A Causal Approach to the Study of TCP Performance

HADRIEN HOURS, ERNST BIERSACK and PATRICK LOISEAU, EURECOM

Communication networks are complex systems whose operation relies on a large number of components that work together to provide services to end users. As the quality of these services depends on different parameters, understanding how each of them impacts the final performance of a service is a challenging but important problem. However, intervening on individual factors to evaluate the impact of the different parameters is often impractical due to the high cost of intervention in a network. It is therefore desirable to adopt a formal approach to understand the role of the different parameters and to predict how a change in any of these parameters will impact performance.

The approach of causality pioneered by J. Pearl provides a powerful framework to investigate these questions. Most of the existing theory is non-parametric and does not make any assumption on the nature of the system under study. However, most of the implementations of causal model inference algorithms and most of the examples of usage of a causal model to predict intervention rely on assumptions such linearity, normality or discrete data.

In this paper, we present a methodology to overcome the challenges of working with real-world data and extend the application of causality to complex systems in the area of telecommunication networks, for which assumptions of normality, linearity and discrete data do no hold. Specifically, we study the performance of TCP, which is the prevalent protocol for reliable end-to-end transfer in the Internet. Analytical models of the performance of TCP exist, but they take into account the state of network only and disregard the impact of the application at the sender and the receiver, which often influences TCP performance. To address this point, we take as application the File Transfer Protocol (FTP), which uses TCP for reliable transfer. Studying a well understood protocol such as TCP allows us to validate our approach and compare its results to previous studies.

We first present and evaluate our methodology using TCP traffic obtained via network emulation, which allows us to experimentally validate the prediction of an intervention. We then apply the methodology to real-world TCP traffic sent over the Internet. Throughout the paper, we compare the causal approach for studying TCP performance to other approaches such as analytical modeling or simulation and and show how they can complement each other.

CCS Concepts: • **Mathematics of computing → Bayesian networks; Causal networks;** *Bayesian computation;* • **Networks → Network performance analysis;** *Network performance modeling;* Transport protocols; Network measurement; Public Internet; • **Computing methodologies → Causal reasoning and diagnostics;**

Additional Key Words and Phrases: Telecommunication Networks, TCP

ACM Reference Format:
Hadrien Hours, Ernst Biersack and Patrick Loiseau, 2014. A Causal Approach to Study TCP Performance *ACM Trans. Intell. Syst. Technol.* V, N, Article A (January YYYY), 24 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

The Internet is an interconnection of many different networks. Each network is only required to offer a very simple service, namely to forward a packet to the next hop towards its final destination. Most of the complex operations are executed in the end-hosts. One example of a protocol executed in the end-hosts is TCP, which offers a reliable data transfer between two end-hosts across the Internet that may loose, delay, or mis-order data. TCP carries the vast majority of todays traffic in the Internet as it is used by application layer protocols such as FTP, HTTP, or mail protocols. For more details we refer the reader to the textbook by Kurose and Ross [Kurose and Ross 2009].

Author's addresses: H. Hours, E. Biersack and P. Loiseau, Networking and Security Department,EURECOM
Some preliminary results were presented in the Algotel and the 10ème Atelier en Evaluation des Performances workshops. The first abstract can be found on http://arxiv.org/abs/1404.5481

DOI: http://dx.doi.org/10.1145/0000000.0000000

Given the importance of TCP, many studies have tried to model and evaluate the performance of TCP. The performance metric used most often is *throughput*, which is defined as the number of bytes delivered per second to the receiving end-host. As we will see, the throughput of a transfer between two end-hosts depends on various parameters such as the average time for a packet to travel between these two end-hosts, which is commonly referred to as round trip time (RTT), or the loss rate experienced, which expresses the percentage of packets that will not reach the destination but get lost due to events such as temporary resource shortage in the network. Aside from modeling the throughput, one also would like to understand the inter-dependencies that exist between the various parameters that all impact the throughput.

The inference and representation of causal models as Bayesian networks provide very powerful tools that we apply to study the performance of TCP and the relationships between the various parameters. When we represent these relationships via a Directed Acyclic Graph (DAG), interventions on individual parameters can be predicted with simple graphical criteria by using data collected prior to this intervention. As interventions on communication networks are expensive or even impossible to carry out, being able to predict them without the need of performing complex and costly controlled experiments represents an important advantage.

However, various challenges arise when trying to apply the existing methods for causal model inference and for intervention prediction in the case of the transport protocol TCP. Indeed, studying TCP performance involves working with a mix of both discrete and continuous parameters whose distributions cannot in general be approximated by normal distributions. Additionally, their dependences do not follow any known functional form, and in particular are not linear. We believe that the challenges of working with data that are non-linear and not normal are common to many real-world scenarios. As we will see later, tools such as WISE [Tariq et al. 2008] or NANO [Tariq et al. 2009] that were proposed in the context of network traffic to study parameter dependences based on their partial correlation do not give the correct results in such scenarios. In this paper, we use the do-calculus [Pearl 2009] and non parametric estimation of multivariate distributions to develop a simple yet formal approach to estimate the effect of interventions on a system where the assumptions of normality and linearity do not hold. Through this work we show that, even when dealing with more complex systems where many existing implementations of causal model inference would fail because they rely on assumptions that do not hold, the causal theory can be applied and give very good and insightful results provided that the adequate recent tools are used appropriately for inference and estimation.

Our contribution can be summarized as follows.

- We show how to adapt the existing methods for the causal study of real systems whose data are nonlinear and not normally distributed.
- To corroborate the soundness of our approach, we perform experiments in a controlled environment where the predictions of interventions can be validated.
- We successfully apply our method to perform the causal study of the performance of TCP used by a file transfer (FTP) application that transmits large amounts of data over the Internet.
- The methods presented are tested and validated through the study of TCP performance, where we illustrate the benefits of adopting a causal approach and we compare it with other approaches.

The theory of causality as developed in [Pearl 2009; Spirtes et al. 2001] 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.[1] Similarly, the examples of predictions of interventions in these works ([Pearl 2009; Spirtes et al. 2001]) only consider discrete data where probability distribution studies can be simplified by the use of conditional

[1]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 [Ramsey 2014] is now available. This test was not present at the beginning of our work and is now under investigation.

probability tables. Dealing with real data, where these assumptions do not hold, poses a number of challenges.

Section 2 discusses the characteristics of TCP and the parameters that impact its performance while Section 3 shows how to adapt existing methods to the nature of these parameters and overcome the challenges of working with continuous data where no distribution or functional form can be assumed.

Once the framework of our work presented, we evaluate in Section 4 our approach on data obtained using the Mininet emulator. Section 5 presents a case study of the performance of TCP using real data obtained via file transfers over the Internet that illustrates the interest of causal modeling for network performance prediction. Section 6 compares our approach using causal modeling to other approaches discussed in the literature. Conclusions and some directions for future work are presented in Section 7.

2. TCP

2.1. Overview

In this section we present TCP and the different parameters that are known to impact its performance. We discuss the different constraints inherent to the study of network performance and how they are handled in our work.

TCP is very widely used in the Internet, which makes it easy to collect traces. TCP has been well studied and there is a good understanding of the different factors that influence its throughput performance. However, the efforts to study the performance of TCP analytically or via simulation typically concentrate on capturing the impact of the *network* on TCP performance and ignore the impact of the TCP sender or receiver. On the other hand, studies of real TCP traffic have shown that it is often not the network that determines TCP performance but receiver limitations or sender limitations [Zhang et al. 2002; Siekkinen et al. 2008]. These mechanisms are captured by our model as we can see later in Figure 4. In the case of receiver limitations, the application at the receiver side is not able to consume data fast enough, in which case the TCP receiver will slow down the sender via flow control. In case of sender limitations, it is the sender that controls the rate to the transmission, which will then determine TCP throughput performance. Another reason for choosing TCP is that we have a long experience studying TCP in the “wild”, which helps us interpret and validate the causal graphs we obtain.

TCP is a relatively complex protocol that implements not only mechanisms for error control and recovery of lost packets, but also mechanisms to react to network congestion by adapting its sending rate to the load of the network. While we cannot describe these mechanisms in detail, it is important to keep in mind that, at a given moment, the sending rate will be controlled by two parameters, namely the *receiver window* and the *congestion window*. The *receiver window* indicates the quantity of data that the receiver can handle, while the *congestion window* regulates the rate at which the sender can inject packets into the network as a function of the load along the path between the two end-hosts.

2.2. TCP performance modeling

In our work, we study the throughput performance experienced by a host that uses TCP. Padhye et al. [Padhye et al. 1998] were among the first to give a formula that expresses the throughput of TCP as a function of the different parameters, which captures the network and operating conditions. A simplified version of this formula is given by

$$throughput = C \cdot \frac{MSS}{RTT \cdot \sqrt{p}} \tag{1}$$

and will be referred to later in this paper as the **PFTK model**. In this equation, *MSS* (Maximum Segment Size) represents the maximum amount of data that can be carried in a single packet, *RTT* (Round Trip Time) represents the time it takes for a packet to traverse the network and its acknowledgement to get back to the sender, p denotes the probability that a loss happens, and C is a constant. The only parameters that can vary dynamically when the state of the network changes are RTT and p. As we can see, the throughput depends in a non-linear way on the parameters RTT and p. The PFTK model was experimentally validated via real data transfers using TCP among end-hosts located in different parts of the world.

The throughput formula of Equation (1) has some limitations: It assumes that the network congestion is the only limiting factor, i.e., it does not consider receiver limitations, it is only valid for the version of TCP that is called TCP Reno, and it assumes that RTT and p are independent. Many extensions to the PFTK model have been proposed in [Fortin-Parisi and Sericola 2004; Kaj and Olsén 2001; Sikdar et al. 2001]. The causal method we present in this paper does not need to make these assumptions. However, to validate our approach, we focus on TCP Reno since this will allow us to use the PFTK model to interpret our result and compare it with the throughput obtained using Equation (1).

TCP is a protocol implemented in every end host connected to the Internet. The load of the network path between two hosts exchanging data on the Internet is not known a priori. Each TCP sender dynamically estimates parameters that capture the state of the network and adapts the rate at which the data is sent accordingly. In our study we are interested in TCP performance when dealing with the network variations, without looking into its implementation details. While the parameterization of the TCP implementation and the causal study of TCP performance could be complementary, the methods we propose in this paper aim to be applicable to a broader range of systems and do not focus on any particular implementation of TCP.

In addition to the inherent nonlinearity of the dependencies of the TCP parameters, another important property is the distribution and the variation of the values of these parameters. The datasets we are working with consist of a mix of discrete and continuous parameters whose values cannot be approximated by normal distributions. These considerations play an important role when working with the densities of the parameters.

2.3. Measuring TCP performance

Compared to cases such as social science or medical studies, a big advantage of studying network performance is the relative ease with which data can be obtained. While, as in every real world experiment, not all the variables can be observed, many of them are available and it is relatively simple to record the traffic at a given point in the network. However, the presence of latent variables is an important concern. Latent variables are variables that are not or cannot be observed but impact more than one of the observed variables.

TCP is a transport protocol that is used by an application to transfer data reliably from the sender to the receiver. In this paper, we consider a file transfer (FTP) application where large files are transmitted from the sender to the receiver using TCP. We investigate two scenarios: The first scenario studies the performance in an emulated environment (c.f. Section 4) and the second scenario comprises a real FTP sender at our university to which the receivers from all over the world connect in order to perform file transfers (c.f. Section 5). In both cases, we perform hundreds of file transfers and passively capture all the data packets sent by the FTP sender. All the packets that belong to the transfer of a single file form a **connection** and each connection corresponds to one sample in our dataset.

