causaux des systèmes étudiés. Le deuxième apparaît assez clairement si l'on observe l'Équation F.1. Lors de l'estimation de la distribution d'un (ensemble de) paramètre(s) suite à une intervention sur l'un des autres paramètres du système considéré, il est nécessaire d'estimer des distributions conditionnelles multidimensionnelles. Ce problème est relativement complexe si l'on ne peut faire aucune hypothèse sur la distribution des paramètres.

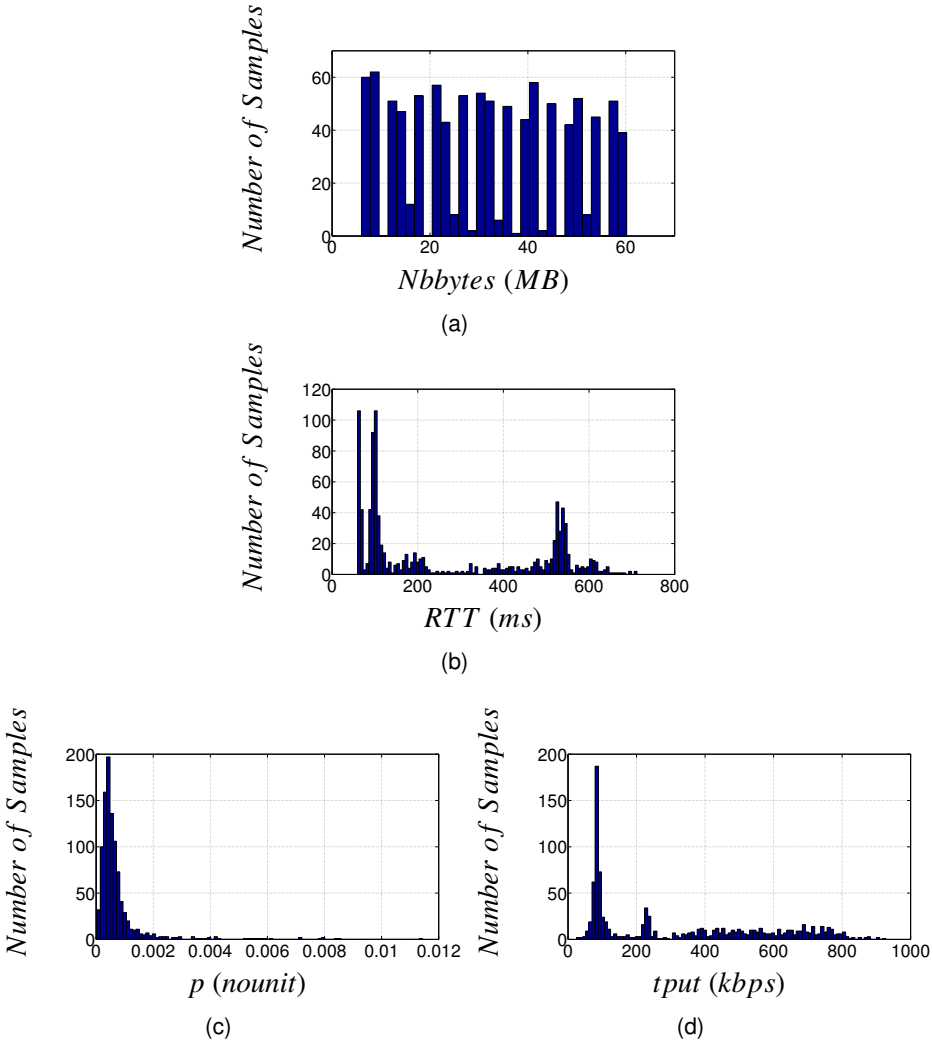


Figure F.2: Histogrammes des paramètres *nbbytes (a)*, *rtt (b)*, *p (c)* et *tput (d)* paramétres observés lors de l'étude du traffic FTP, presenté dans la Section 5.2

F.4 Adaptation d'une approche causale aux contraintes inhérentes à l'étude des performances des réseaux de télécommunications

F.4.1 Test d'indépendance

La plupart des implémentations existantes de l'algorithme PC testent l'absence de corrélation (partielle) pour tester l'indépendance (conditionnelle) entre deux paramètres. Malheureusement, ce test n'est valide que sous les hypothèses de normalité et linéarité. Bien que certaines études aient montré que le PC four-

nit des résultats satisfaisant si l'on relâche la contrainte de normalité [VD08], l'absence de linéarité a une très forte influence sur le PC.

Le test utilisé dans notre travail est le Kernel Conditional Independence test (KCI) [ZPJS12]. Ce test ne fait aucune hypothèse sur la nature des dépendances entre les paramètres dont l'indépendance est testée et ne nécessite pas l'estimation de la distribution des paramètres donc ne fait aucune hypothèse non plus sur leurs distributions. Cependant, ses avantages font du KCI un test plus complexe qui entraine une forte augmentation du temps nécessaire pour tester l'indépendance entre deux paramètres. D'autre part, dans son implémentation, le KCI utilise des factorisations de Cholesky qui sont connues pour faillir dans certains cas de figures où le nombre d'observations est trop important (voir, e.g., Appendix B dans [Tip01]).

Afin de s'affranchir de ces deux limitations, nous avons adapté le KCI en le couplant à une approche inspirée du bootstrap. Pour tester l'indépendance $X \perp\!\!\!\perp Y|Z$, nous commençons par créer N sous datasets de taille S (inférieure à celle du dataset initial) à partir du dataset initial en utilisation du random resampling avec remise. $X \perp\!\!\!\perp Y|Z$ est alors testé, avec le KCI, sur chacun des sous datasets et, en prenant la médiane des p-values des différents sous tests, nous obtenons le résultat du test $X \perp\!\!\!\perp Y|Z$. Cette approche permet de s'affranchir des limitations de l'utilisation de factorisations de Cholesky car les tests sont effectués sur des datasets de plus petite taille. D'autre part, nous avons pu observer que le temps nécessaire pour effectuer un test d'indépendance avec le KCI augmente plus que linéairement avec le nombre d'observations, par conséquent nous arrivons aussi à diminuer le temps nécessaire pour tester une indépendance et donc, plus globalement, à l'inférence d'un modèle causal à l'aide du PC+KCI. Nous pouvons ensuite diminuer encore en-deçà le temps d'inférence en utilisant plusieurs machines et en effectuant les sous-tests en parallèle.

À l'aide d'une dataset artificiel, nous avons pu vérifier la validité de cette approche à la fois en terme de précision et en terme de temps de calcul, Section 3.3.2. La même étude nous a aussi permet d'étudier l'effet du choix de S et N et de paramétrer notre méthode.

F.4.2 Estimation des distributions

En ce qui concerne l'estimation des distributions multidimensionnelles, nous avons utilisé des noyaux où histogrammes pour estimer les marginales et des copulae [JDHR10] pour capturer les dépendances entre les différentes marginales.

Une copula est un fonction de distribution de probabilité multidimensionnelle pour laquelle chaque marginale suit une distribution uniforme. Une copula est utilisée pour décrire les dépendances entre les différentes marginales d'une

distribution multidimensionnelle donnée. Plus formellement, une copula, C peut être définie comme

$$\begin{aligned} C :[0,1]^n &\longrightarrow [0,1] \\ U_1,\ldots,U_n &\longrightarrow C(U_1,\ldots,U_n) \end{aligned} \tag{F.3}$$

Le théorème de Sklar stipule que, si F est une CDF multidimensionnelle avec des marginales $(F_1,\ldots,F_n)$, il existe une copula C telle que:

$$F(x_1,\ldots,x_n) = C(F_1(x_1),\ldots,F_n(x_n)). \tag{F.4}$$

Dans le cas bi-dimensionel, en dérivant l'Equation (F.4) par rapport à X_1, X_2 nous obtenons:

$$f(x_1,x_2) = c(F_1(x_1),F_2(x_2))f_1(x_1)f_2(x_2), \tag{F.5}$$

où $f_i(x_i)$ est la fonction de densité de probabilité de la marginale F_i et c est la dérivée de C. Si l'on assume $f_2(x_2) > 0 \;\forall\; x_2$, on obtient:

$$f_{X_1|X_2}(x_1,x_2) = c(F_1(x_1),F_2(x_2))f_1(x_1). \tag{F.6}$$

Ce résultat est très utile pour l'estimation d'une intervention en utilisant le théorème du Back-door adjustement, Théorème 11, où des probabilités multi-dimensionnelles conditionnelles doivent être estimées.

F.5 Validation de notre approche à l'aide d'un émulateur

F.5.1 Scénario

Afin de valider notre approche et les différents choix faits pour adapter les méthodes existantes aux contraintes des réseaux de télécommunications, la première étape a été de tester celle-ci dans un environnement où des interventions étaient possibles. Pour ce faire nous avons utilisé l'outil Mininet [HHJ+12] afin d'émuler le réseau représenté dans la Figure F.3 où un client C_1 télécharge des fichiers présent sur un serveur S_1. Afin de recréer une situation proche d'un scénario réel, nous avons ajouté un deuxième client et un deuxième serveur et créer du cross-traffic.

F.5.2 Dataset

En capturant les paquet au niveau du serveur S_1 pour 1000 connections, nous obtenons le dataset présenté dans la Table F.1

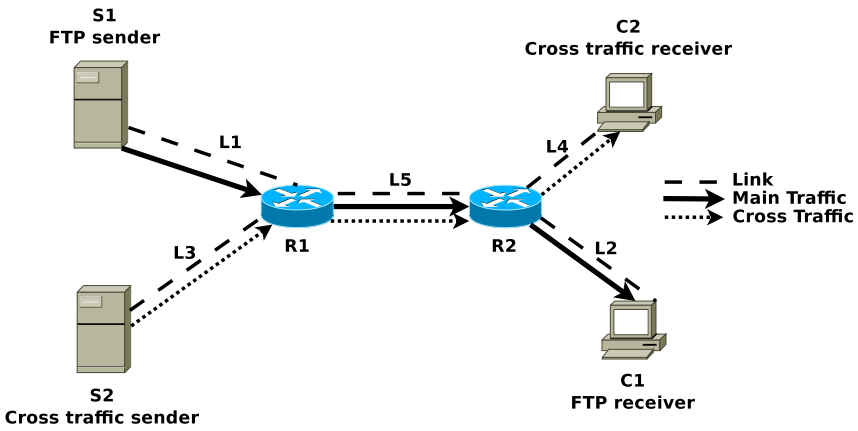


Figure F.3: Emulation d'un réseau avec Mininet

Table F.1: Résumé du dataset des expériences du réseau émulé à l'aide de Mininet

Paramètre	Définition	Min	Max	Avg	CoV
bw	capacité minimum (MBps)	1	25	7.1	0.91
delay	delai de propagation (ms)	30	180	86	0.48
queue1	taille du buffer **R1** (pkts)	10	400	98	1.10
queue2	taille du buffer **R2** (pkts)	10	400	100	0.99
nlac	Capacité du plus petit lien (kBps)	12	3070	630	5.00
rwin	Receiver window communiqué par**C1** (kB)	74	2155	288	0.65
bufferingdelay	partie du RTT due délai d'attente (ms)	1	6760	120	2.40
rtt	Round Trip Time (ms)	84	6910	310	0.99
timeouts	nombre de timeouts (unités)	0	682	79	1.50
retrscore	fraction de paquets retransmis (pas d'unité)	0	0.61	0.04	5.10
p	fraction d'évenement de perte (pas d'unité)	0	0.64	0.04	8.40
nbbytes	Nombre de bytes envoyés par le serveur (MB)	6	150	110	0.21
tput	throughput (kBps)	6	1100	280	0.81

F.5.3 Modèle causal

En utilisant le PC et la version modifiée du KCI nous obtenons le graphe causal présenté dans la Figure F.4. On peut noter que les paramètres présents dans le modèle de l'Equation (F.2) sont tous présents comme parents directs du throughput. D'autre part, dans le modèle de l'Equation (F.2), seule la partie de TCP responsable d'adapter la rapidité d'envoi des paquets à l'état du congestion est prise en compte. L'autre partie, où l'envoyeur adapte sa vitesse

d'envoi aux ressources du récepteur est capturé par le paramètre *receiver window* (*rwin*), parent direct du throughput lui aussi. Ces remarques nous conforte dans la validité de notre modèle.