Once these packet captures made, we perform an off-line analysis using the Tstat and Intrabase tools [Mellia et al. 2005; Siekkinen et al. 2005] to extract the values of the key parameters for each of the connections. The full list and description of the parameters can be found in the Appendix C.1. Examples for parameters are loss rate, round trip time, or receiver window. Since a connection may last for several minutes, the values of the parameters can vary over the lifetime of a connection. For each connection, we compute the *average value* for each parameter. We will come back to the issue of the sampling frequency of the parameters in Section 7 and we discuss the issue of how to choose the set of parameters that are needed to model our system in Section 3.1.4.

As we show in the following sections, using causal graphs to model TCP *at the level of connections* will allow us to obtain a meaningful model that can be used to make predictions about the effect of interventions on the system.

3. CAUSAL METHODS

In this section we first review some of the existing methods to build a causal graph from observational data, evaluate the performance of the methods that apply to our data, and justify the methodological choices we make. Finally, we show how to use this model to predict the effect of interventions in our system.

3.1. Inference of causal models

In our work we only consider graphical causal models represented by Bayesian networks. Other models, such as structural equation models, exist. However, the use of Bayesian networks presents many advantages, the main one being the existence of simple graphical criteria to infer the mathematical equations to predict the effect of interventions [Pearl 2009; Spirtes et al. 2001]. In addition, the Bayesian networks offer a concise visual presentation of the different causal dependences, which greatly simplifies the understanding and manipulation of complex systems.

3.1.1. Algorithms. Inference algorithms for graphical causal models can be divided roughly into two categories: The score-based ones and the constraint-based ones. The score-based methods 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 given data. There has been an important progress in recent years in “exact Bayesian network inference” using 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. [Koivisto 2006; Koivisto and Sood 2004] propose an algorithm with lower than super-exponential complexity in the number of parameters of the system. These algorithms always converge and allow to stop at any moment obtaining 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.

Constraint-based methods 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 independences between the different parameters and 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 [Spirtes and Glymour 1991], 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 which corresponding parameters are found independent. For the parameters which nodes are still adjacent, we test if there exists a conditioning set of size 1 that makes two adjacent nodes 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 colliders (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 or cycles [Pearl 2009] in a second step. The second method, which is implemented in the kPC algorithm, is based on Weakly Additive Noise models (WAN) [Tillman et al. 2009] 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 study, we used the Tetrad software [Spirtes et al. 2001] to compare the graphs obtained for score-based methods and for different constraint-based methods. Tetrad implements many algorithms for causal model inference (as Bayesian network) such as the PC [Spirtes and Glymour 1991], GES [Maxwell 2002] or iMAGES [Ramsey et al. 2009] algorithms for Pattern inference, LiNGAM [Hoyer and Hyttinen 2009] for Directed Acyclic Graph inference, FCI [Spirtes et al.] 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 assumes either linearity or normality. More precisely, at the time of these tests, no independence test for not normal and not linear data was present. However, despite these limitations, it turned out that constraint-based methods give more informative graphs, which are more in line with our understanding of TCP performance. While these experiments did not prove that score-based methods will not give correct results for our data, we decided to use constraint-based methods for the rest of 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 (We present some of the models obtained with the kPC algorithm in Appendix D.1.6 and Appendix E.1.5). Therefore, we use the method relying on colliders for edge orientation for the rest of work.

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 partially directed acyclic graph, PDAG). It often happens that several graphs exist that imply the detected independences (see Section 3.2.1 on 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 [Murphy 2001] 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 ($X \not\rightarrow Y$). Such knowledge can be used in the inference of a causal model consistent with both, the detected independences and such knowledge. In [Meek 1995], 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 [Meek 1995] is proved 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 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 [Spirtes et al.]. However, [Borboudakis and Tsamardinos 2012] 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]

3.1.2. Independence test. The choice of the independence test is essential for inferring the correct graph. In the different implementations of the PC algorithm from Tetrad [Spirtes et al. 2001], bnt toolbox [Murphy 2001] or the Pcalg R package [Kalisch et al. 2012], the Z-Fisher independence

[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.

criterion is used.[3] The Z-Fisher test assumes linearity and normality of the data. It was shown [Voortman and Druzdzel 2008] that the PC algorithm, with the Z-Fisher criterion, still performs correctly when relaxing the assumption of normality. However, the nonlinearity has a strong negative impact on its accuracy. Our own tests on artificially generated data to study the effect of nonlinearity and non normality on the accuracy of the Z-Fisher criterion confirmed these conclusions (see Appendix A for more details).

To better take into account the nature of our data we use a different independence test that takes into consideration the non linearity of the dependencies between the different parameters and their distributions. The Kernel Conditional Independence test (KCI) from [Zhang et al. 2012] defines covariance operators in the Reproducing Kernel Hilbert Spaces (RKHS) corresponding to the parameters being tested and derives a statistic which distribution can be estimated under the hypothesis of independence. This independence test is very similar to the Hilbert Schmidt Independence Criterion (HSIC) [Gretton et al. 2007], the statistics of the two tests being exactly the same in the unconditional case. The use of kernel based independence test was proposed to handle the cases where no formal functional dependence or distribution can be assumed from the data. As we show in Appendix A, the KCI test achieves better performance than the Z-Fisher criterion when tested on our data. However, its implementation has two problems: First, the computation time to perform independence tests with this criterion is 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. In this case, the test will not return an answer. However, 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.

To solve these two problems, we modified the independence test by including a bootstrap method [Davison and Hinkley 1997], 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 percentage of at least δ of sub-tests detects that the variables are independent. As shown in Appendix B.3, this testing procedure KCI+bootstrap does not decrease the accuracy 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 very 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 more accurate 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.[4]

3.1.3. Study of KCI test performance and choice of parameters. Appendix B 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. The 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

[3]As mentioned in the footnote 1 (page 2), recent versions of Tetrad implement a test for non linear and non normal data that we are currently investigating

[4]We did not further investigate this option in our study

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 (which is not the case 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 9 Appendix B.2.2). 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 [Andrews and Buchinsky 2001], the number of repetitions is often dictated in practice by computational constraints. In Table IX of Appendix B.2.2, 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 (c.f. Appendix B.3, which shows that using $l = 100$, together with $N = 400$, yields the same accuracy of the KCI test on the full dataset).

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. 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 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 dependences 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 Appendix B where, for two different scenarios, the parameterization $N = 400$ and $l = 100$ is validated.

3.1.4. Feature selection. A last point that we would like to mention briefly concerns the choice of the parameters that will form the nodes of the Bayesian network representing the causal model of our system. 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 $\mathcal{L}$, see Section 3.2.2. The causal theory in [Pearl 2009; Spirtes et al. 2001] handles the presence of latent variables. However, as illustrated by the models presented in Figure 18 and Figure 23 in Appendix D.1.4 and Appendix E.1.3 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 Appendix E.1.3.

3.2. Prediction

In this section we assume that a Directed Acyclic Graph (DAG) was obtained and we summarize the different graphical criteria that are used in our work to estimate intervention from a causal graphical model. We use the notation $do(X = x)$ to represent the intervention that sets variable X to the value x.

3.2.1. 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 node where arrows meet head to head). The presence of a collider on a path blocks this path and conditioning on a collider unblocks the path.[5]

3.2.2. Total effect vs. direct effect. In our work we are interested in predicting the effect an intervention on one of the parameters will have on the system. By intervention we mean that we isolate a parameter from the effect of any of its ancestors and we manually set its value (*atomic intervention* [Pearl 2009]). An intervention removes the influence of all external parameters on the (set of) parameter(s) of our intervention and fix its (their) distribution(s) according to the nature of the intervention. Only the distributions of the descendants of the parameters fixed by the intervention will be modified, the ancestors will keep the distributions they were following prior to this intervention. Pearl [Pearl 2009] calls the effect that an intervention has on all parameters the *total effect*.

On the other hand, one can be interested in the effect that one variable has on another, independently of the influence of any other parameter. This effect is referred to as *direct effect*. In the case of linear dependences, the direct effect can be computed by holding constant all the external variables and regressing Y on X (if interested in the direct effect X $\rightarrow$ Y). Instead of holding all the external variables constant we can block their influences by conditioning on a specific set of variables, Z, and obtain the value of the direct of X on Y from the coefficient of the partial regression of X in the regression of Y on X and Z. However, when the assumption of linearity does not hold the value of this coefficient of regression depends on the value at which Z is fixed. Without this assumption, the formal computation of direct effect is more complex [Pearl 2013]. For this reason, while the study of direct effects could give interesting insights, it is not central to our problem and left for future work.

To estimate the total effect, on a parameter Y, of an intervention on parameter X, we can use the *Back-door Criterion* or the *Front-door Criterion* [Pearl 2009]. In this work we use the back-door criterion defined as follows.

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, Y) *in a DAG* G *if:*

(*1*) *no node in Z is a descendant of X; and*
(*2*) *Z blocks every path between X and Y that contains an arrow into X.*

Using the back-door definition, we can now predict the effect of an intervention with the following theorem:

THEOREM 1. ***(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:*

$$P(y \mid do(X = x)) = \sum_{z} P(Y = y \mid X = x, Z = z)P(Z = z). \tag{2}$$

3.2.3. Density estimation. We can see from Equation (2) 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 (2) the variables are assumed to be discrete, which is not the case for all variables in our data set. In addition, the parameters we consider in our system do not have an a priori distribution nor can they be approximated with a mixture of known distributions such as normal, gamma, beta, and so on (see Appendix C.2 for examples of our parameter distributions and [Nagode and Fajdiga 2011] for an example of an estimation of a univariate distribution with a mixture of pre determined distributions). We address these difficulties using copulae [Jaworski

[5]Intuitively, two independent causes become dependent if one conditions on their common consequence

et al. 2010]. Copulae are models to build multivariate densities from their marginals. Some of their properties are summarized hereafter.

The Sklar Theorem stipulates that, if F is a multivariate cumulative distribution function with marginals $(F_1, \dots, F_n)$, there exists a copula C such that

$$F(x_1, \dots, x_n) = C(F_1(x_1), \dots, F_n(x_n)). \tag{3}$$

In the bivariate case, differentiating Equation (3) gives the following result:

$$f(x_1, x_2) = c(F_1(x_1), F_2(x_2)) f_1(x_1) f_2(x_2), \tag{4}$$

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

$$f_{X_1|X_2}(x_1, x_2) = c(F_1(x_1), F_2(x_2)) f_1(x_1). \tag{5}$$

For the back-door criterion, we often place ourselves in the case illustrated in Equation (5). When $dim(X_2) > 1$, we use Equation (3) to estimate the multivariate CDF of X_2 and use this estimation in Equation (5).

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 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

$$C_{\rho,\nu}(u, v) = \int_{-\infty}^{t_\nu^{-1}(u)} \int_{-\infty}^{t_\nu^{-1}(v)} \frac{1}{2\pi(1-\rho^2)^{1/2}} \left(1 + \frac{x^2 - 2\rho xy + y^2}{\nu(1-\rho^2)}\right)^{-\frac{(\nu+2)}{2}} \mathrm{d}x\mathrm{d}y \tag{6}$$

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.

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

4. NETWORK EMULATION

In this section, we validate the approach described in the previous section in the context of network data obtained via emulations. We use the Mininet emulation tool [Lantz et al. 2010], which allows us to make interventions and compare the results to the predictions obtained from the causal graph.

4.1. Mininet network emulator

Mininet is a powerful network emulator based on Container Based Emulation (CBE) [Handigol et al. 2012]. 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 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.

4.2. Experimental setup

Figure 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 [FreeBSD 2009]. It consists of the different file downloads issued by the client **C1**. The traffic is 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**.

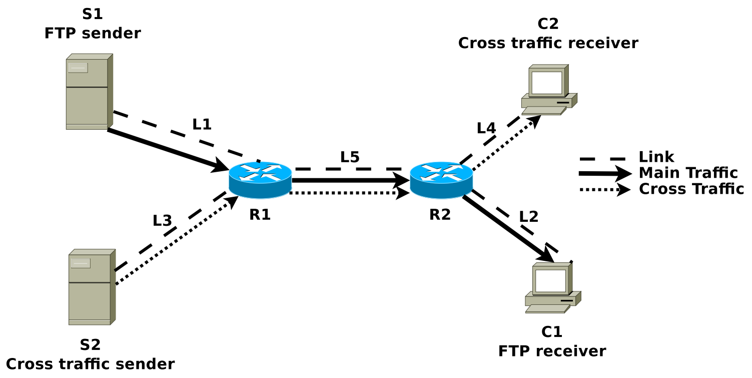


Fig. 1: Emulated network using Mininet

Once we have captured the packet trace, we use Tstat [Mellia et al. 2005] and Intrabase [Siekkinen et al. 2005], to compute the different metrics summarized in Table I. 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*.

Table I: 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

4.3. Causal model

The causal graph of our emulated FTP traffic is presented in Figure 2. It was inferred using the PC algorithm with the KCI test, implemented with the bootstrap method (Section 3.1.2), 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 I 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 [Audrius et al. 2011] 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 2 (*retrscore*→ *p*) and the orientation found in the real case scenario (*p*→ *retrscore*, see Figure 4 in Section 5.2).

The models presented in Figure 2 and in Figure 4 both indicate that *rtt* impacts *p*. Such dependence is absent from the PFTK model and would be interesting to study, which we leave for future work. 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, as we mentioned in Section 2.1, represents the capacity of the receiver to handle the quantity of data that is sent by the Server. The PFTK model assumes that the TCP performance is not "receiver limited" and ignores the parameter *rwin* in Equation (1). In the model of Figure 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 Appendix D.1.4, we present in Figure 18 the model obtained using the IC* algorithm [Pearl 2009] 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 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.[6]

4.3.1. Remark on the Z-fisher criterion. Due to space constraints, the model we obtain with the classical approach that uses the Z-Fisher criterion to test for independence is presented in Figure 16 in Appendix D.1.2. We can see that this model is very different from the one presented in Figure 2, which shows the inadequacy of the Z-Fisher criterion to test for independence in the case of our dataset.

4.4. 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 3.2.2) for the dataset Table I (and in the associated graph of Figure 2), to the performance observed in a controlled emulation where the same intervention is actually made. Given the model shown in Figure 2 and the interventions that we can perform, the best parameter to intervene on is the retransmission score

[6]The experiment was repeated, selecting random values for the parameters of the simulation. The corresponding model is presented in Figure 15 in Appendix D.1.1

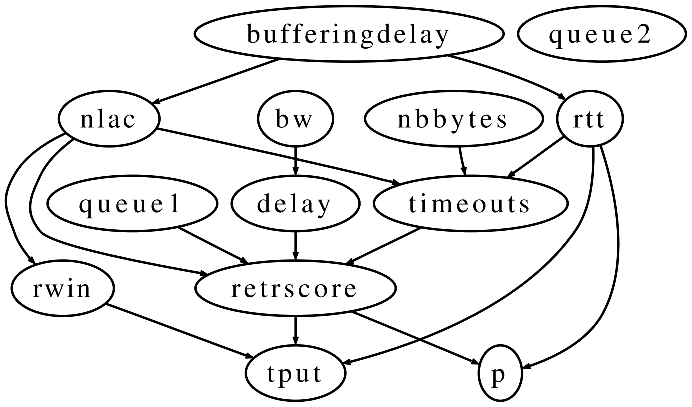


Fig. 2: Causal model inferred by the PC algorithm, with KCI test, for the emulated network traffic

(*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 (2), it is necessary to have, in our dataset, samples where the values of the parameter we want to set, by intervention, are observed. 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 (2)) 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 3.2.1, 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 back-door 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 3.2.2, 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

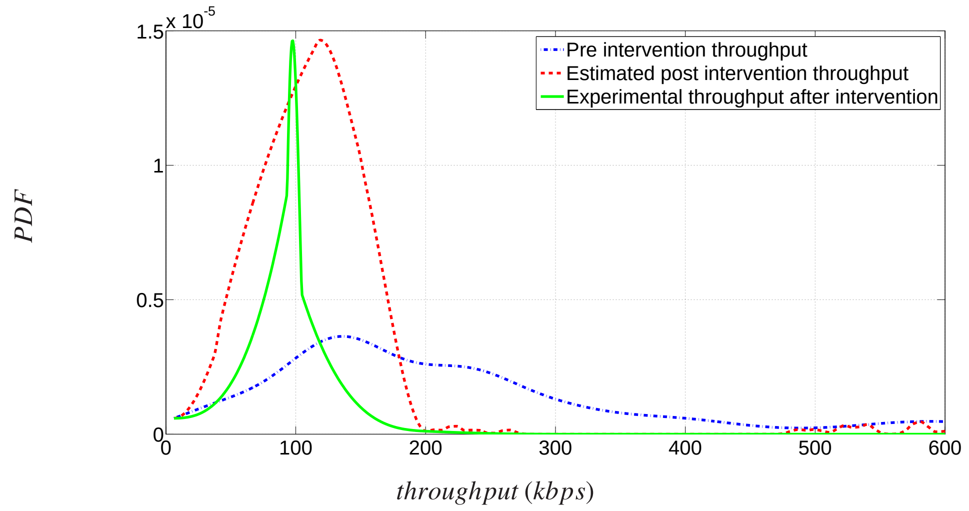


Fig. 3: Effect of an intervention on the retransmission score, on the throughput distribution

nlac must still be taken into account, but their impact on the variable of intervention, *retrscore*, is spurious and must be neglected, which is implied by the back-door 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 3 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.

In this section we have validated the causal method using a network emulation system called Mininet where the prediction of an intervention could be tested. Despite the limitations of Mininet, the prediction we obtain validates our causal approach and demonstrates its usefulness. In the next section, we apply our causal approach to real traffic data sent over the Internet.

5. CAUSAL STUDY OF TCP PERFORMANCE UNDER FTP

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.

5.1. 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 receivers, 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 II.

Table II: 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, beginning of the connection (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 between client and server (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

5.2. Causal model

Using the PC algorithm with the KCI test, implemented with the bootstrap method (Section 3.1.2), we obtain the graph presented in Figure 4.[7] Below 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 detected in the model presented in Figure 2 and it was mentioned in Section 4.3. The study of this dependence is out of the scope of this paper but illustrates the limitations of empirical models such as the PFTK model.

This last point illustrates the advantage of a causal study over a purely statistical or empirical one, such as the model represented in Equation (1). 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 4.4.

[7]For more insights on possible latent variables we refer the reader to Appendix E.1.3 and Figure 23 which presents the output of the IC* algorithm [Pearl 2009] where we can see that almost all parents of *tput* have marked directed edges, excluding the possibility of latent variables. The model obtained with FCI (Fast Causal Inference) algorithm [Spirtes et al. 2001] is also presented Figure 24 in Appendix E.1.4

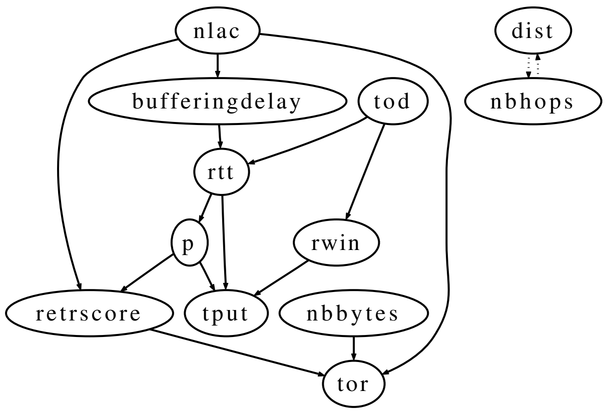


Fig. 4: Causal graph modeling FTP transfers for real traffic

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 4.3). 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 II). 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 II 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 4.2, *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 [Ford et al. 2002] 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 [En-Najjary and Urvoy-Keller 2006], 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 from the server and the client is the main factor influencing the *RTT* and plays an important role in the TCP Performance.

5.2.1. Remark on Z-Fisher criteria. Due to space constraints the model obtained using the Z-Fisher criterion instead of the KCI test is presented in Figure 21 of the Appendix E.1.1. It should be mentioned that, as discussed previously, the model we obtain with the Z-Fisher criterion shows some properties in contradiction with how TCP operates and confirms that this criterion is not able to detect the independences for our dataset.

5.3. 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 (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 4.4).

5.3.1. Intervening on the Round Trip Time. From the causal graph presented in Figure 4, we want to predict the effect of manually intervening on the *RTT*. To do so, we use the back-door criterion (cf Section 3.2.2) which ensures the identifiability of this intervention. There is one trek between *rtt* and *tput* blocked by *tod*.[8] We can apply the back-door adjustment, Theorem 1, with the blocking set $Z = \{tod\}$. As in Section 3.2.3, 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 II 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 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 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 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 4, 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 4.4, 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 have 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.[9]

5.3.2. 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* 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.

[8]"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" [Spirtes et al. 2001]
[9]Amazon claimed that every 100 ms of latency costs it 1% of profit.

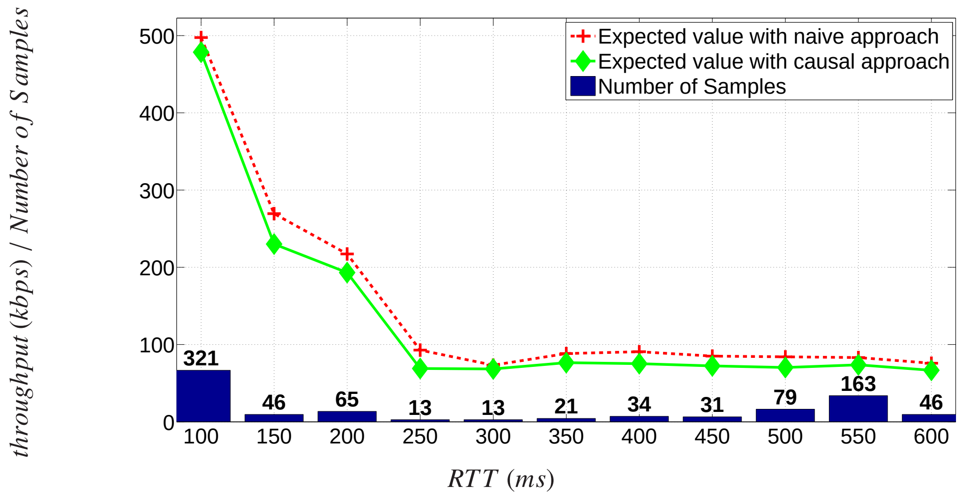


Fig. 5: Effect of an intervention on the RTT, on the throughput distribution

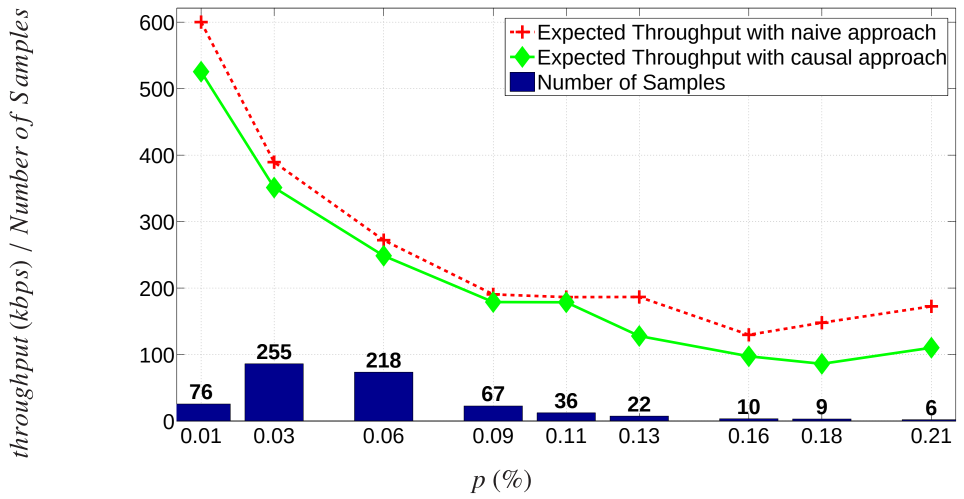


Fig. 6: Effect of an intervention on p, on the throughput distribution

Figure 6 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 when intervening on RTT.