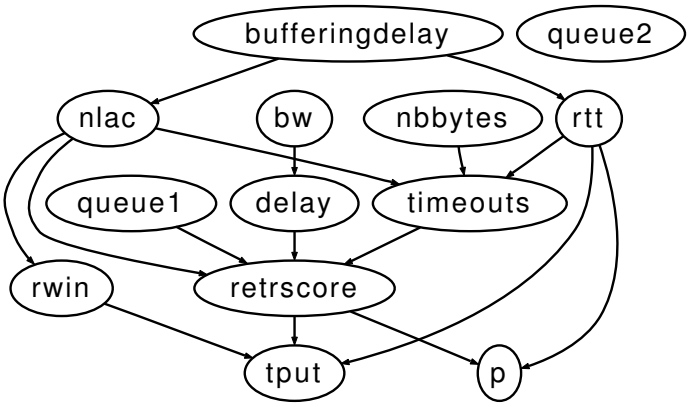


Figure F.4: Modèle causal in ferré par l'algorithme PC, avec le KCI, pour le trafic du réseau emulé

F.5.4 Prédiction d'intervention

Notre intéret dans une approche causale réside dans la capacité qu'il offre de prédire l'effet d'interventions sur les performances du système. Aux vues de nos données et de notre système, l'étape suivante consiste à intervenir manuellement sur le router R_1 ou R_2 en le forçant à écarter 1% des paquets et obtenir le throughput dans cette situation. Nous pouvons ensuite comparer ce throughput avec celui estimé avec les données présentes dans la Table F.1 et notre modèle causal, Figure F.4.

Les résultats sont présentés dans la Figure F.5, où l'on représente le throughput pré-intervention (ligne trait-point bleue), le throughput observé en forçant l'un des routeurs à écarter 1% des paquet (ligne solide verte) et notre estimation (ligne en pointillés rouge). On peut voir que notre prédiction parvient à estimer avec une bonne fidélité le comportement du système lorsque l'on crée 1% de pertes.

Afin de montrer l'avantage d'une approche causale sur une approche classique nous comparons l'espérance du throughput observé lorsque l'on force 1% de perte, l'espérance du throughput estimée à l'aide de notre approche causale et enfin l'espérance du throughput que l'on obtient en sélectionnant dans les données initiales les connections où 1% de perte est observée. La dernière estimation représentant l'approche qui serait adoptée naïvement sans utiliser notre modèle causal. Nous obtenons des espérances de 83 kBps, 98 kBps et 133 kBps respectivement, comparées à une espérance du throughput de

280 kBps avant intervention. Ces résultats montrent (i) La validité de notre approche et nos méthodes (ii) L'avantage d'une approche causale sur une approche classique.

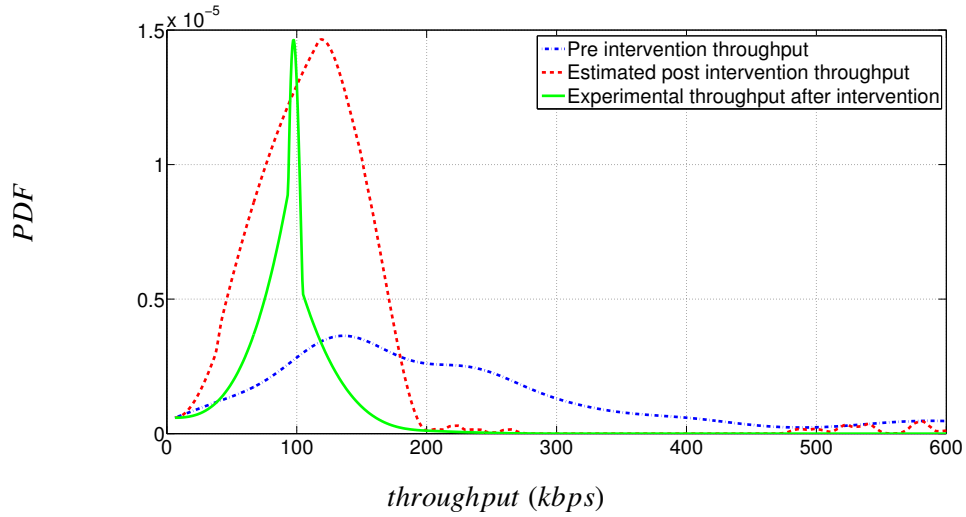


Figure F.5: Effet d'une intervention sur le taux de retransmissions, sur la distribution du throughput

F.6 Études causales de systèmes réels

Après avoir validé nos méthodes, nous pouvons les utiliser pour l'étude cas réels qui sont présentés dans cette section.

F.6.1 FTP

Notre première étude a consisté en l'étude des performances d'un serveur FTP installé au sein d'EURECOM. Relativement semblable au scénario emulé avec Mininet, ce scénario est cependant beaucoup plus complexe dû à la grande quantité de paramètres influençant les performances que l'on peut observées au niveau du serveur et l'absence de vision ou contrôle sur ces paramètres.

Données

En demandant à différents clients à différents endroit en Europe de télécharger des fichiers présents sur notre serveur FTP, nous avons pu capturer les différents paquets échangés lors de ces téléchargements et estimer à l'aide d'Intrabase [SBUK+05] le dataset présenté dans la Table F.2.

Table F.2: Résumé du dataset du trafic FTP

Paramètre	Définition	Min	Max	Avg	CoV
dist	distance entre serveur et client (km)	14	620	250	0.95
tod	Heure de la journée (s)	740	82000	46000	0.53
nbbytes	Nombre de bytes envoyés par le serveurr (MB)	6	60	32	0.51
nlac	Capacité du plus petit lien (kBps)	48	43000	5900	1.70
nbhops	Nombre de routeurs entre le serveur et le client (hops)	9	27	11	0.24
rtt	Round Trip Time (ms)	60	710	270	0.76
bufferingdelay	part du RTT due à l'attente (ms)	4.2	470	84	1.20
retrscore	fraction de paquets retransmis (pas d'unité)	0.001	0.01	2e-3	0.92
p	fraction d'événements de perte (pas d'unité)	3e-5	0.01	7e-4	1.30
tor	fraction du *retrscore* due à des timeouts (no unit)	0	0.01	6e-4	1.70
rwin	Receiver Window (kB)	11	254	137	0.68
tput	throughput (kBps)	24	928	332	0.77

Modèle causal

A partir des données résumées dans la Table F.2 et en utilisant le PC+KCI nous obtenons le graphe causal représenté dans la Figure F.6. Il est possible de faire certaines observations vis-à-vis de ce graphe.

Tout d'abord, nous voyons que le graphique est entièrement cohérent avec les connaissances du domaine TCP. En particulier, les parents de *tput*, à savoir *{p, rtt, RWIN}*, sont les paramètres empiriques qui influent sur le *throughput* et {p, *rtt*} sont présents dans le modèle de Pahdye, Equation(F.2).

Un autre résultat intéressant est que les délais d'attente, une fois normalisés par le nombre de paquets (*tor*), n'ont pas d'impact sur le *throughput* (*tput*). Par conséquent, intervenir sur ce paramètre ne conduira pas à une amélioration directe du *throughput*. Intervenir sur un paramètre peut être considéré comme de déconnecter ce paramètre de tous ses parents, et crér un nouveau parent qui représente notre intervention. Comme il n'y a pas de chemin entre les *tor* et *tput* dans le graphe $G_{\overline{tor}}$ (le graphique modifié où tous les arrêtes qui entrent dans *tor* sont supprimées) si nous devions intervenir sur le paramètre *tor*, cela serait sans effet sur throughput.

Le graphique montre une dépendance causale entre le RTT et p. Une telle dépendance était déjà présente dans le modèle déduit dans l'étude précédente, la Figure F.4, et pourrait être expliqué par la relation entre la gigue et

la perte. Il est important de noter que cette dépendance n'est pas présent dans le modèle de Padhye et qu'elle montre aussi les limites des modèles empiriques. Ce dernier point illustre l'avantage d'une étude causale sur une étude purement statistiques ou empiriques. En étudiant simplement les corrélations entre les p et $tput$ et entre rtt et $tput$, nous ne verrions pas la dépendance entre rtt et p et obtiendrions des estimations biaisées de nos interventions sur ces deux paramètres.

La présence d'un sous-graphe isolé entre $dist$ et $nbhops$ illustre le fait que la distance est liée au retard de propagation, qui n'a pas de dépendance avec le RTT. De même, le nombre de routeurs traversés le long du chemin entre le client et le serveur, est très dépendant de la distance et ne présente pas de variations importantes dans nos données (voir Table F.2). p représente la probabilité d'un événement de perte alors que $retrscore$ représente la probabilité d'une perte de paquets. Compte tenu de la différence entre p et $retrscore$ et les faibles valeurs que nous observons pour ces deux paramètres, leurs variations relatives et absolues, il est difficile de saisir pleinement leur dépendance.

L'influence de tod sur RTT saisit l'effet des heures de pointe, lorsque le réseau est très chargé. D'autre part, la dépendance trouvé entre tod et $rwin$ est plus subtile. Il peut être expliqué par le fait que le client TCP peut mettre en oeuvre la fonction de Receiver-Window-Auto-Tuning (RWAT) [FSS02] pour adapter la taille de la fenêtre de réception au produit de la bande passante et délai (BDP) qui à son tour est influencé par le moment de la journée. Toutefois, intuitivement nous pourrions penser que tod devrait influer sur le RTT et $NLAC$ qui, à son tour, pourrait influencer $rwin$. Comme le $NLAC$ est calculée en utilisant l'outil PPRate [ENUK06], son estimation est moins précise lorsqu'il est utilisé avec des traces capturés au niveau du serveur où il peut sous-estimer la capacité des chemins. Dans ce cas, l'effet des heures de pointes (tod) sur la capacité du chemin, qui à son tour un aurait un impact sur $rwin$, pourraient ne pas être capturé par l'estimation de $NLAC$.

Le chemin $NLAC \rightarrow bufferingdelay \rightarrow rtt \rightarrow tput$ montre que le retard de mise en attente, dans les routeurs traversés le long du chemin entre le serveur et le client, est le principal facteur influençant le RTT et joue un rôle important dans la performance de TCP.

Interventions

A l'aide du modèle causal obtenu nous pouvons estimer l'effet sur le throughput d'interventions sur les paramètres de notre système. Figure F.7 représente l'évolution du throughput à la suite d'une intervention sur le RTT et Figure F.8 à la suite d'une intervention sur le paramètre p. Ces deux figures présentent aussi les résultats obtenus avec une approche naïve où l'on peut voir que l'on surestime l'effet des interventions. Ceci est expliqué par notre modèle causal

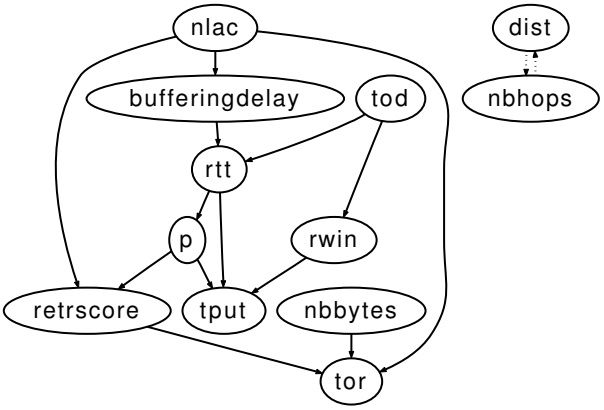


Figure F.6: Graphe causal modélisant les transferts FTP dans le cas de trafic réel

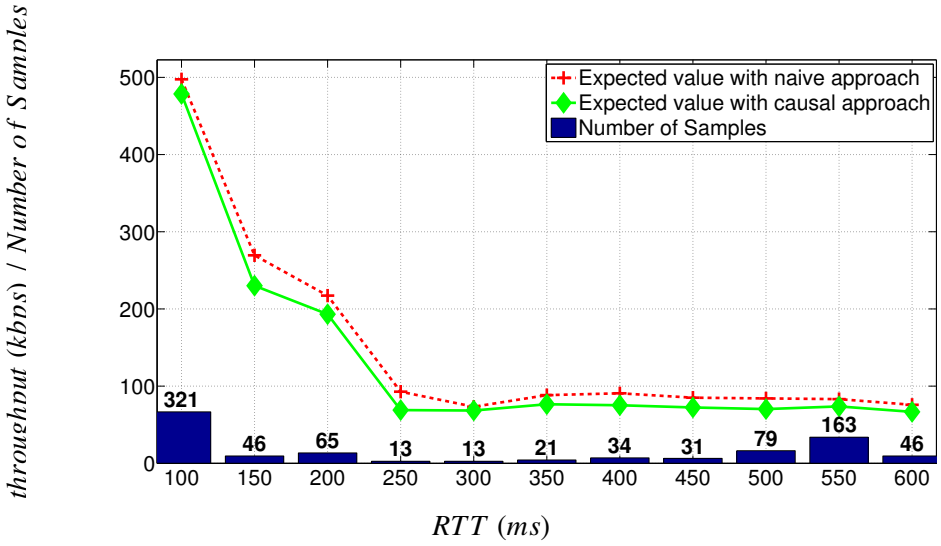


Figure F.7: Effect une intervention sur RTT, sur la distribution du throughput

où l'approche naïve consistant à conditionner sur la variable d'intervention "ouvre" les chemins entre la variables influençant à la fois la variable d'intervention et le throughput et où la variable d'intervention est un collider. Celui montre à nouveau l'intérêt d'une approche causale où l'on bloque ces fausses associations. Finalement, les données présentes sous forme d'histogrammes en bleu sur les Figures F.7 et F.8 représentent le nombre d'observations utilisé pour éstimer l'intervention correpsondante. Notre approche ne requiert donc pas une quantité de donnée trop importante (notamment comparée à une approche beaucoup moins formelle présentée dans [TZV+08]).

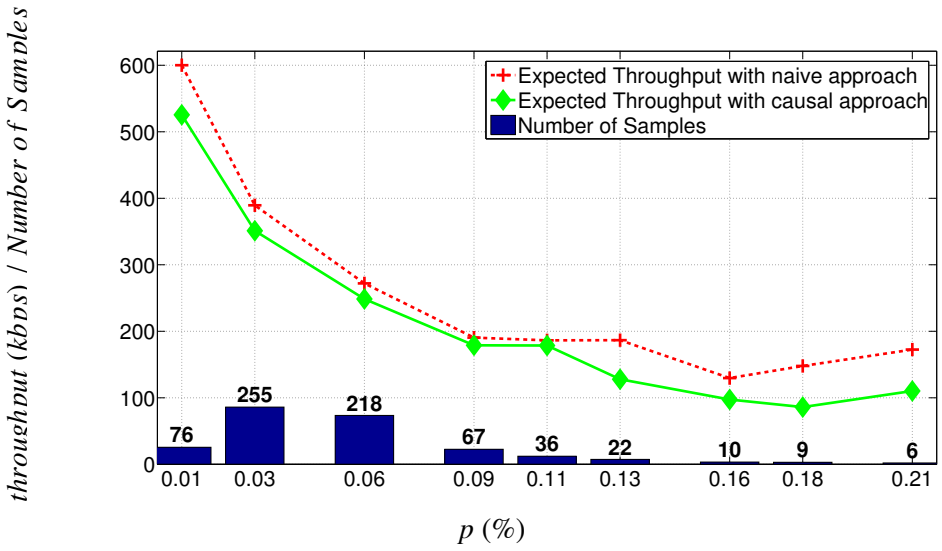


Figure F.8: Effet d'une intervention dur p, sur la distribution du throughput

F.6.2 DNS et CDN

La dernière étude qui a été faite est celle de l'impact du service DNS sur les performance du CDN Akamai. Pour cette étude nous comparons les performances de clients d'un FAI en Italie qui accèdent à des ressources détenues par Akamai et qui utilisent deux types de service DNS. Soit le DNS de leur fournisseur, soit un DNS publique le DNS de Google. Certaines études [HMLG11, AMSU10] ayant trouvé que l'utilisation des DNS publiques aurait un impact négatif sur les performances des utilisateurs des CDNs, dû à leur incapacité à communiquer avec précision l'emplacement de leurs clients, notre étude tente de mieux comprendre et formaliser et estimer cet impact.

Data

Afin d'observer les requêtes DNS des utilisateurs du service DNS de leur fournisseur nous nous plaçons à un point de présence (PoP) du réseau du fournisseur et observons tout le trafic dirigé vers Akamai et à l'aide de l'outil Tstat [FMM+11] nous obtenons les données résumées dans la Table F.3 correspondant à l'observation de 10000 connections. En nous plaçant à un point de présence du fournisseur nous découpons notre vision du réseau en deux. Celle avant le PoP (paramètres *isp*-*) et celle après le PoP (paramètres *inet*-*).

Table F.3: Summary of the different parameters

Parameter	μ	min	max	σ	CoV
dstip	N.A.	1300	34000	N.A.	N.A
dns	N.A	1	3	N.A.	N.A.
dow	N.A.	4	7	N.A.	N.A.
tod (s)	7100	52000	78000	4400	0.1
isprttavg (ms)	76	0	19000	460	6.1
isprttstd (ms)	100	0	37000	960	9.2
ispnbhops	1.8	1	3	0.51	0.3
inetrttavg(ms)	26	0.48	660	27	1.0
inetrttstd (ms)	8.2	0	4700	61	7.5
inetnbhops	9.4	2	21	2.8	0.3
rwin0	0.83	0	360	11	13
rwinmin (kB)	31.3	0.004	65	22.5	0.9
rwinmax (kB)	213	17.5	2625	150	0.7
cwinmax (kB)	150	7.3	1625	103	0.7
cwinmin (kB)	0.9	0.001	1.5	0.6	0.7
retrscore	0.005	0	0.19	0.009	1.9
rto (bool)	0.11	0	1	0.32	2.8
nbbytes (MB)	23.8	2.1	3875	138	5.7
tput (Mbps)	3.2	0.006	35	2.6	0.8

Table F.4: Résumé des différentes métriques pour les deux DNS: DNS Local (LD) and DNS Google (GD) (*dow* et *tod* ayant les mêmes variations, ces deux paramètres ne sont pas présentés)

Par	μ		min		max		σ	
	LD	GD	LD	GD	LD	GD	LD	GD
isprttavg (ms)	80	61	0	0	19000	15000	470	440
isprttstd (ms)	1100	76	0	0	32000	37000	920	1100
ispnbhops	1.8	1.9	1	1	3	3	0.53	0.4
inetrttavg (ms)	20	48	0.48	11	510	660	20	38
inetrttstd (ms)	8.6	6.5	0	0	4700	1400	65	44
inetnbhops	8.7	12	2	5	17	21	2.4	2.7
rwin0	0.97	0.29	0	0	330	360	12	9.2
rwinmin (kB)	35	12	0.004	0.03	65	65	28	14
rwinmax (kB)	213	213	18	18	2625	2000	150	138
cwinmax (kB)	163	118	7.3	7.8	1625	738	108	72
cwinmin (kB)	0.9	1.2	0.001	0.001	1.5	1.5	0.6	0.5
retrscore	0.005	0.004	0	0	0.19	0.06	0.01	0.01
rto (bool)	0.11	0.11	0	0	1	1	0.32	0.31
nbbytes (MB)	29	7	2.1	2.1	3875	1375	150	44
tput (Mbps)	3.2	3	0.006	0.007	35	29	2.7	2

Modèle causal

A partir des données résumées dans la Table F.3 et en utilisant le PC+KCI nous obtenons le graphe causal représenté dans la Figure F.9. Il est possible de faire certaines observations vis-à-vis de ce graphe.

Nous pouvons observer que le jour de la semaine (Dow) et le moment de la journée (tod) sont deux paramètres exogènes, ce qui n'est pas surprenant.

Nous pouvons aussi observer que le moment de la journée (*tod*) impacte le délai entre la sonde et le serveur (*inetrttavg*), qui capte le *peak hours*.

D'autre part, nous observons que le jour de la semaine (*Dow*) a une influence sur le service DNS utilisé par les clients (*dns*). Étant donné que nos observations sont faites sur deux jours seulement (jeudi et dimanche), nos conclusions sont limitées. Cependant, en regardant les données, il apparaît que le jeudi 72 % des connexions utilisent le service DNS de l'FAI contre 28% utilisant le service de DNS de Goole, tandis que le dimanche 93 % des connexions utilisent le service DNS de l'FAI contre 7 % utilisant le Service DNS de Google. Il serait intéressant d'identifier les clients qui utilisent un service DNS et de comparer leurs emplacements avec ceux des clients utilisant un service DNS différent afin de mieux comprendre cette dépendance. Le jour de la semaine peut capturer la différence dans l'utilisation d'Internet et les appareils utilisés chez soi et au travail. Cependant, pour des raisons de confidentialité, les adresses IP des clients sont obscurcis, ce qui nous empêche d'enquêter sur cette hypothèse.