5.3.3. Intervening on the receiver window. One parameter absent from the the PFTK model, Equation (1), but present in the model Figure 4 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. The average value of the receiver window prior to intervention is 137 kB for an average throughput of 332 kBps, see Table II. 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 [Siekkinen et al. 2005] result that show that the impact of the receiver window on the throughput is less important than the one of loss and delay.

This does not imply that an intervention on the receiver window should never be made. To decide on an intervention, an operator must balance benefits and costs. Intervening on the loss or delay means upgrading network equipments or moving their locations, 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).

5.3.4. Discussion. 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. Then, our study shows that, provided that the right tools are used appropriately to cope with the data characteristics (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.

6. RELATED WORK

Some of the preliminary results concerning the emulated network and the FTP traffic studies can be found in workshop papers [Hours et al. 2014a] and [Hours et al. 2014b] respectively. The current version significantly extends both the results and description of the proposed methodology.

To the best of our knowledge, the only previous work applying causal models in the area of computer networks is the WISE system [Tariq et al. 2008]. WISE is a system that estimates a causal graph from network data collected in the core of a CDN (Content Distribution Network) and uses it to predict the impact of configurations changes (e.g., locations of the servers). Our study differs from WISE in some important aspects. First, when inferring the causal graph, WISE uses the Z-Fisher independence criterion and the definition of "no cause variables". We could observe that the Z-Fisher criterion performs very poorly due to the properties of our dataset, namely nonlinearity and non-normality. Adapting the Kernel-based Conditional Independence test [Zhang et al. 2012], we obtain good results with methods requiring less resources than the WISE system. Second, when using the causal model, WISE only uses the graph to reduce the number of regressors when applying knowledge propagation to estimate the post intervention distribution. In contrast, we adopt a more formal approach based on the works of [Pearl 2009; Spirtes et al. 2001] which allows us to obtain consistent results with fewer resources. When we used non parametric regression to orient the edges in the kPC algorithm [Tillman et al. 2009] we observed that this approach does not show enough accuracy in modeling the dependencies between the different TCP parameters, while the use of causal graphs and the criteria based on d-separation [Pearl 2009] lead to more accurate results and require fewer resources (please refer to the models inferred by the kPC algorithm presented in Appendix D.1 and Appendix E).

There have been cases where the Internet Service Providers (ISP) were found to violate the network neutrality of their users [Dischinger et al. 2008]. Network neutrality means that no application or (subset of) client gets privileged access to any service or resource. In Nano [Tariq et al. 2009], the authors make a causal study of the violation of network neutrality. The challenge in this case is to separate the part of the performance degradation due to the ISP policy from the factors impacting network performance that are not related to the ISP, such as the client OS, its distance from the service, and so on. These parameters are called *confounding factors*. To estimate the causal effect of an ISP on network performance, Nano clusters the confounding factors in bins of similar values and blocks spurious associations. An exhaustive list of confounding factors is made based on domain knowledge. However, Nano does not use a causal graph to block spurious associations. As a consequence, Nano must condition on all possible confounding variables. If one parameter happens to be a collider in a path between another confounding variable and the outcome variable, such conditioning could create a spurious association. Additionally, Nano uses partial correlation to test for independence inside a bin, which turned out to be inaccurate for our scenario.

The performance of TCP has been modeled extensively. The first model was proposed by [Padhye et al. 1998]. This model was based on Markov processes and was extended by other works along the same line that study TCP performance under various conditions (see e.g., [Loiseau et al. 2010] and references therein). However, such models generally limit the number of parameters they include and do not permit to predict the impact of interventions in the system.

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 designed 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, that approaches that blindly mimic the system under study cannot provide [Winstein and Balakrishnan 2013].

In [Shah et al. 2007; Mirza et al. 2007] a statistical approach is used to model TCP performance. However, these models become obsolete when the conditions under which they were built change. Most importantly, these models are highly dependent of the presence of spurious correlations that will be taken into account in the computations and will bias the estimates, as shown in our study (cf Section 5.3). One important contribution of causal models comes from their ability to detect spurious associations and the graphical criteria to cancel them. As we could observe for the dependence between the Round Trip Time and loss probability, causal models detect the dependencies among the explanatory variables which, otherwise, would lead to biased estimates of the performance of the system and its evolution post intervention.

Finally, while the parameters considered in our work rely on a given version of TCP, [Sikdar et al. 2001; Kaj and Olsén 2001; Fortin-Parisi and Sericola 2004] present studies and models of more recent versions of this protocol that should guide the application of the causal methods presented in this paper to a broader range of communication network protocols.

7. CONCLUDING DISCUSSION

This paper presents a first attempt to use a formal causal approach to study the performance of communication networks. While the theory of causal modeling is well developed and a number of software tools are available to apply it, our work showed that, in order to successfully apply these tools, one needs to carefully study the nature of the data and check whether or not the assumptions made by these tools hold. In our case of network traffic, we need to handle continuous variables that are non-normal and with non-linear dependencies, which entails some challenges. First, using the appropriate tool for independence testing in the PC algorithm was key to infer plausible causal graphs. We adapted the Kernel based Conditional Independence test (KCI) with a bootstrap method that allows handling all the constraints of our datasets. Second, for the prediction of post-intervention distributions, we introduced the use of copulae to model multidimensional conditional densities in order to adapt the existing methods to our datasets. The use of copulae gave consistent results even when working on a limited amount of data. From this perspective, our study shows that it is possible to use all the methods offered by the causal theory to study complex systems, even if no assumption can be made about the parameters distributions or the functional form of the parameter dependences.

Using an appropriate methodology as described above, we obtained graphs consistent with our domain knowledge that enable very useful intervention predictions. The independences detected with the KCI+bootstrap method lead to graphs that are fully in accordance with our knowledge of the telecommunication network mechanisms. While such knowledge was used to orient the edges that these independences could not orient (the edges for which choosing an orientation does not change the Markov Equivalence Class), the models we obtain also present dependencies that domain knowledge alone may not be able to uncover (see for example the discussion about the Receiver Window Auto-Tuning mechanism or the dependence between the *RTT* parameter and the p parameter in Section 5.2).

We first validated our method by verifying the accuracy of the predictions based on our causal approach with controlled emulations (using Mininet). Then, we used our method on real-world traffic generated by the FTP application. We successfully built a causal model of TCP performance and used it to predict the effect on throughput of modifying its three main causes, the RTT, the loss probability and the receiver window. Our results also show the benefits of our causal approach by shedding light on the biased estimates that a correlation based approach would give and by highlighting dependences that would have been missed otherwise.

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 [Thomas et al. 2012] 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 paper 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 5.3.3). 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 results accuracy will likely be very sensitive to the assumptions made. We leave investigation of this problem for future work.

We plan to continue our work along to following lines. Causal models can be used for the prediction of intervention plans where more complex and more realistic scenarios (for which we intervene on several parameter distributions simultaneously) can be predicted. Handling those more complex scenarios will require to overcome the difficulties encountered in modeling the conditional probability density functions with many dimensions. One possible approach could be the use of parametric models for approximating the marginals, which extend the range of situations that can be predicted. Another approach relies on Chow-Liu tree [Chow and Liu 1968; Pearl 1988] in combination with the Bayesian network representation of our causal model to estimate our joint probability distribution as a product of second-order conditional probability. This approach could highly benefit from the use of copulae, as seen in Section 3.2.3. However, the presence of continuous variables with several parents (cf Figure 2 where one node has 4 parents) might lead to important approximations and loss in accuracy when compared with the joint probability model given by the Bayes chain rule and the underlying Markovian hypothesis.

Another important challenge to address in future work is the sampling frequency of observations. So far, we have represented the parameters of a connection by their average values. However, sub-sampling may impact the inference of the causal model as it can prevent the detection of dependencies or create non structural ones. We started to use the Web10G software [Mathis et al. 2003] to continuously observe the variables of the TCP state machine during the duration of a connection, and plan to investigate how these fine-grain data can be use to build more accurate causal models.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Kun Zhang of the Max Planck Institute in Tuebingen, Germany, for providing us with the implementation of the Kernel based Conditional Independence test (KCI) and for answering questions related to its implementation. We would also like to thank the editors and the reviewers for their very constructive and insightful comments on earlier versions of this paper.

REFERENCES

D. W. K. Andrews and M. Buchinsky. 2001. Evaluation of a three-step method for choosing the number of bootstrap repetitions. *Journal of Econometrics* 103, 1-2 (July 2001), 345–386.

J. Audrius, L. Jukka-Pekka, H. Matti, and W. Alf Inge. 2011. An Empirical Study of NetEm Network Emulation Functionalities.. In *ICCCN*. IEEE, 1–6.

G. Borboudakis and I. Tsamardinos. 2012. Incorporating Causal Prior Knowledge as Path-Constraints in Bayesian Networks and Maximal Ancestral Graphs. *ArXiv e-prints* (June 2012).

C. Chow and C. Liu. 1968. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on* 14, 3 (May 1968), 462–467. DOI:http://dx.doi.org/10.1109/TIT.1968.1054142

A. C. Davison and D.V. Hinkley. 1997. *Bootstrap Methods and their Application*. Cambridge University Press.

M. Dischinger, A. Mislove, A. Haeberlen, and K. P. Gummadi. 2008. Detecting Bittorrent Blocking. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement (IMC '08)*. ACM, 3–8.

T. En-Najjary and G. Urvoy-Keller. 2006. PPrate: A Passive Capacity Estimation Tool. In *E2EMON*. 82–89.

P. Ford, A. Shelest, and N. Srinivas. 2002. Method for automatic tuning of TCP receive window. (Aug. 15 2002). US Patent App. 09/736,988.

S. Fortin-Parisi and B. Sericola. 2004. A Markov Model of TCP Throughput, Goodput and Slow Start. *Perform. Eval.* 58, 2+3 (Nov. 2004), 89–108.

FreeBSD. 2009. FreeBSD File Formats Manual. online, http://www.freebsd.org/cgi/man.cgi?query=tar. (2009).