L'une des dépendances les plus intéressants, qui a motivé ce travail, est celle présente entre le DNS (*dns*) et le RTT externe (*inetrttavg*). Nous avons pu observer dans les logs de Tstat que les serveurs joint par les clients utilisant le service DNS de leur FAI sont souvent situés à l'intérieur de l'AS de l'FAI. Cela n'est pas le cas pour les clients qui utilisent le service DNS de Google. Comme mentionné dans l'introduction et dans [HMLG11], les clients qui utilisent le service DNS local bénéficie d'une redirection vers des serveurs plus proches que ceux des clients utilisant un service de DNS public. Nous observons un RTT externe moyen de 20 ms pour les utilisateurs du services DNS de leur FAI, tandis que les utilisateurs de du service DNS Google observent un RTT externe qui est en moyenne de 48 ms (voir la Table F.4).

Nous pouvons également observer que les paramètres liés à l'estimation de l'état de congestion au niveau de l'envoyeur (*cwinmin* et *cwinmax*) ont un impact direct sur le throughput (*tput*). En outre, le minimum de congestion estimé par TCP au niveau de l'envoyeur (*cwinmin*) a comme parent le paramètre lié au choix du service DNS (*DNS*). Sa valeur moyenne pour les clients utilisant le DNS de Google est de 1,2 kb contre 0,9 kb pour les utilisateurs utilisant le DNS de leur FAI, voir la Table F.4.

Un paramètre présent dans un modèle causal représente également les mécanismes capturés par ce paramètre. Tel est le cas de *cwinmin* qui capte également le paramétrage TCP du serveur, telle que la fenêtre de congestion initiale qui est liée à la fenêtre d'encombrement minimale.

Les clients qui utilisent le DNS de leur FAI récupèrent souvent leurs objets à partir de serveurs qui sont situés *à l'intérieur* du réseau du FAI. Ces serveurs peuvent avoir une configuration différente de celle des serveurs accèdés par les utilisateurs du service DNS de Google. Cette hypothèse pourrait également expliquer le fait que les deux services DNS conduisent à un débit similaire mal-

gré une différente de RTT. Deux autres raisons pourraient être l'impact des pertes sur la fenêtre de congestion ou l'impact de la charge des serveurs contactés par les clients.

Pour capturer la charge du serveur, nous estimons le temps de traitement d'une requête HTTP par le serveur étudié. Ce temps est défini à partir du moment où un serveur reçois la requête HTTP/GET du client et le moment auquel le serveur envoie le premier paquet de données. Cependant, le temps de traitement des serveurs affiche une valeur moyenne de 43 ms pour les utilisateurs de DNS de leur FAI contre 64 ms pour les utilisateurs du service DNS de Google. Un temps de traitement plus élevé pour les serveurs contactés par les utilisateurs du service DNS de Google suggère donc qu'ils sont plus chargés. D'autre part, la fenêtre de congestion est aussi impacté par les pertes, mais peu de pertes sont actuellement détectées au cours de la période d'observation du système et aucune dépendance significative n'est constatée entre le taux de pertes (*retrscore*) et le service DNS (*dns*).

Il n'est par ailleurs pas surprenant que la RTT interne (*isprttavg*) soit un parent du throughput (*tput*). L'absence d'une dépendance entre le moment de la journée (*tod*) et le RTT interne peut être expliqué par le fait que tous les utilisateurs observés utilisent le même chemin "interne" (le chemin d'accès des utilisateurs jusqu'à la sonde).

Nous voyons aussi que la fenêtre de réception maximale annoncée par le client (*rwinmax*) a comme parent le moment de la journée (*tod*). Un tel effet peut être dû aux mécanismes d'ajustement automatique de la mémoire tampon au niveau de TCP [FSS02]. Ce mécanisme est responsable de l'ajustement de la fenêtre de réception en fonction de la quantité et la fréquence des données reçues par le client qui, elle même, est influencée par le moment de la journée.

Nous pouvons remarquer l'absence d'arête entre le DNS (*dns*) et l'adresse IP de destination (*dstip*) et l'absence de toute adjacence entre la taille de l'objet (*nbbbytes*). Une explication sur l'absence de ces arrêtes peut venir de la répartition inégale des serveurs contactés par les utilisateurs du service DNS de leur ISP par rapport à la distribution des serveurs contactés par les utilisateurs du service DNS de Google. Une solution afin de détecter des dépendances les plus faibles consiste à augmenter le taux d'acceptation dans les tests d'indépendance. Augmenter le taux d'acceptation implique un risque plus élevé de ne pas rejeter les indépendances faibles et doit donc être utilisées avec prudence. L'indépendance de la taille d' un objet avec les autres paramètres influençant le throughput n'est pas nécessairement surprenant étant donné que nous ne considérons que de longues connexions. Cependant, l'absence d'une dépendance entre le DNS (*dns*) et l'IP de destination (*dstip*) est plus difficile à expliquer car il pourrait également venir d'un manque de granularité dans la représentation de l'adresse IP par son numéro d'AS. Ce dernier point n'a pas un impact important sur les résultats présentés

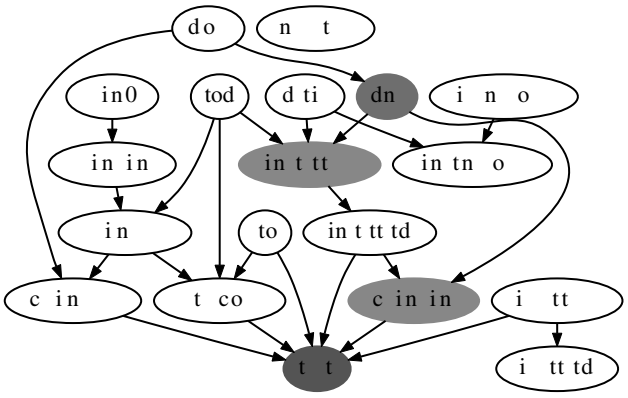


Figure F.9: Réseau Bayésien représentant le modèle causal des performances du CDN Akamai pour deux DNS: le DNS publique de Google le DNS du FAI local

dans cette étude.

Les paramètres capturant les pertes (*retrscore* et *RTO*) et les paramètres capturant les délais (*inetrttstd* et *isprttavg*) sont quatre des six parents directs du throughput. Cette observation est cohérente avec notre compréhension de TCP. Les parents supplémentaires du throughput étant les paramètre de congestion estimés au niveau du serveur (*cwinmin* et *cwinmax*).

Le fait qu'aucun des paramètres de la fenêtre de réception (*rwin*) ne soit un des parents du throughput (*tput*) n'a rien de surprenant. Les tests menés avec l'outil Intrabase [SBUK+05] diagnostiquent que les connexions ne sont jamais limitées par le récepteur.

Interventions

Intervention du le délai extérieur Notre intérêt résidant dans l'étude et la comparaison des mécanismes influençant le throughput des clients du CDN, dans cette section nous étudions l'effet qu'aurait une intervention consistant à modifier la distance du client au serveur. Plus précisément afin de comprendre le rôle et l'impact du choix d'un service DNS sur le choix d'un serveur CDN et sa distance jusqu'au client, nous étudions l'intervention qui correspond à rediriger un client utilisant le DNS de son FAI vers le serveur vers lequel il aurait été redirigé s'il avait utilisé le DNS de Google à la place. Ceci correspond à estimer

$$Pr(Tput|LDNS, do(RTT \sim f(RTT|do(GDNS)))), \tag{F.7}$$

où RTT représente *inetrttavg* (le délai extérieur $\simeq$ distance du FAI jusqu'au serveur), LDNS représente le DNS local (FAI) et GDNS représente le DNS Google.

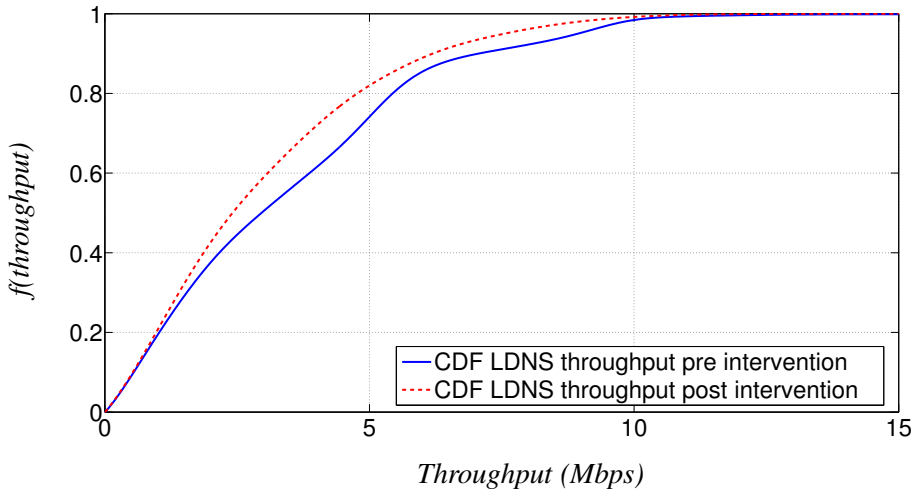


Figure F.10: Évolution de la distribution du throughput avant et après une intervention sur le délai externe pour les utilisateurs du DNS local (LDNS)

Avec l'utilisation du modèle de la Figure F.9, l'identification des variables tod et dns comme vérifiant le back-door criterion pour estimer l'effet sur le throughput d'une intervention sur le RTT, l'absence de back-door paths entre RTT et dns et le résultat de [Pea09, Section 4.2] qui nous permet d'estimer l'effet sur Y d'une intervention sur X, où cette intervention est opérer avec une probabilité $f^*(X|Z)$

$$f(y)_{|f^*(x|z)} = \int_{D_X}\int_{D_Z} f_{Y|do(X),Z}(y,x,z)f^*(x|z)f(z)\mathrm{d}x\mathrm{d}z, \tag{F.8}$$

on obtient le résultat suivant:

$$\begin{aligned} &f(tput|LDNS)_{f(rtt|do(GDNS))} = \\ &\int_{D_{RTT}}\int_{D_{TOD}} f(tput|do(rtt),LDNS,tod)f(tod)f(rtt|do(GDNS))P(GDNS) = \\ &\int_{D_{RTT}}\int_{D_{TOD}} f(tput|rtt,LDNS,tod)f(tod)f(rtt|GDNS)P(GDNS) \end{aligned} \tag{F.9}$$

Le résultat obtenu montre qu'une telle intervention ferait passer l'espérance du throughput pour les utilisateur de DNS local de **3.5Mbps** à **3.0Mbps**, soit une dégradation de leur performance de 14%. Par opposition ce résultat nous permet d'estimer le gain, en termes de throughput, dont bénéficie les utilisateur du DNS local lorsqu'ils sont redirigés vers des serveurs plus proches. La comparaison des CDFs avant et après intervention est présentée dans la Figure F.10.