A. Gretton, K. Fukumizu, C. H. Teo, L. Song, B. Schölkopf, and A. J. Smola. 2007. A Kernel Statistical Test of Independence.. In *NIPS*. Curran Associates, Inc.

N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown. 2012. Reproducible Network Experiments Using Container-based Emulation. In *CoNEXT '12*. ACM, New York, NY, USA, 253–264.

H. Hours, E. Biersack, and P. Loiseau. 2014a. A causal study of an emulated network. In *10ème Atelier en Evaluation de Performances, 11-13 juin 2014, Sophia-Antipolis, France*.

H. Hours, E. Biersack, and P. Loiseau. 2014b. Causal study of network performance. In *ALGOTEL 2014, 16èmes Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications, June 3-6, 2014, Le-Bois-Plage-en-Ré, France*.

P. O. Hoyer and A. Hyttinen. 2009. Bayesian Discovery of Linear Acyclic Causal Models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI '09)*. AUAI Press, Arlington, Virginia, United States, 240–248.

P. Jaworski, F. Durante, W.K. Härdle, and T. Rychlik. 2010. *Copula Theory and Its Applications*.

I. Kaj and J. Olsén. 2001. Throughput modeling and simulation for single connection TCP-Tahoe. In *ITC-17*. Teletraffic Science and Engineering, Vol. 4. Elsevier, 705–718.

M. Kalisch, M. Mächler, D. Colombo, M. H. Maathuis, and P. Bühlmann. 2012. Causal Inference Using Graphical Models with the R Package pcalg. *Journal of Statistical Software* 47, 11 (2012), 1–26.

M. Koivisto. 2006. Advances in Exact Bayesian Structure Discovery in Bayesian Networks.. In *UAI*.

M. Koivisto and K. Sood. 2004. Exact Bayesian Structure Discovery in Bayesian Networks. *J. Mach. Learn. Res.* 5 (Dec. 2004), 549–573.

J. F. Kurose and K. W. Ross. 2009. *Computer Networking: A Top-Down Approach* (5th ed.). Addison-Wesley Publishing Company, USA.

B. Lantz, B. Heller, and N. McKeown. 2010. A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-IX)*. ACM, New York, NY, USA, Article 19, 6 pages.

P. Loiseau, P. Gonçalves, J. Barral, and P. Vicat-Blanc Primet. 2010. Modeling TCP Throughput: an Elaborated Large-Deviations-Based Model and its Empirical Validation. In *In Proceedings of IFIP Performance*.

M. Mathis, J. Heffner, and R. Reddy. 2003. Web100: extended TCP instrumentation for research, education and diagnosis. *Computer Communication Review* 33, 3 (2003), 69–79.

D. C. Maxwell. 2002. Learning Equivalence Classes of Bayesian-network Structures. *J. Mach. Learn. Res.* 2 (March 2002), 445–498.

C. Meek. 1995. Causal Inference and Causal Explanation with Background Knowledge. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence (UAI'95)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 403–410.

M. Mellia, R. Lo Cigno, and F. Neri. 2005. Measuring IP and TCP Behavior on Edge Nodes with Tstat. *Comput. Netw.* 47, 1 (Jan. 2005), 1–21.

M. Mirza, J. Sommers, P. Barford, and X. Zhu. 2007. A machine learning approach to TCP throughput prediction. In *SIGMETRICS '07*. ACM, New York, NY, USA, 97–108.

K. P. Murphy. 2001. The Bayes Net Toolbox for MATLAB. *Computing Science and Statistics* 33 (2001), 2001.

M. Nagode and M. Fajdiga. 2011. The REBMIX Algorithm for the Univariate Finite Mixture Estimation. *Communications in Statistics - Theory and Methods* 40, 5 (2011), 876–892.

J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. 1998. Modeling TCP throughput: a simple model and its empirical validation. *SIGCOMM Comput. Commun. Rev.* 28, 4 (Oct. 1998), 303–314.

J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

J. Pearl. 2009. *Causality: Models, Reasoning and Inference*. Cambridge University Press, New York, NY, USA.

J. Pearl. 2013. Direct and Indirect Effects. *CoRR* abs/1301.2300 (2013).

J. Ramsey. 2014. A Scalable Conditional Independence Test for Nonlinear, Non-Gaussian Data. *CoRR* abs/1401.5031 (2014).

J. D. Ramsey, S. J. Hanson, C. Hanson, Y. O. Halchenko, R. A. Poldrack, and C. Glymour. 2009. Six problems for causal inference from fMRI. *Neuroimage* (2009).

S.M.H. Shah, A.u. Rehman, A.N. Khan, and M.A. Shah. 2007. TCP throughput estimation: A new neural networks model. In *Emerging Technologies, 2007. ICET 2007. International Conference on*. 94–98.

M. Siekkinen, E.W. Biersack, G. Urvoy-Keller, V. Goebel, and T. Plagemann. 2005. InTraBase: integrated traffic analysis based on a database management system.. In *E2EMON*. IEEE, 32–46.

M. Siekkinen, G. Urvoy-Keller, E.W. Biersack, and D. Collange. 2008. A root cause analysis toolkit for TCP. *Computer Networks* 52, 9 (2008), 56.

B. Sikdar, S. Kalyanaraman, and K. S. Vastola. 2001. Analytic Models for the Latency and Steady-State Throughput of TCP Tahoe, Reno and SACK. In *In Proc. of the IEEE GLOBECOM*. 25–29.

P. Spirtes and C. Glymour. 1991. An Algorithm for Fast Recovery of Sparse Causal Graphs. *Social Science Computer Review* 9 (1991), 62–72. Issue 1.

P. Spirtes, C. Glymour, and R. Scheines. 2001. *Causation, Prediction, and Search* (second ed.). The MIT Press, Cambridge, MA, USA.

P. Spirtes, C. Meek, and T. Richardson. Causal Inference in the Presence of Latent Variables and Selection Bias. In *In Proceedings of Eleventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 499–506.

M. Tariq, M. Motiwala, N. Feamster, and M. Ammar. 2009. Detecting network neutrality violations with causal inference.. In *CoNEXT*. ACM, 289–300.

M. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar. 2008. Answering what-if deployment and configuration questions with wise. In *SIGCOMM '08*. ACM, New York, NY, USA, 99–110.

B. Thomas, R. Jurdak, and I. Atkinson. 2012. SPDYing Up the Web. *Commun. ACM* 55, 12 (Dec. 2012), 64–73.

R. Tillman, A. Gretton, and P. Spirtes. 2009. Nonlinear directed acyclic structure learning with weakly additive noise models. In *In Advances in Neural Information Processing Systems (NIPS)*. Vancouver, Canada.

M. E. Tipping. 2001. Sparse Bayesian Learning and the Relevance Vector Machine. *J. Mach. Learn. Res.* 1 (2001), 211–244.

M. Voortman and M. J. Druzdzel. 2008. Insensitivity of Constraint-Based Causal Discovery Algorithms to Violations of the Assumption of Multivariate Normality.. In *FLAIRS Conference*. AAAI Press, 690–695.

K. Winstein and H. Balakrishnan. 2013. Tcp ex machina: Computer-generated congestion control. In *ACM SIGCOMM Computer Communication Review*, Vol. 43. 123–134.

K. Zhang, J. Peters, D. Janzing, and B. Schölkopf. 2012. Kernel-based Conditional Independence Test and Application in Causal Discovery. *CoRR* abs/1202.3775 (2012).

Y. Zhang, L Breslau, V. Paxson, and S. Shenker. 2002. On the Characteristics and Origins of Internet Flow Rates *(SIGCOMM)*. ACM, 309–322.

Online Appendix to: A Causal Approach to the Study of TCP Performance

HADRIEN HOURS, ERNST BIERSACK and PATRICK LOISEAU, EURECOM

A. STUDY OF Z-FISHER AND KCI CRITERIA

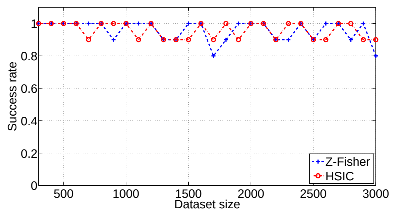
In this appendix we present the different tests that were executed to compare the performance of the KCI and Z-Fisher criteria. We generate artificial data with different functions and under different conditions.

A.0.5. 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 7, 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:

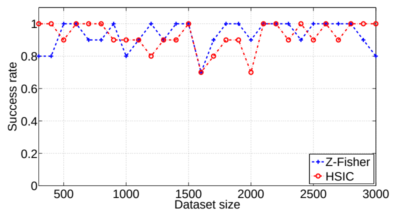
- **Test 1** [Normal distribution, same variance]: The values for both parameters are drawn independently from two normal distributions $\mathcal{N}(0, 1)$
- **Test 2** [Normal distribution, different variance]: The values for both parameters are drawn independently from two different normal distributions, $X \sim \mathcal{N}(100, 5)$, $Y \sim \mathcal{N}(5, 2)$
- **Test 3** [Real parameter distribution, same variance]: We randomly draw samples from the RTT and Throughput data values of Table II, and normalize them[10]
- **Test 4** [Real parameter normal distribution, different variance]: We randomly draw samples from the RTT and Throughput data values of Table II

[10] $X_{normalized} = \frac{X-\mu_X}{\sigma_X}$

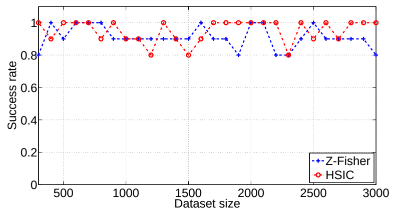
© YYYY ACM. 2157-6904/YYYY/01-ARTA $15.00
DOI: http://dx.doi.org/10.1145/0000000.0000000



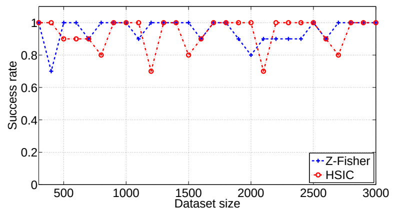
(a) Test 1



(b) Test 2



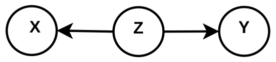
(c) Test 3



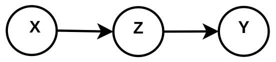
(d) Test 4

Fig. 7: 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

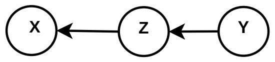
These results do not show any preference for one criteria over the other in any of the situations. They also do not show a clear improvement when the sample size increases.



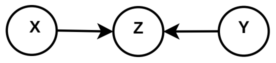
(a) $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$



(b) $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$



(c) $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$



(d) $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$

Fig. 8: Different graphical configurations for orienting V-structure

A.0.6. 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 $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$. In the first configuration we have $\mathbf{X} \perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$ 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 8. However, the situations presented in Figure 8b and Figure 8c are similar, so we will not differentiate them (inverting the role of $\mathbf{X}$ and $\mathbf{Y}$ will not give more information on the performance of a criteria). We will only study the cases represented by Figure 8a, Figure 8b and Figure 8d.

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 III-V and their corresponding results in Tables VI-VIII.

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 II. 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 III: 16 scenarios for testing conditional independences with the Z-Fisher criterion and KCI test, for the case illustrated in Figure 8a, $Z^\circ = Z - min(Z) + 1$.

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 IV: 16 scenarios for testing conditional independences with the Z-Fisher criterion and KCI test, for the case illustrated in Figure 8b, $X^{\circ} = X - min(X) + 1$, $Y^{\circ} = Y - min(Y) + 1$.