Intervention sur la fenêtre minimale de congestion Une observation intéressante vient de la comparaison des performances que l'on observe entre

les utilisateurs du DNS local et les utilisateurs du DNS de Google. En effet ces derniers ont une espérance de leur throughput de 3.3 Mbps, ce qui correspond à seulement 6% de différence avec l'espérance du throughput des utilisateurs du DNS local (3.5 Mbps). Cette observation suggère qu'un autre mécanisme est présent et permet aux utilisateurs du DNS de Google d'obtenir des performances meilleures que celles escomptées aux vues de l'impact de la redirection sous optimal dont souffre les utilisateurs du DNS de Google. Ce mécanisme nous est suggéré par le modèle causal, Figure F.9, où le seul autre enfant du noeud dns est la fenêtre de congestion minimale ($cwinmin$). Afin de comprendre l'effet de ce mécanisme, nous adoptons la même approche que celle utilisée pour étudier l'effet du DNS sur le délai externe et le throughput. Nous estimons l'effet qu'aurait une intervention où l'on redirigerait les utilisateurs du DNS local vers des serveurs Akamai présentant une fenêtre de congestion minimale correspondant à celle observée par les utilisateurs du DNS de Google.

$$Pr(TPUT|LDNS, do(cwinmin \sim f(cwinmin|do(GDNS))) \tag{F.10}$$

En utilisant la même approche que précédemment, nous arrivons à estimer cette intervention et le résultat que nous obtenons suggère qu'une telle intervention ferait évoluer l'espérance du throughput des utilisateurs de DNS local de 3.5 Mbps à 4.6 Mbps, soit une augmentation de 32%, voir Figure F.11. Bien que notre modèle ne puisse pas identifier avec précision le mécanisme capturé par la variable $cwinmin$, ce résultat montre l'impact important de ce mécanisme sur les performances du CDN. Une explication possible, étant donné que les utilisateurs du DNS Google sont souvent redirigés vers des serveurs Akamai en dehors de l'AS du FAI, serait que les serveurs ont une configuration différente (software ou hardware) qui leur permet d'obtenir de meilleures performances.

Remarques Cette étude a été la dernière publiée au sein de notre travail. Elle montre le potentiel d'une approche causale et la possibilité de modéliser, comprendre et étudier des problèmes pouvant être relativement complexes. A la différence des études fait sur Mininet ou sur le serveur FTP, nous ne bornons plus à prédire des interventions atomiques mais bel et bien des scénarios d'interventions plus complexes qui nous permettent d'isoler et comprendre les différents mécanismes responsables des performances du système étudié. Bien que limité dans l'identification du mécanisme derrière le paramètre $cwinmin$, notre étude permet de mettre en avant un mécanisme qu'il aurait été très difficile d'identifier autrement.

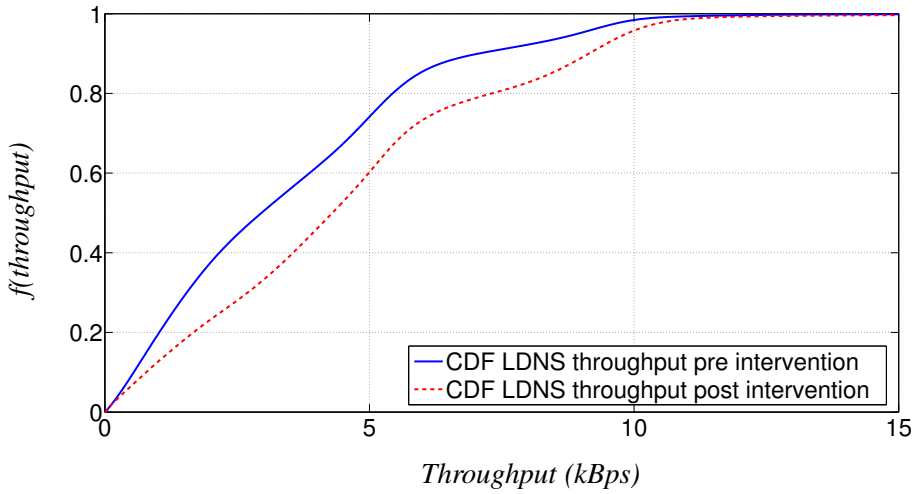


Figure F.11: Evolution de la distribution du throughput avant et après une intervention sur la fenêtre de congestion minimale des serveurs Akamai contactés par les clients du DNS local

F.7 Conclusions et directions futures

F.7.1 Conclusions

Notre travail s'est concentré sur l'application d'une approche causale pour l'étude des performances des réseaux de télécommunications. Bien que la théorie de la causalité soit présente depuis longtemps dans le milieu scientifique et utilisée dans de nombreux domaines tels que les sciences humaines, les sciences économiques où le domaine médical et biologique, les contraintes inhérentes à l'étude des réseaux de télécommunications font de son application une problématique relativement complexe.

Au sein de notre travail, nous avons pu identifier les hypothèses faites dans les domaines existants qui nous empêchent d'utiliser les travaux faits dans ces domaines dans le domaine de notre travail. En reprenant ces hypothèses et adaptant les solutions existantes aux contraintes des réseaux de télécommunications où les hypothèses de linéarité et normalité ne sont pas vérifiées, nous avons pu développer nos propres méthodes afin d'utiliser les avantages d'une approche causale dans l'étude et la compréhension des performances des réseaux de télécommunications.

L'utilisation de réseaux Bayésien comme modèle causaux et les travaux faits par Judea Pearl [Pea09] pour tirer profit de cette modélisation des dépendances causales afin de prédire la façon dont un système réagit lorsque l'on modifie les mécanismes dictant son fonctionnement, nous ont permis de mettre

en place une série de méthodes et d'étapes qui simplifie grandement l'étude de systèmes aussi complexes que les réseaux de télécommunications.

Après avoir validé notre solution dans un milieu où nos prédictions pouvaient être vérifiées, nous avons pu appliquer avec succès une approche causale afin d'étudier les performances de l'application FTP puis, plus complexe, l'impact du service DNS sur les performances d'un CDN. Ces différentes études ont mis en avant les avantages et la supériorité d'une approche causale sur une approche "classique" où l'absence de modèle causal et l'impossibilité d'estimer l'effet des différents mécanismes sur les performances d'un système rend l'étude d'un tel système relativement limitée.

La majeure partie de notre travail s'est faite sur l'adaptation des méthodes existantes aux contraintes des réseaux de télécommunications. Bien que cette partie ait été relativement complexe et laborieuse, les résultats que nous présentons montrent que ces démarches étaient nécessaires et ont porté leur fruits. De ce point de vue, notre travail a mis en avant le contraste historique entre théorie et pratique ce qui nous a permis de tirer plusieurs leçons de nos différentes études qui sont résumées ci-dessous

- **Know your methods:** Lorsque l'on envisage d'utiliser une approche qui n'a pas été largement utilisée dans un nouveau domaine, il est important de prendre en compte toutes les hypothèses et restrictions des méthodes qu'elle comprend. Notamment lors de l'implémentation des méthodes et leur applications à des cas réelles, si un effort n'a pas été fait en aval pour sécuriser l'utilisation d'une approche pour étudier un problème donné, au moment d'obtenir les résultats de l'étude d'un système donné il devient très difficile de discerner entre une erreur dans la représentation du problème que l'on tente de résoudre et une erreur dans les méthodes utilisées.

- **Know your data:** Il est commun de citer le problème du "Garbage In, Garbage Out" et notre travail n'y fait pas défaut. Il est donc important, avant de commencer l'étude causale d'un système donné, de dédier suffisamment de temps dans l'élaboration du scénario, l'acquisition des données et d'obtenir une visualisation globale des données qui vont être manipulées. Il arrive souvent que les données utilisées soient incomplètes, biaisées où qu'elles contiennent des erreurs faites lors de leur estimation ou acquisition. Ce qui nous amène au dernier point.

- **Do not under estimate pre-processing:** Il est important de noter que, lors d'une étude causale, plus de 75% du temps est utilisé dans la définition du système, l'acquisition des données et la transformation de ces données en un "input" qui convient à une étude causale. Tout le temps passé dans cette phase représente autant de temps qui sera gagné lors de l'étude du modèle causal et des effets causaux dans une étape

postérieure. En général, une étude causale n'est pas nécessairement linéaire et il est souvent nécessaire de revenir en arrière pour ajouter certains paramètres, enlever certaines mesures ou en ajouter. Ces étapes représentent un temps considérable et le temps passé au pré formattage des données permet, dans une certaine mesure, de limiter le temps passé dans la recherche des mécanismes manquant dans le modèle ou bien dans la recherche d'erreurs possibles au sein des données.

F.7.2 Directions futures

Notre travail a permis de mettre en place de nombreuses méthodes facilitant l'étude causale de systèmes allant au-delà des réseaux de communications. Cependant certains problèmes sont encore non résolus et nous proposons ici certaines directions pour poursuivre notre travail.

Tout d'abord, dû à notre volonté de ne faire aucune hypothèse quant aux distributions des paramètres considérés dans nos modèles ainsi que la nature de leurs dépendances, concevoir un critère de sélection pour nos graphes causaux est relativement complexe. Une possibilité serait d'utiliser les p-values des tests d'indépendances opérés lors de l'inférence de nos modèles causaux qui nous donnerait une certaine confiance dans la présence ou l'absence d'une relation causale directe entre deux paramètres. Bien que cette approche semble prometteuse elle n'en reste pas moins complexes car le nombre de tests à considérer pour une dépendance donné est important et définir un critère global devient donc relativement complexe, chaque choix (minimum, maximum, moyenne, moyenne pondérée) ayant un impact très important sur le critère choisi.

Une limitation importante dans notre approche vient de notre modélisation des marginales (distributions des paramètres considérés dans nos modèles). En effet, l'utilisation de kernels ou histogrammes permet de nous affranchir d'une quelconque hypothèse sur leur distributions. Cependant, elle limite aussi le nombre de prédictions que l'on peut faire. Par exemple, si un taux de perte de 15% n'rdy jamais observé, il nous est impossible d'estimer comment le système réagira dans une telle situation. La possibilité d'extrapoler les distributions des marginales en dehors des domaines où les paramètres ont été observés serait donc très intéressant. Cependant les études préliminaires que nous avons faites nous ont montré que des modèles paramétriques, bien que pouvant capturer les distributions des paramètres dans leurs domaines d'observations, deviennent inutilisable pour estimer la probabilité de certaines valeurs qui n'ont pas pu être observées durant notre phase d'observation du système.

Enfin, les méthodes que nous avons développées au sein de notre travail permettent d'étudier de nombreux systèmes qui pourrait tirer profit d'une approche causale.

C'est en particulier le cas du domaine de la sécurité et la cyber insurance où la possibilité de prédire quand une machine pourrait être infectée serait un avantage incontestable dans la conception d'assurances pour des compagnies ou particuliers.

Un autre domaine, en plein essor aujourd'hui, est celui du cloud computing où suffisamment de connaissances sont présentes pour connaitre les paramètres qui influencent les performances d'un cluster mais où le nombre de ces paramètres et leurs interdépendances rend le paramétrage d'un cluster relativement complexe. Dans un tel scénario, obtenir un modèle du rôle de chaque paramètre dans les performances d'un cluster serait très utile et une approche causale permettrait de choisir, pour un budget donné, quelle la meilleure configuration pour atteindre un niveau de performance souhaité.

Dans ces deux domaines, des projets ont été initiés. Aucun résultat ne peut être présenté à ce stade des études.

Finalement, notre travail a donné lieu à trois publications au sein de conférence [HBL14a, HBL14b, HBL+15b] et un article dans le journal ACM-TIST pour la special issue sur la causalité qui sera publié en 2016 [HBL15a].