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 V: 16 scenarios for testing conditional independences with the Z-Fisher criterion and KCI test, for the case illustrated in Fig.8d, $X^{\circ} = X - min(X) + 1$.

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 VI: Results of the 16 conditional independence scenarios for the model in Figure 8a, 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 VII: Results of the 16 conditional independence scenarios for the model in Figure 8b 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	0	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 VIII: Result of the 16 tests for the model Fig.8d for 6 different data set sizes using the Fisher criterion and KCI test $\mathcal{I}_1 = X \perp\!\!\!\perp Y$, $\mathcal{I}_2 = X \perp\!\!\!\perp Y/Z$.

B. INDEPENDENCE TEST: KCI AND BOOTSTRAP

In this appendix, we present experiments on the efficiency and accuracy of the KCI tests (with and without our bootstrap method) that support our choice of using the bootstrap and our choices of parameters presented in Section 3.1.3 (l and N).

B.1. KCI test algorithm and numerical considerations

We first briefly describes the KCI test principle and the important numerical considerations that affect its accuracy.

The Kernel based Conditional Independence test (KCI) maps the initial dimensions into a new space, Hilbert Schmidt Space (HSS), using definite positive kernels. The translation can be seen as a bijective operation. In the new space, Reproducing Kernel Hilbert Space (RKHS), a new operator, the cross covariance operator, inspired by the covariance and conditional covariance in the original spaces, allows to test the non linear independences. The computation of the cross-covariance operator and conditional cross-covariance operator can be approximated and some simplifications allows to reduce the complexity of the independence test between two parameters. However, one of these simplifications makes use of the Cholesky factorization which tends to fail if the variability of one of tested parameter is not big enough. This is a known numerical limitation of the HSIC implementation (see, e.g., Appendix B in [Tipping. 2001]); due to a matrix which, although theoretically invertible, appears singular in practice.

Taking as example the conditional test of the *time of the day* and *distance* conditionally on {*bufferingdelay*,*rtt*,*p*} for the dataset presented in Table II, we observed that the KCI test on the full dataset fails 100% of the time, in its use of Cholesky factorization. This problem was initially solved by rejecting the H_0 hypothesis in case of failure, but this solution is not satisfying. In contrast, the bootstrap method overcomes this limitation for two reasons: (*i*) using smaller re-sampled datasets ($N = 400$ for us) reduces the numerical problems in the Cholesky factorization and (*ii*) out of a large number ($l = 100$ for us) of randomly re-sampled datasets, we always find many where the Cholesky factorization does not fail. As a result, we are able to detect independence in datasets where the Cholesky factorization was systematically failing with the KCI test alone on the full dataset; which, to some extent, can be interpreted as an increase of the test accuracy thanks to the bootstrap method.

B.2. Completion time

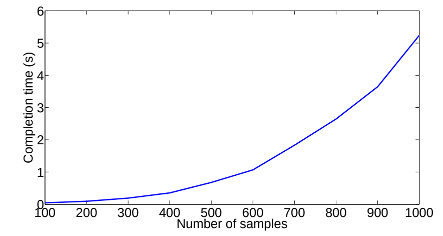
We now discuss computation time constraints.

B.2.1. KCI test completion time. In a first experiment, we record the time it takes for the KCI test implementation from [Zhang et al. 2012] to complete an independence test for different dataset sizes and for different conditional set sizes.[11]

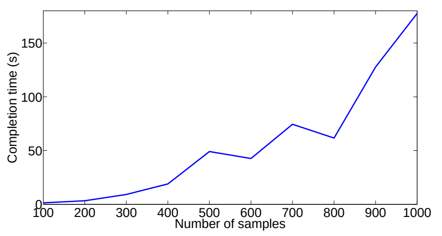
The results are shown in Figure 9. All graphs show that the completion time seems to increase faster than linearly with the number of samples. In practice, for a sample size larger than 400, the completion time starts to increase very fast with the number of samples. Together with the observation that no important improvements are obtained for sample sizes bigger than 400 samples (see Section B.3), this motivated our choice of the value 400 for N.

B.2.2. Influence of bootstrap parameterization on completion time. We now provide estimates of real computation times to infer a graph with the bootstrap method for different parameters. Taking as an example the dataset from Section 5 and the number of independences tested by the PC algorithm, Table IX presents the time it would take for different values of our bootstrap implementation to complete the inference of the causal graph. The values were obtained by counting the number of independences for each conditioning set size in the case where the Z-Fisher criterion is used and by

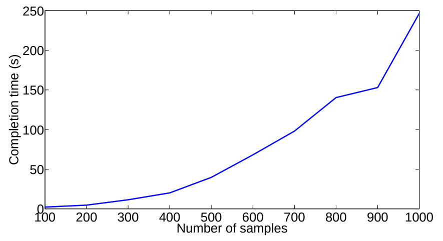
[11]Note that this completion time was obtained with MATLAB 2009b, while the same trends are still valid for latest release of MATLAB.



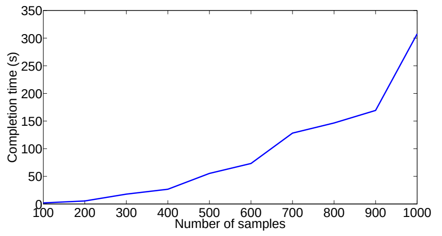
(a) Unconditional test



(b) Conditioning set of size 1



(c) Conditioning set of size 2



(d) Conditioning set of size 3

Fig. 9: Evolution of the completion time of the KCI test as function of the number of samples for different conditioning set sizes

recording the time it takes for each type of independence test with different parameterization in our bootstrap method.

Note that, due to time constraints, 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 CPU was used during the tests so the difference in the number of CPUs is not important). This 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 IX 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 B.1), 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 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.

Table IX: 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

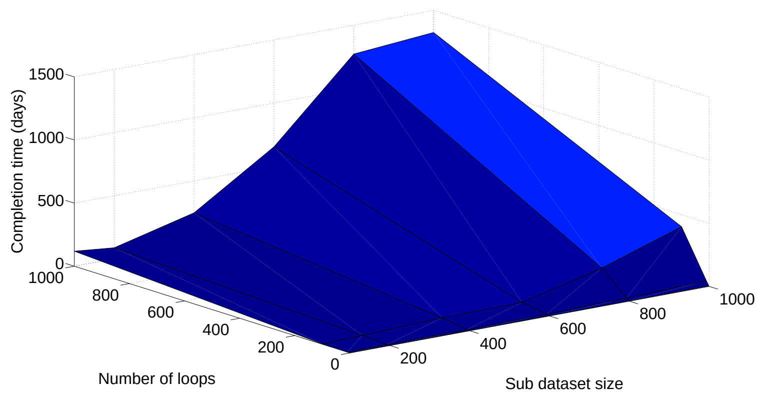


Fig. 10: Evolution of the completion time for different values of the bootstrap method

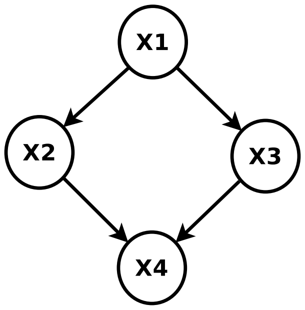


Fig. 11: Causal graph of the artificial dataset of 4 variables

B.3. KCI and bootstrap tests accuracy

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

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 errors of the test in a conditional independence setting, we consider the graph of Figure 11. In order to stay close to our original system, we generate X_1 by randomly re-sampling with replacement the throughput, from the dataset of Table II first, and from the dataset of Table I after, to keep the same distribution; and we generate X_2, X_3 and X_4 using non-linear functions:

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

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

- $\mathcal{I}_\mathbf{1}$: $X_1 \perp\!\!\!\perp X_4 | \{X_2, X_3\}$: H_0 should be accepted;
- $\mathcal{I}_\mathbf{2}$: $X_2 \perp\!\!\!\perp X_3 | \{X_1, X_4\}$: H_0 should be rejected.

B.3.1. Generating X1 from the throughput observed at the FTP server in the dataset of Table II in Section 5. Figures 12a and 12b 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 12a 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 12b 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 12c 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).

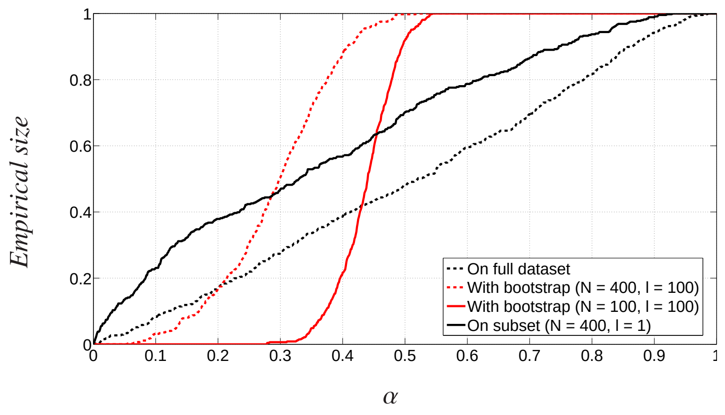
B.3.2. Generating X1 from the throughput observed at the FTP server in the dataset of Table I in Section 4. We repeat the same experience as previously but, this time, X_1 is generated from the throughput of the emulated network, Table I. The size, the power and the power as a function of the size are represented in Figure 13a, Figure 13b and Figure 13c respectively.

We focus our discussion on the graph of the power as function of the size, Figure 13c. 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 12c. 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 4. 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.

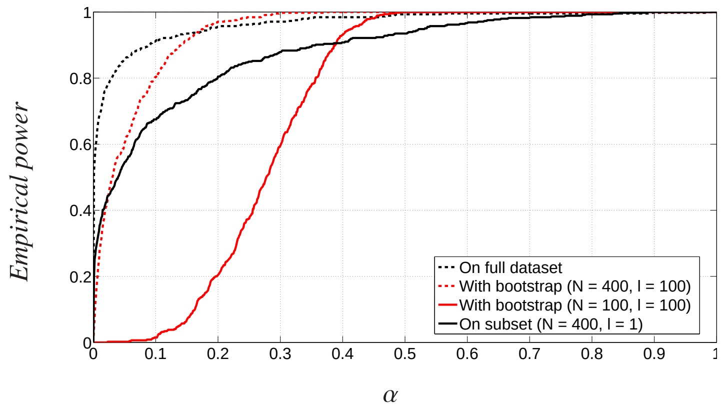
B.3.3. Concluding remarks. The important difference between the previous two cases is that, in the case where X_1 was generated from the emulated network throughput, the setting $\{N = 100, S = 400\}$ was used to infer the graph that supported the predictions we made and that could be verified. As independences are not known in advance we used 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 last point validates the different steps.

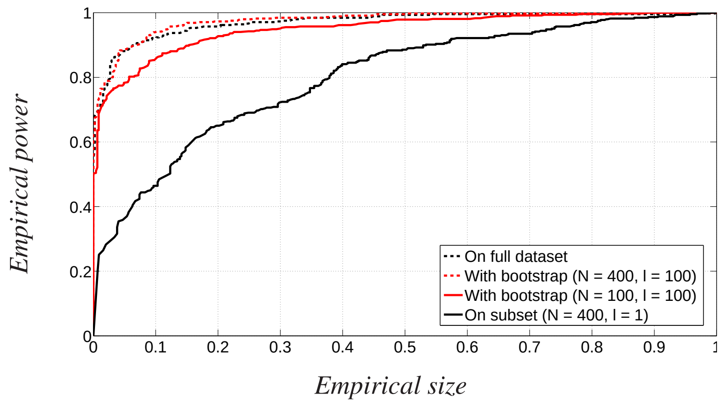
The fact that, for both scenarios, the setting $N = 400$ and $l = 100$ meets our objectives in terms of accuracy and resources validates this choice. The small difference in the observed performance of the test for the two scenarios is also validating the approach as i) We can see that the two datasets 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. This last point suggests that we can expect the performance of this setting of the KCI test + bootstrap to be at least as good in the real case scenario as in the emulated case, where we could verify that our methods perform correctly.



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

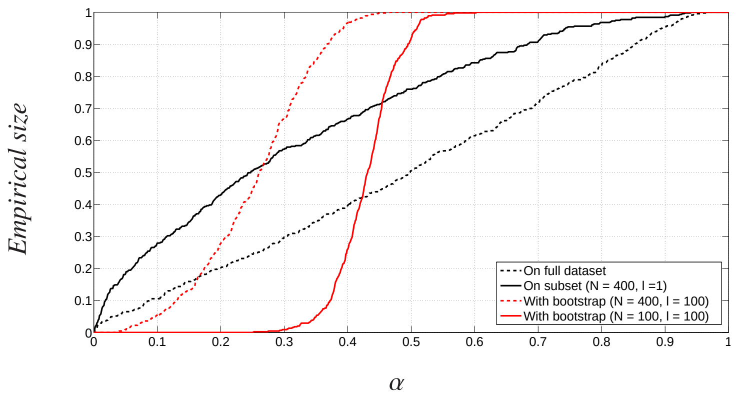


(b) Empirical test power (i.e., fraction of correctly rejected independence $\mathcal{I}_2$) as function of test parameter α

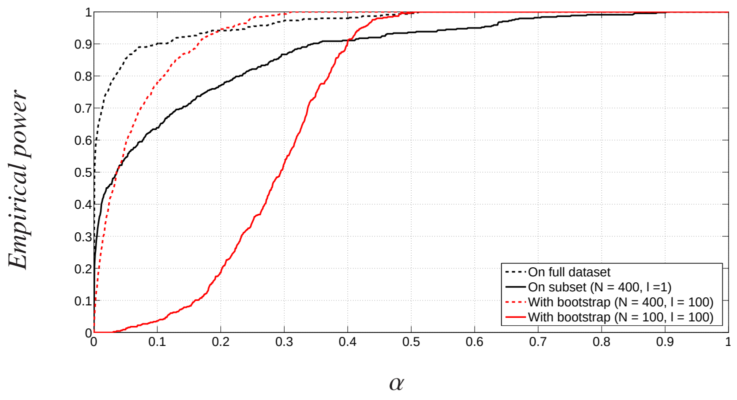


(c) Empirical test power as a function of the empirical test size

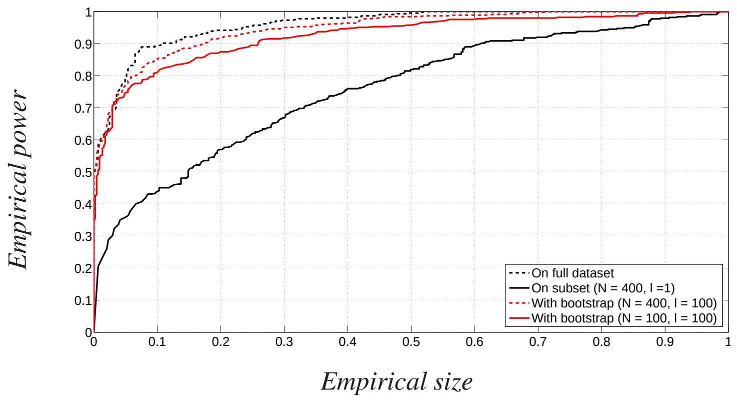
Fig. 12: Comparison of size and power of the KCI test with and without the use of bootstrap



(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

Fig. 13: 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

C. TCP PERFORMANCE PARAMETERS

C.1. List and definition of the different parameters

In the emulated environment we record the following parameters

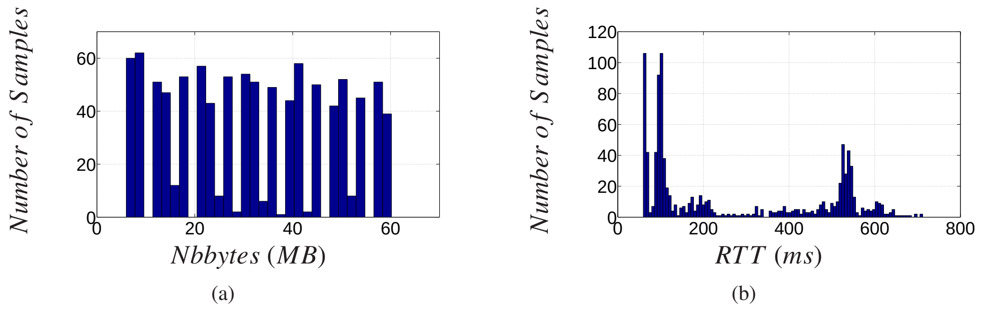
- Bandwidth: The maximum capacity of the path between the client and the server [*bw*]
- Propagation delay: The time, when no queuing happens, for a packet to go from the server to the client [*delay*]
- Size of buffer: The maximum number of packets that can be stored by a router, R_X, in case of congestion [*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*]

In the real case scenario, due to the fact that we place ourselves at the edge of the network, we do not record parameters that describe the state of the routers. However, we additionally record the following parameters

- Distance: Distance between the server and the client [*dist*]
- Time of the day: Number of elapsed seconds since midnight when the connections was opened. This parameter captures the peak hour effect [*tod*]
- Number of hops: Estimation of the number of routers between the server and the client

C.2. Empirical distributions of some key parameters

In this section we present the histogram of some of the parameters of the real FTP traffic dataset, described in Table II in Section 5. Figure 14 shows the histograms of the following four parameters: *nbbytes*, *RTT*, *p* and *throughput*. It see from these histograms that the parameters do not follow a specific distribution.



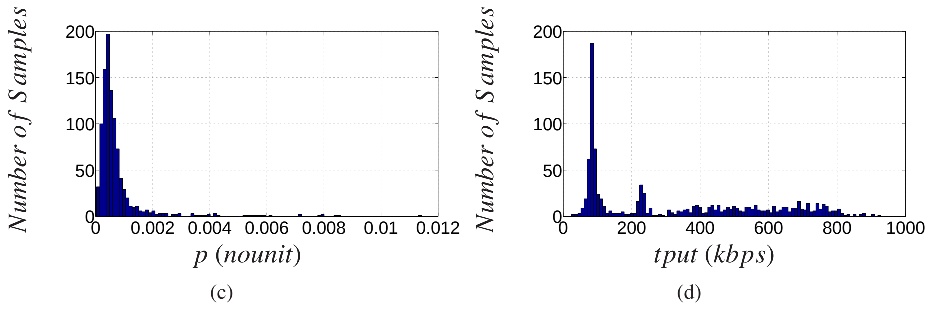


Fig. 14: Histograms of the *nbbytes (a)*, *rtt (b)*, *p (c)* and *tput (d)* parameters observed in the study of Real FTP traffic

D. ADDITIONAL RESULTS FOR THE MININET DATA SET

D.1. Causal model

D.1.1. Randomizing the simulation. We repeated the simulation presented in Section 4.2 by randomly selecting the values of the simulation parameters (delay, jitter, bandwidth, buffer size).

The corresponding dataset is presented in Table X and the corresponding causal model in Figure 15. We can observe that the graph is lacking many important edges when compared to the one obtained with no randomization (Figure 2 of Section 4.3). In the experiment we set up in Section 4.2 we choose the values of the parameters such that 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 dependences 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 2 are not present in Figure 15, 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 X: 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

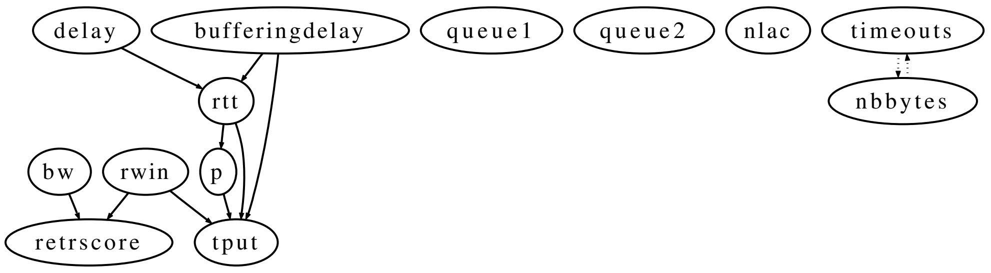


Fig. 15: Causal model obtained for the emulated network scenario when randomization is used

D.1.2. Z-Fisher criterion. Figure 16 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 16 is quite different from the one presented in Figure 2. The absence of *retrscore* as a parent of *tput* is an important dependence that is not present here. The path *bufferingdelay* → *bw* → *delay* → *queue2* → *queue1* → *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).

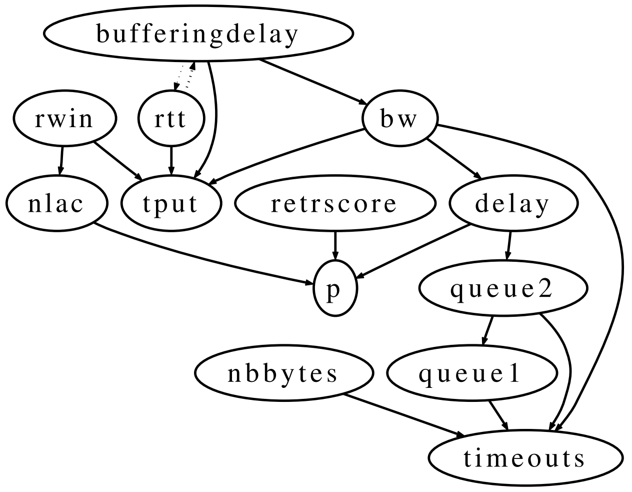


Fig. 16: Causal model inferred by the PC algorithm, with Z-Fisher criterion, for the emulated network traffic

D.1.3. Result log linear. Figure 17 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 I. Based on the Equation (1), applying a log transformation to the data could have could have brought the dependences 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 implied by the original model (Figure 16). 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* → *rwin* → *nlac* → *tput*).

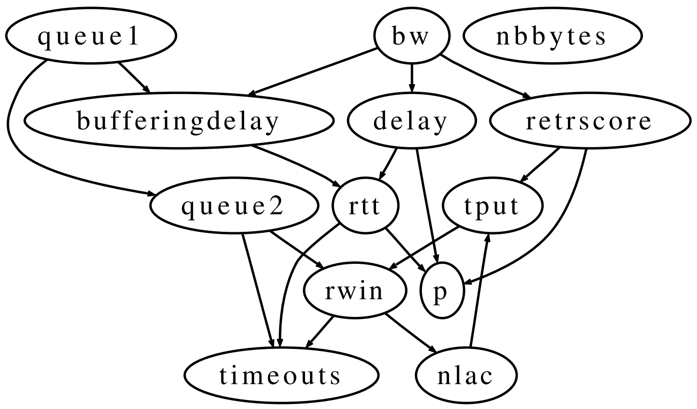


Fig. 17: Model for the network emulation data after applying log-linear transform, using Z-Fisher criterion

D.1.4. Result IC algorithm.* Figure 18 presents the model inferred using the IC* algorithm [Pearl 2009] on the emulated network dataset, summarized in Table I in Section 4.2. The graph output by the IC* algorithm may contain four types of edges:

(1) A marked arrow, signifying a directed path from *X* to *Y*, in the underlying model.
(2) 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.
(3) A undirected edge signifying the presence of a latent common cause of *X* and *Y*, in the underlying model.
(4) 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 model 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. 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 Section 4.4.