Appendix G

Glossary

G.1 Acronyms and Abbreviations

ADSL Asynchronous Digital Suscriber Line 14
ARPA Advance Research Project Agency 5
ARPANET Advance Research Project Agency Network 5
BNT Bayes Net Toolbox 32
CDF Cumulative Distribution Function 70
CPU Central Processing Unit 63
DAG Directed Acyclic Graph 20
DNS Domain Name System 72
FTP File Transfer Protocol 95
FTTH Fiber To The Home 14
GDNS Google DNS 128
HSIC Hilbert Schmidt Independence Criterion 54
HTTP Hyper Text Transfer Protocol 3
IP Internet Protocol 6
ISPs Internet Service Providers 14
ITU International Telecommunication Union 14
KCI Kernel Conditional Independence 54
LDNS Local DNS 128
LFN Long Fat Network 13
MSS Maximum Segement Size 53

PAG Partial Ancestral Graph 20
PDAG Partially Directed Acyclic Graph 24
PDF Probability Distribution Function 70
PSN Packet Switched Network 4
QoE Quality of Experience 15
RKHS Reproducing Kernel Hilbert Space 54
RTT Round Trip Time 53
SEM Structural Equation Model 19
SMTP Simple Mail Transfer Protocol 6
TCP Transmission Control Protocol 7
UDP User Datagram Protocol 6
WAN Weakly Additive Noise 23
LAN Local Area Network 14

List of Figures

1.1 A telecommunications network 4
1.2 OSI stack 6
1.3 TCP Windows 11
1.4 TCP Sliding Window 12
1.5 Example of a Bayesian network 21
1.6 Bayesian network representing an intervention on X 21
1.7 Illustration of d-separation 27
1.8 Do calculus graphs 30

3.1 KCI completion time 61
3.2 Polynomial estimate of KCI completion time 62
3.3 Polynomial estimate of KCI completion time log 62
3.4 Bootstrap and completion time 65
3.5 Four variables model 65
3.6 Size vs power first study 67
3.7 Size vs power second study 69
3.8 FTP parameter histograms 71
3.9 Example of a T-copula 72
3.10 Example of a Gaussian copula 73

4.1 Dropbox upload 79
4.2 Causal models of the first Dropbox study, PC algorithm 83
4.3 Causal models of the first Dropbox study, kPC algorithm 83
4.4 Causal models of the second Dropbox study, PC algorithm 88

4.5 Causal models of the second Dropbox study, kPC algorithm . . . 89

5.1 Emulated network using Mininet . . . 96
5.2 Causal model emulated traffic . . . 100
5.3 IC algorithm model, emulated network . . . 102
5.4 kPC algorithm model, emulated network . . . 102
5.5 PC Fisher model, emulated network . . . 103
5.6 PC Fisher log model, emulated network . . . 104
5.7 FCI algorithm model, emulated network . . . 105
5.8 Causal model, randomized emulated network . . . 106
5.9 Throughput post intervention, emulated network . . . 108
5.10 Causal model of FTP traffic . . . 112
5.11 IC algorithm, model of FTP traffic . . . 113
5.12 PC algorithm, Fisher, model of FTP traffic . . . 114
5.13 PC algorithm, Fisher, log transform, model of FTP traffic . . . 115
5.14 FCI algorithm, Fisher, model of FTP traffic . . . 115
5.15 kPC algorithm, model of FTP traffic . . . 116
5.16 Throughput post RTT intervention, FTP traffic . . . 118
5.17 Throughput post loss intervention, FTP traffic . . . 119
5.18 Receiver window variations in the FTP dataset . . . 119

6.1 ISP DNS and CDN . . . 126
6.2 Google DNS and CDN . . . 126
6.3 DNS impact, causal model . . . 132
6.4 Throughput vs receiver window, DNS study . . . 135
6.5 External RTT histograms, DNS study . . . 137
6.6 Intervention on LDNS delay . . . 138
6.7 Intervention on LDNS cwinmin . . . 140
6.8 Cwinmin histograms, DNS study . . . 141
6.9 Destination IP DNS dataset . . . 145

A.1 Unconditional independence tests . . . 164

A.2 Different graphical configurations for orienting V-structure 165
A.3 Causal model of 4 variables 170

B.1 Causal model of DNS traffic performance 174
B.2 Histogram inetrttavg per DNS 175
B.3 Histogram cwinmin per DNS 176
B.4 Artificial dataset dependencies 180
B.5 Distribution of the different parameters for both values of X_2 . . . 182
B.6 Causal Model of the first dataset 182
B.7 PDF $X \mid X_2$ first dataset 186
B.8 PDF $X \mid X_2$ second dataset 186
B.9 PDF $X \mid X_2$ third dataset 188

C.1 Comparison of the original and estimated PDFs 196
C.2 Comparison of the different estimates 198
C.3 Throughput CDN distribution modeling 200
C.4 Normal components of throughput PDF, removing 10% 201
C.5 Weibull components of throughput PDF, removing 10% 201
C.6 Throughput PDF, removing 10% 202
C.7 Weibull components of throughput PDF, removing 20% 203
C.8 Throughput PDF, removing 20% 203
C.9 Weibull components of throughput PDF, removing 30% 204
C.10 Throughput PDF, removing 30% 204
C.11 Throughput PDF, removing 30% 205

D.1 From copula modeling to multivariate distribution 208
D.2 G-copula 210
D.3 T-copula 212

F.1 Exemple de V-structure 228
F.2 Exemples de paramétres 233
F.3 Emulation d'un réseau avec Mininet 236
F.4 Modèle causal pour le trafic emulé 237

F.5 Throughput post intervention, réseau émulé 238
F.6 Modèle causal du trafic FTP 241
F.7 Throughput post RTT intervention, FTP trafic 241
F.8 Throughput post loss intervention, FTP trafic 242
F.9 DNS impact, causal model 246
F.10 Intervention sur le délai du DNS local 247
F.11 Intervention sur cwinmin LDNS 249

List of Tables

3.1 Impact of bootstrap parameters on completion time 64

4.1 First Dropbox study 81
4.2 Second Dropbox study 87

5.1 Emulated network dataset 98
5.2 Randomized emulated network dataset 106
5.3 FTP traffic dataset 110

6.1 DNS dataset, both DNS 131
6.2 DNS dataset, per DNS 131

A.1 Testing for conditional independence, configuration 1 166
A.2 Testing for conditional independence, configuration 2 167
A.3 Testing for conditional independence, configuration 3 168
A.4 Testing for conditional independence, configuration 1 169
A.5 Testing for conditional independence, configuration 2 169
A.6 Testing for conditional independence, configuration 3 169
A.7 Test KCI categorical 171

B.1 Parameterization effect, ideal scenario, first intervention 183
B.2 Parameterization effect, ideal scenario, second intervention . . . 184
B.3 Parameterization effect when removing few samples, first intervention187
B.4 Parameterization effect when removing few samples, second intervention187
B.5 Parameterization effect when removing many samples, first intervention189

B.6 Parameterization effect when removing many samples, second intervention189

C.1 Rebmix Weibull 14 components for estimating the 10 components Weibull mixture198

F.1 Dataset du réseau émulé 236
F.2 Dataset du trafic FTP 239
F.3 DNS dataset 243
F.4 Dataset par DNS 243

List of Publications 2011—2015

Conference and Journal Publications

1. Hours, Hadrien; Biersack, Ernst; Loiseau, Patrick; A causal study of an emulated network, 10ème Atelier en Evaluation de Performances, 11-13 juin 2014, Sophia-Antipolis, France

2. Hours, Hadrien; Biersack, Ernst W; Loiseau, Patrick; Causal study of network performance ALGOTEL 2014, 16èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, June 3-6, 2014, Le-Bois-Plage-en-Ré, France

3. Hours, Hadrien; Biersack, Ernst; Loiseau, Patrick, A causal approach to the study of telecommunication network performance, ACM-TIST Accepted in May 2015, to appear in 2016.

4. Hours, Hadrien; Biersack, Ernst; Loiseau, Patrick, Alessandro Finamore, Marco Mellia, A Study of the Impact of DNS Resolvers on Performance Using a Causal Approach, ITC27, September 2015 (Accepted in May 2015)

Teaching

Introduction lecture on Probability as part of Patrick Loiseau's course *Statistical data analysis*, EURECOM, Fall 2014.

Bibliography

[AB01] D. W. K. Andrews and M. Buchinsky, *Evaluation of a three-step method for choosing the number of bootstrap repetitions*, Journal of Econometrics **103** (2001), no. 1-2, 345–386.

[AJPMI11] J. Audrius, L. Jukka-Pekka, H. Matti, and W. Alf Inge, *An empirical study of NetEm Network Emulation functionalities.*, ICCCN, IEEE, 2011, pp. 1–6.

[AMSU10] B. Ager, W. Mühlbauer, G. Smaragdakis, and S. Uhlig, *Comparing DNS resolvers in the wild*, Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC '10, ACM, 2010, pp. 15–21.

[And84] T. W. Anderson, *An introduction to multivariate statistical analysis*, second ed., Wiley, New York, NY, 1984.

[ARS13] H. A. Alzoubi, M. Rabinovich, and O. Spatscheck, *The anatomy of LDNS clusters: Findings and implications for Web Content Delivery*, Proceedings of the 22Nd International Conference on World Wide Web (Republic and Canton of Geneva, Switzerland), WWW '13, International World Wide Web Conferences Steering Committee, 2013, pp. 83–94.

[bar81] *Hyperbolic distributions and ramifications: contributions to theory and application*, Statistical Distributions in Scientific Work **4** (1981).

[Bau09] M. Baumgartner, *Uncovering deterministic causal structures: a boolean approach.*, Synthese **170** (2009), no. 1, 71–96.

[BT12] G. Borboudakis and I. Tsamardinos, *Incorporating Causal Prior Knowledge as Path-Constraints in Bayesian Networks and Maximal Ancestral Graphs*, ICML, June 2012.

[CAA+08] R. Chou, N. Aronson, D. Atkins, A. S. Ismaila, P. Santaguida, D. H. Smith, E. Whitlock, T. J. Wilt, and D. Moher, *Assessing Harms*

When Comparing Medical Interventions, Methods Guide for Effectiveness and Comparative Effectiveness Reviews, Agency for Healthcare Research and Quality, 2008, pp. 1–23.

[CAR13] T. Callahan, M. Allman, and M. Rabinovich, *On modern DNS behavior and properties*, SIGCOMM Comput. Commun. Rev. **43** (2013), no. 3, 7–15.

[CBS12] I. Canadi, P. Barford, and J. Sommers, *Revisiting broadband performance*, Proceedings of the 2012 ACM Conference on Internet Measurement Conference, 2012, pp. 273–286.

[CG08] T. Chu and C. Glymour, *Search for additive nonlinear time series causal models*, J. Mach. Learn. Res. **9** (2008), 967–991.

[CJAZ11] Y. Chen, S. Jain, V. K. Adhikari, and Z. Zhang, *Characterizing roles of front-end servers in end-to-end performance of Dynamic Content Distribution*, Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, 2011, pp. 559–568.

[CM12] D. Colombo and M. H. Maathuis, *Order-independent constraint-based causal structure learning*, ArXiv e-prints (2012).