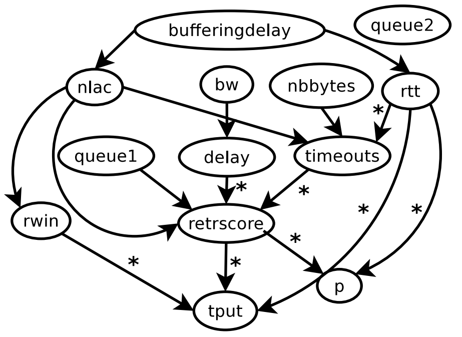


Fig. 18: Model for the network emulation data using the IC* algorithm

D.1.5. Result FCI algorithm. The FCI algorithm [Spirtes et al. 2001], 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[12] and a subset of observable variables.
- The presence of a tail means that the tail is present in every MAG of the equivalence class
- 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 egdemark 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.

Figure 19 presents the model inferred using the FCI algorithm with the Z-Fisher criterion.[13] As it was already observed, Z-Fisher does not seem to be able to correctly capture the dependences 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 3.2.1. As this criterion tends to consider a bigger set of possibilities, by having an inaccurate independence criterion, it leads to a less informative model.

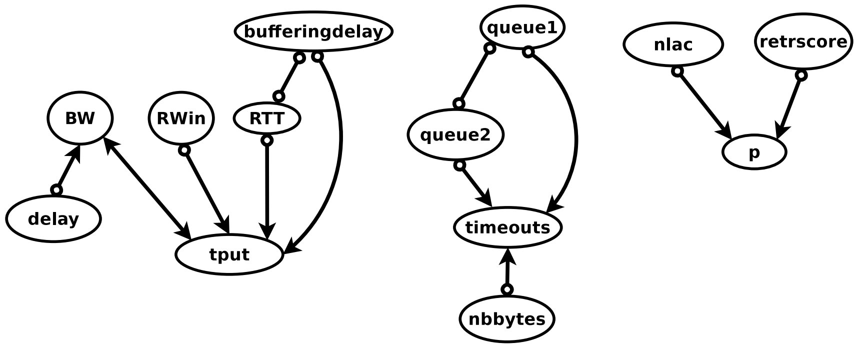


Fig. 19: Causal model inferred by the FCI algorithm and Z-Fisher criterion for the network emulation data

D.1.6. Result kPC algorithm. Figure 20 presents the model inferred by the kPC algorithm when applied to the emulated network scenario, which dataset is summarized in Table I. 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 independent one from another and the edges connecting them to the throughput are left unoriented. 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

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

[13]In our work we used the implementation from the R software Pcalg [Kalisch et al. 2012] which does not provide a solution for testing independences in the presence of nonlinearity and non normality.

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.[14].

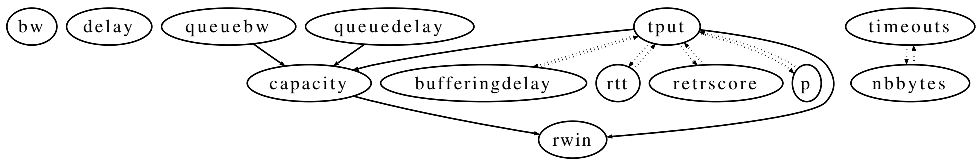


Fig. 20: Causal model inferred by the kPC algorithm for the emulated network scenario

[14]The kPC algorithm implementation from [Tillman et al. 2009] originally uses Support Vector Regression but due to error in its implementation from the Spider toolbox, Multi Ridge Regression is used instead.

E. ADDITIONAL RESULTS FOR REAL FTP DATA

E.1. Causal models

E.1.1. Result using 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 21 the model inferred by the PC algorithm using the Z-Fisher criterion instead of the KCI test. 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 21 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.

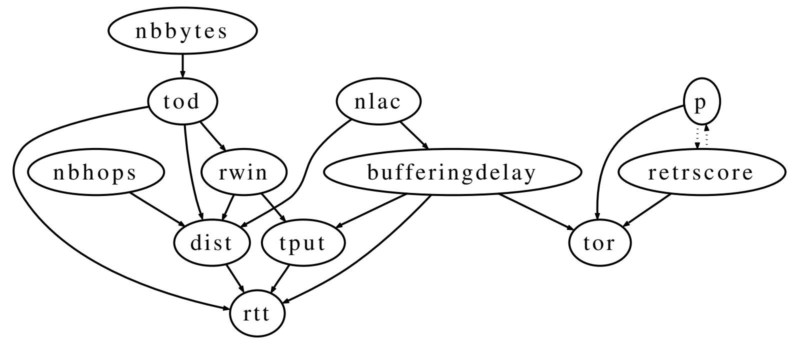


Fig. 21: Causal model inferred by the PC algorithm, with Z-Fisher criterion, for the real FTP traffic

E.1.2. Result log linear. Figure 22 presents the causal model obtained with the PC algorithm and Z-Fisher criterion when applying a log-linear transformation. Based on Equation (1), this transformation of the data tries to obtain approximately linear dependencies.

Here again some of the causal dependences support the possible inaccuracy, even after log-linear transform, of the Z-Fisher criterion to capture the dependences between the parameters. Some of the direct parents of the throughput (*rtt*, *rwin*, *retrscore*, *p*) are the ones we expect to find even if some of the dependences are somehow counter intuitive (for example *rwin* → *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 re oriented without changing the Markov Equivalence Class to which the model belongs.

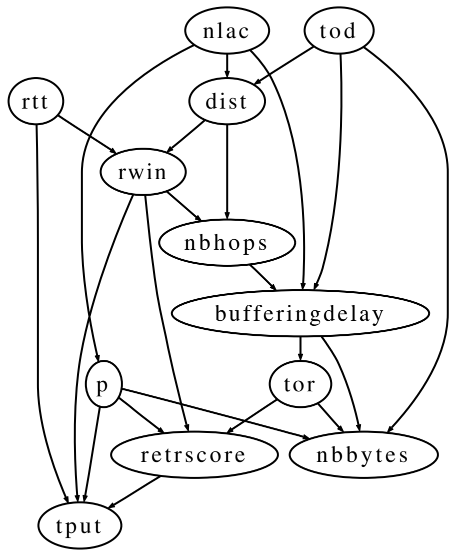


Fig. 22: Causal model inferred with PC and Z-Fisher when applying log-linear transform to the real FTP scenario

E.1.3. Result IC algorithm.* Figure 23 presents the causal model inferred by the IC* algorithm when applied to the real FTP dataset, summarized in Table II in Section 5.1. 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 or not a latent variable between the two parameters. In this case it is our understanding of the TCP mechanisms that helps 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 23 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.

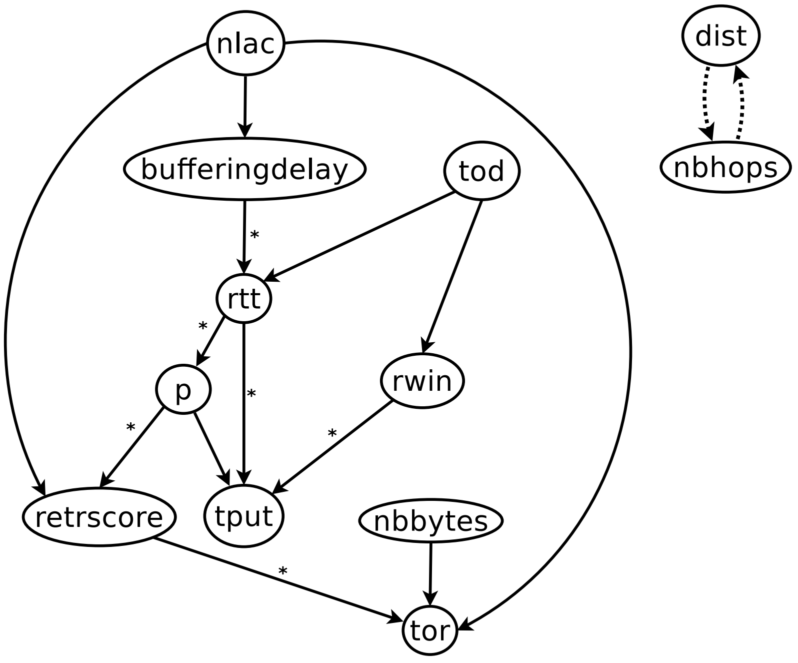


Fig. 23: Model for the real FTP data using the IC* algorithm

E.1.4. Result FCI algorithm. The results obtained when using the FCI with the Z-Fisher criterion are presented Figure 24. 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.

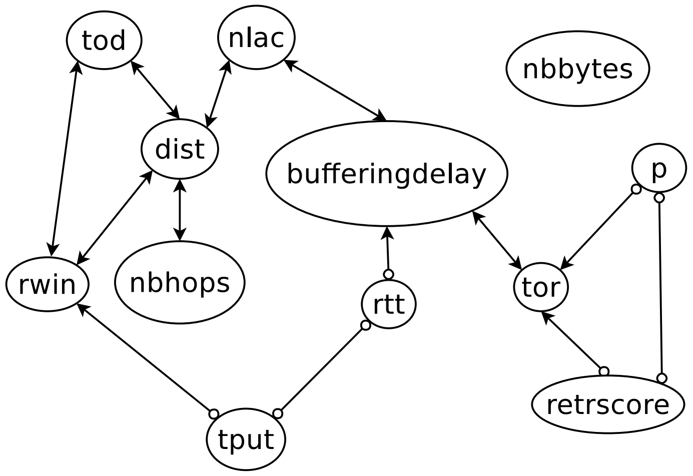


Fig. 24: Model obtained when using FCI algorithm with Z-Fisher criterion in the real FTP scenario

E.1.5. Result kPC algorithm. Figure 25 presents the model inferred by the kPC algorithm when trying to build a causal model of the real FTP scenario, whose dataset is summarized in Table II.

Here also 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 not rely on Meek rules [Pearl 2009] 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 dependences and its impact on methods that rely on their characterization.

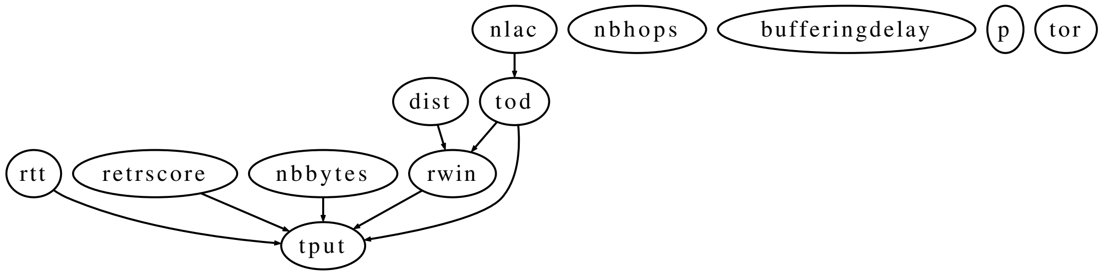


Fig. 25: Graphical causal model inferred by the kPC algorithm for the real FTP scenario