[CMKR12] D. Colombo, M. H. Maathuis, M. Kalisch, and T. S. Richardson, *Learning high-dimensional directed acyclic graphs with latent and selection variables*, The Annals of Statistics **40** (2012), no. 1, 294–321.

[CWRZ07] X. Chen, H. Wang, S. Ren, and X. Zhang, *Maintaining strong cache consistency for the Domain Name System*, IEEE Trans. on Knowl. and Data Eng. **19** (2007), no. 8, 1057–1071.

[Dar09] A. Darwich, *Modeling and reasoning with Bayesian networks*, 1st ed., Cambridge University Press, New York, NY, USA, 2009.

[dCVM+11] J.A.R.P. de Carvalho, H. Veiga, N. Marques, C.F.F.R. Pacheco, and A.D. Reis, *Performance measurements of ieee 802.11 b, g laboratory wep and wpa point-to-point links using TCP, UDP and FTP*, Applied Electronics (AE), 2011 International Conference on, Sept 2011, pp. 1–6.

[DG08] J. Dean and S. Ghemawat, *Mapreduce: Simplified data processing on large clusters*, Commun. ACM **51** (2008), no. 1, 107–113.

[DGK+15] A. Datta, D. Garg, D. Kaynar, D. Sharma, and A. Sinha, *Programs as Actual Causes: A Building Block for Accountability*, ArXiv e-prints (2015).

[DH97] A. C. Davison and D.V. Hinkley, *Bootstrap methods and their application*, Cambridge University Press, 1997.

[DKCDS12] R. M. Daniel, M. G Kenward, S. N Cousens, and B. L De Stavola, *Using causal diagrams to guide analysis in missing data problems*, Statistical Methods in Medical Research **21** (2012), no. 3, 243–256.

[DMMM+12] I. Drago, M. Mellia, M. M. Munafo, A. Sperotto, R. Sadre, and A. Pras, *Inside dropbox: Understanding personal cloud storage services*, Proceedings of the 2012 ACM Conference on Internet Measurement Conference (New York, NY, USA), ACM, 2012, pp. 481–494.

[DRC+10] N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin, *An argument for increasing TCP's initial congestion window*, SIGCOMM Comput. Commun. Rev. **40** (2010), no. 3, 26–33.

[DSA03] Lindskog F. Daul S., De Giorgi E. and Mcneil A., *The grouped t-copula and with an application to credit risk*, RISK **16** (2003), 73–76.

[DST+12] M. Dhawan, J. Samuel, R. Teixeira, C. Kreibich, M. Allman, N. Weaver, and V. Paxson, *Fathom: A browser-based network measurement platform*, Proceedings of the 2012 ACM Conference on Internet Measurement Conference, IMC '12, 2012, pp. 73–86.

[Dun75] O.D. Duncan, *Introduction to Structural Equation Models*, Academic Press, New York, USA, 1975.

[ENUK06] T. En-Najjary and G. Urvoy-Keller, *Pprate: A passive capacity estimation tool*, E2EMON (Al-Shaer E., Pras A., and Brownlee N., eds.), IEEE, 2006, pp. 82–89.

[FA13] N.C. Fofack and S. Alouf, *Modeling modern DNS caches*, Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools (ICST, Brussels, Belgium, Belgium), ValueTools '13, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2013, pp. 184–193.

[FBJ04] K Fukumizu, F. R. Bach, and M. I. Jordan, *Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces*, J. Mach. Learn. Res. **5** (2004), 73–99.

[FGXS08] K. Fukumizu, A. Gretton, Sun X., and B Scholkopf, *Kernel measures of conditional dependence*, Advances in Neural Information Processing Systems 20 (J.C. Platt, D. Koller, Y. Singer, and S.T. Roweis, eds.), Curran Associates, Inc., 2008, pp. 489–496.

[FMM+11] A. Finamore, M. Mellia, M. Meo, M.M. Munafo, and D. Rossi, *Experiences of internet traffic monitoring with tstat*, Network, IEEE **25** (2011), no. 3.

[Fre09] FreeBSD, *Freebsd file formats manual*, online, http://www.freebsd.org/cgi/man.cgi?query=tar, 2009.

[FSM14] K. Fukuda, S. Sato, and T. Mitamura, *Towards evaluation of DNS server selection with geodesic distance.*, NOMS, 2014, pp. 1–8.

[FSS02] P. Ford, A. Shelest, and N. Srinivas, *Method for automatic tuning of TCP receive window*, August 15 2002, US Patent App. 09/736,988.

[fuk08] *Kernel measures of conditional dependence*, NIPS **20** (2008), 489–496.

[GBSS05] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, *Measuring statistical dependence with hilbert-schmidt norms*, Proceedings of the 16th International Conference on Algorithmic Learning Theory, ALT'05, Springer-Verlag, 2005, pp. 63–77.

[GE03] I. Guyon and A. Elisseeff, *An introduction to variable and feature selection*, J. Mach. Learn. Res. **3** (2003), 1157–1182.

[GFT+08] A. Gretton, K. Fukumizu, CH. Teo, L. Song, B. Schölkopf, and AJ. Smola, *A kernel statistical test of independence*, Advances in neural information processing systems 20, September 2008, pp. 585–592.

[GHS+05] A. Gretton, R. Herbrich, A. Smola, O. Bousquet, and B. Schölkopf, *Kernel methods for measuring independence*, J. Mach. Learn. Res. **6** (2005), 2075–2129.

[GKH+08] A. Gretton, Fukumizu K., Teo C. H., Le S., B. Scholkopf, and Smola A. J., *A kernel statistical test of independence*, Advances in Neural Information Processing Systems 20 (J.C. Platt, D. Koller, Y. Singer, and S.T. Roweis, eds.), Curran Associates, Inc., 2008, pp. 585–592.

[GM04] A. Gelman and X. Meng, *Applied Bayesian Modeling and Causal Inference from Incomplete-Data Perspectives: Wiley Series in Probability and Statistics*, Wiley, Chichester, 2004.

[GPR09] A. Gretton, Spirtes P., and Tillman R.E., *Nonlinear directed acyclic structure learning with weakly additive noise models*, Advances in Neural Information Processing Systems 22, Curran Associates, Inc., 2009, pp. 1847–1855.

[HA12] S. Haryadi and R. Andina, *QoS measurement of file transfer protocol services in 3G networks using aggregation method*, Telecommunication Systems, Services, and Applications (TSSA), 2012 7th International Conference on, Oct 2012, pp. 107–110.

[Haa43] T. Haavelmo, *The Statistical Implications of a System of Simultaneous Equations*, Econometrica **11** (1943), 1–12.

[HB12] A. Hauser and P. Bühlmann, *Characterization and greedy learning of interventional markov equivalence classes of directed acyclic graphs*, J. Mach. Learn. Res. **13** (2012), no. 1, 2409–2464.

[HB14] A. Hauser and P. Bühlmann, *Two optimal strategies for active learning of causal models from interventional data*, Int. J. Approx. Reasoning **55** (2014), no. 4, 926–939.

[HBL14a] H. Hours, E. Biersack, and P. Loiseau, *A causal study of an emulated network*, 10ème Atelier en Evaluation de Performances, Sophia-Antipolis, France, Jun. 2014.

[HBL14b] ———, *Causal study of network performance*, ALGOTEL 2014, 16èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, Le-Bois-Plage-en-Ré, France, Jun. 2014.

[HBL15a] H. Hours, E. Biersack, and P. Loiseau, *A causal approach to the study of TCP performance*, ACM Transactions on Intelligent Systems and Technology, 2015 (2015).

[HBL+15b] H. Hours, E. Biersack, P. Loiseau, A. Finamore, and M. Mellia, *A study of the impact of DNS resolvers on performance using a causal approach*, ITC 2015, 27th International Teletraffic Congress (ITC 27) September 8-10, 2015, Ghent, Belgium (Ghent, BELGIUM), 09 2015.

[HH09] P. O. Hoyer and A. Hyttinen, *Bayesian Discovery of Linear Acyclic Causal Models*, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (Arlington, Virginia, United States), UAI '09, AUAI Press, 2009, pp. 240–248.

[HHJ+12] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, *Reproducible network experiments using container-based em-*

ulation, CoNEXT '12 (New York, NY, USA), ACM, 2012, pp. 253–264.

[HKZ+15] M. Hamstra, H. Karau, M. Zaharia, A. Konwinski, and P. Wendell, *Learning spark: Lightning-fast big data analytics*, O'Reilly Media, Incorporated, 2015.

[HMLG11] C. Huang, D. A. Maltz, J. Li, and A. G. Greenberg, *Public DNS system and global traffic management.*, INFOCOM, IEEE, 2011, pp. 2615–2623.

[HRX08] S Ha, I Rhee, and L Xu, *Cubic: A new TCP-friendly high-speed TCP variant*, SIGOPS Oper. Syst. Rev. **42** (2008), no. 5, 64–74.

[Hua10] T. Huang, *Testing conditional independence using maximal nonlinear conditional correlation*, The Annals of Statistics **38** (2010), no. 4, 2047–2091.

[IP11] S. Ihm and V. S. Pai, *Towards understanding modern web traffic*, Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, 2011, pp. 295–312.

[J.80] Daudin. J. J., *Partial association measures and an application to qualitative regression*, Biometrika **67** (1980), 581–590.

[JDHR10] P. Jaworski, F. Durante, W.K. Härdle, and T. Rychlik, *Copula theory and its applications*, Lecture Notes in Statistics, Springer, 2010.

[JSBM02] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, *DNS performance and the effectiveness of caching*, IEEE/ACM Trans. Netw. **10** (2002), no. 5, 589–603.

[JTCT11] D. Joumblatt, R. Teixeira, J. Chandrashekar, and N. Taft, *Performance of networked applications: The challenges in capturing the user's perception*, Proceedings of the First ACM SIGCOMM Workshop on Measurements Up the Stack, W-MUST '11, 2011, pp. 37–42.

[JZQM12] Z. Jiao, L. Zou, N. Qian, and Z. Ma, *Effective connectivity analysis of fMRI time-series based on granger causality and complex network*, Biomedical Engineering and Informatics (BMEI), 2012 5th International Conference on, Oct 2012, pp. 1367–1370.

[KMC+12] M. Kalisch, M. Mächler, D. Colombo, M. H. Maathuis, and P. Bühlmann, *Causal inference using graphical models with the R package pcalg*, Journal of Statistical Software **47** (2012), no. 11, 1–26.

[Koi06] M. Koivisto, *Advances in exact Bayesian structure discovery in Bayesian networks.*, UAI, 2006.

[KR09] J. F. Kurose and K. W. Ross, *Computer networking: A top-down approach*, 5th ed., Addison-Wesley Publishing Company, USA, 2009.

[KS04] M. Koivisto and K. Sood, *Exact Bayesian structure discovery in Bayesian networks*, J. Mach. Learn. Res. **5** (2004), 549–573.

[LA04] S. Ladha and P.D. Amer, *Improving file transfers using sctp multistreaming*, Performance, Computing, and Communications, 2004 IEEE International Conference on, 2004, pp. 513–522.

[LGBVBP10] P. Loiseau, P. Gonçalves, J. Barral, and P. Vicat-Blanc Primet, *Modeling TCP throughput: an elaborated large-deviations-based model and its empirical validation*, In Proceedings of IFIP Performance, 2010.

[LHM10] B. Lantz, B. Heller, and N. McKeown, *A network in a laptop: Rapid prototyping for software-defined networks*, Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (New York, NY, USA), Hotnets-IX, ACM, 2010, pp. 19:1–19:6.

[LZZ+10] Z. Li, M. Zhang, Z. Zhu, Y. Chen, A. Greenberg, and Y. Wang, *Webprophet: Automating performance prediction for web services*, Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, NSDI'10, 2010, pp. 10–10.

[Mar05] D. Margaritis, *Distribution-free learning of bayesian network structure in continuous domains*, Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference„ 2005, pp. 825–830.

[Mas51] F. J. Massey, *The Kolmogorov-Smirnov test for goodness of fit*, Journal of the American Statistical Association **46** (1951), no. 253, 68–78.

[Max02] D. C. Maxwell, *Learning equivalence classes of bayesian-network structures*, J. Mach. Learn. Res. **2** (2002), 445–498.

[Mee95] C. Meek, *Causal inference and causal explanation with background knowledge*, Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (San Francisco, CA, USA), UAI'95, Morgan Kaufmann Publishers Inc., 1995, pp. 403–410.

[MLCN05] M. Mellia, R. Lo Cigno, and F. Neri, *Measuring ip and TCP behavior on edge nodes with tstat*, Comput. Netw. **47** (2005), no. 1, 1–21.

[MSBZ07] M. Mirza, J. Sommers, P. Barford, and X. Zhu, *A machine learning approach to TCP throughput prediction*, SIGMETRICS '07 (New York, NY, USA), ACM, 2007, pp. 97–108.

[Mur01] K. P. Murphy, *The Bayes Net Toolbox for MATLAB*, Computing Science and Statistics **33** (2001), 2001.

[NF11] M. Nagode and M. Fajdiga, *The rebmix algorithm for the univariate finite mixture estimation*, Communications in Statistics - Theory and Methods **40** (2011), no. 5, 876–892.

[OSRB12] J. S. Otto, M. A. Sánchez, J. P. Rula, and F. E. Bustamante, *Content delivery and the natural evolution of DNS: Remote DNS trends, performance issues and alternative solutions*, Proceedings of the 2012 ACM Conference on Internet Measurement Conference, IMC '12, ACM, 2012, pp. 523–536.

[PAS+04] J. Pang, A. Akella, . Shaikh, B. Krishnamurthy, and S. Seshan, *On the responsiveness of DNS-based network control*, Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement (New York, NY, USA), IMC '04, ACM, 2004, pp. 21–26.

[PCK06] M. Pitt, D. Chan, and R. Kohn, *Efficient bayesian inference for gaussian copula regression models*, Biometrika **93** (2006), no. 3, 537–554.

[Pea09] J. Pearl, *Causality: Models, Reasoning and Inference*, Cambridge University Press, New York, NY, USA, 2009.

[Pea13] ———, *Direct and indirect effects*, CoRR **abs/1301.2300** (2013).

[PFTK98] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, *Modeling TCP throughput: a simple model and its empirical validation*, SIGCOMM Comput. Commun. Rev. **28** (1998), no. 4, 303–314.

[PJS11] J. Peters, D. Janzing, and B. Scholkopf, *Causal inference on discrete data using additive noise models*, IEEE Transactions on Pattern Analysis and Machine Intelligence **33** (2011), no. 12.

[PMW10] N Parvez, A Mahanti, and C Williamson, *An analytic throughput model for TCP newreno*, IEEE/ACM Trans. Netw. **18** (2010), no. 2, 448–461.

[PSN+09] L. Ping, H. Subramoni, S. Narravula, A. Mamidala, and D.K. Panda, *Designing efficient ftp mechanisms for high performance data-transfer over infiniband*, Parallel Processing, 2009. ICPP '09. International Conference on, Sept 2009, pp. 156–163.

[Ram14] J. Ramsey, *A scalable conditional independence test for nonlinear, non-gaussian data*, CoRR **abs/1401.5031** (2014).

[RHH+09] J. D. Ramsey, S. J. Hanson, C. Hanson, Y. O. Halchenko, R. A. Poldrack, and C. Glymour, *Six problems for causal inference from fMRI*, Neuroimage (2009).

[Ric13] T. Richardson, *A discovery algorithm for directed cyclis graphs*, CoRR **abs/1302.3599** (2013).

[RJN13] A. Rau, F. Jaffrezic, and G. Nuel, *Joint estimation of causal effects from observational and intervention gene expression data*, BMC Systems Biology **7** (2013), no. 1, 111.

[Rub74] D. B. Rubin, *Estimating Causal Effects of Treatments in Randomized and Nonrandomized Studies*, Journal of Educational Psychology **66** (1974), no. 5, 688–701.

[RYS09] J. Ravi, Z. Yu, and W. Shi, *A survey on dynamic web content generation and delivery techniques*, J. Netw. Comput. Appl. **32** (2009), no. 5, 943–960.

[SA05] Demarta S. and McNeil A., *The t copula and related copulas*, INTERNATIONAL STATISTICAL REVIEW **73** (2005), 111–129.

[SBUK+05] M. Siekkinen, E.W. Biersack, G. Urvoy-Keller, V. Goebel, and T. Plagemann, *Intrabase: integrated traffic analysis based on a database management system.*, E2EMON, IEEE, 2005, pp. 32–46.

[SG91] P. Spirtes and C. Glymour, *An Algorithm for Fast Recovery of Sparse Causal Graphs*, Social Science Computer Review **9** (1991), 62–72.

[SGS01] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*, second ed., The MIT Press, Cambridge, MA, USA, January 2001.

[SGS05] R. Silva, C. Glymour, and P. Spirtes, *Learning the structure of linear latent variable models*, Journal of Machine Learning Research **7** (2005).

[SJKC06] S. Sohail, S.K. Jha, S.S. Kanhere, and Chun T. C., *QoS driven parallelization of resources to reduce file download delay*, Parallel

and Distributed Systems, IEEE Transactions on **17** (2006), no. 10, 1204–1215.

[SJSF07] X. Sun, D. Janzing, B. Schölkopf, and K. Fukumizu, *A kernel-based causal learning algorithm*, ICML '07 (New York, NY, USA), ACM, 2007, pp. 855–862.

[SK02] P. Sarolahti and A. Kuznetsov, *Congestion control in linux TCP*, Proceedings of the FREENIX Track: 2002 USENIX Annual Technical Conference, 2002, pp. 49–62.

[SMR95] P. Spirtes, C. Meek, and T. Richardson, *Causal inference in the presence of latent variables and selection bias*, Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (San Francisco, CA, USA), UAI'95, Morgan Kaufmann Publishers Inc., 1995, pp. 499–506.

[Son09] K. Song, *Testing conditional independence via rosenblatt transforms*, The Annals of Statistics **37** (2009), no. 6B, 4011–4045.

[SRKS07] S.M.H. Shah, A.u. Rehman, A.N. Khan, and M.A. Shah, *TCP throughput estimation: A new neural networks model*, Emerging Technologies, 2007. ICET 2007. International Conference on, nov. 2007, pp. 94–98.

[STWB03] M. Steyvers, J. B. Tenenbaum, E. Wagenmakers, and B. Blum, *Inferring causal networks from observations and interventions*, Cognitive Science **27** (2003), no. 3, 453–489.

[SUKBC08] M. Siekkinen, G. Urvoy-Keller, E. W. Biersack, and D. Collange, *A root cause analysis toolkit for TCP*, Comput. Netw. **52** (2008), no. 9, 1846–1858.

[SW07] L. Su and H. White, *A consistent characteristic function-based test for conditional independence*, Journal of Econometrics **141** (2007), no. 2, 807 – 834.

[SW08] _____, *A nonparametric hellinger metric test for conditional independence*, Econometric Theory **24** (2008), no. 4, 829–864.

[TGS09] R. Tillman, A. Gretton, and P. Spirtes, *Nonlinear directed acyclic structure learning with weakly additive noise models*, In Advances in Neural Information Processing Systems (NIPS) (Vancouver, Canada), December 2009.

[Tip00] M. E. Tipping, *The relevance vector machine*, 2000.

[Tip01] M. E. Tipping., *Sparse Bayesian Learning and the Relevance Vector Machine*, J. Mach. Learn. Res. **1** (2001), 211–244.

[TJA12] B Thomas, R Jurdak, and I Atkinson, *Spdying up the web*, Commun. ACM **55** (2012), no. 12, 64–73.

[TMFA09] M. Tariq, M. Motiwala, N. Feamster, and M. Ammar, *Detecting network neutrality violations with causal inference.*, CoNEXT, ACM, 2009, pp. 289–300.

[TZV+08] M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar, *Answering what-if deployment and configuration questions with wise*, SIGCOMM '08 (New York, NY, USA), ACM, 2008, pp. 99–110.

[VD08] M. Voortman and M. J. Druzdzel, *Insensitivity of constraint-based causal discovery algorithms to violations of the assumption of multivariate normality.*, FLAIRS Conference (Wilson D. and Lane H. C., eds.), AAAI Press, 2008, pp. 690–695.

[VDD10] M. Voortman, D. H. Dash, and M. J. Druzdzel, *Learning causal models that make correct manipulation predictions with time series data*, Journal of Machine Learning Research: Workshop and Conference Proceedings **6** (2010), 257–266, published in: JMLR Workshop and Conference Proceedings: Volume 6. Causality: Objectives and Assessment (NIPS 2008 Workshop) December 12, 2008, Whistler, Canada.

[VDM+11] M. Vajapeyam, A. Damnjanovic, J. Montojo, Tingfang Ji, Yongbin Wei, and D. Malladi, *Downlink ftp performance of heterogeneous networks for lte-advanced*, Communications Workshops (ICC), 2011 IEEE International Conference on, June 2011, pp. 1–5.

[WB13] K. Winstein and H. Balakrishnan, *TCP ex machina: Computer-generated congestion control*, Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (New York, NY, USA), SIGCOMM '13, ACM, 2013, pp. 123–134.

[Wri21] S. Wright, *Correlation and Causation*, Journal of Agricultural Research **20** (1921), 557–85.

[XKS00] P. Xue-Kun Song, *Multivariate dispersion models generated from gaussian copula*, Scandinavian Journal of Statistics **27** (2000), no. 2, 305–320.

[ZBPS02] Y. Zhang, L Breslau, V. Paxson, and S. Shenker, *On the characteristics and origins of internet flow rates*, Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM, ACM, 2002, pp. 309–322.

[ZLD+12] L. Zhenyu, B. Lijun, Ruwei D., Z. Chongguang, W. Hu, Y. Youbo, W. Wenjuan, and T. Jie, *Exploring the effective connectivity of resting state networks in mild cognitive impairment: An fmri study combining ica and multivariate granger causality analysis*, Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE, Aug 2012, pp. 5454–5457.

[ZPJS12] K. Zhang, J. Peters, D. Janzing, and B. Schölkopf, *Kernel-based conditional independence test and application in causal discovery*, CoRR **abs/1202.3775** (2012).

[ZSJ12] K. Zhang, B. Schölkopf, and D. Janzing, *Invariant gaussian process latent variable models and application in causal discovery*, CoRR **abs/1203.3534** (2012).