Shifting Network Tomography Toward A Practical Goal

Denisa Ghita, Can Karakus[*], Katerina Argyraki, Patrick Thiran
EPFL, Switzerland

ABSTRACT

Boolean Inference makes it possible to observe the congestion status of end-to-end paths and infer, from that, the congestion status of individual network links. In principle, this can be a powerful monitoring tool, in scenarios where we want to monitor a network without having direct access to its links. We consider one such real scenario: a Tier-1 ISP operator wants to monitor the congestion status of its peers. We show that, in this scenario, Boolean Inference cannot be solved with enough accuracy to be useful; we do not attribute this to the limitations of particular algorithms, but to the fundamental difficulty of the Inference problem. Instead, we argue that the "right" problem to solve, in this context, is compute the *probability* that each set of links is congested (as opposed to try to infer which particular links were congested when). Even though solving this problem yields less information than provided by Boolean Inference, we show that this information is more useful in practice, because it can be obtained accurately under weaker assumptions than typically required by Inference algorithms and more challenging network conditions (link correlations, non-stationary network dynamics, sparse topologies).

1. INTRODUCTION

Network performance tomography can be a powerful monitoring tool: it makes it possible to observe the status of end-to-end paths and infer, from that, the status of individual network links. The basic idea is to express the status of each observed path as a function of the status of the links that make up the path; in this way, we can form a system of equations, where the known entities are the path observations and the network topology, while the statuses of the links constitute the unknowns. The appeal of the approach is that it is applicable in scenarios where one needs to monitor a network without having direct access to its links: a coalition of end-users could use network performance tomography to monitor the behavior and performance of their Internet Service Providers (ISPs); an ISP operator could use it to monitor the behavior and performance of its peers.

[*]Currently at Bilkent University,Turkey.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ACM CoNEXT 2011, December 6–9 2011, Tokyo, Japan.
Copyright 2011 ACM 978-1-4503-1041-3/11/0012 ...$10.00.

On the other hand, there are reasons to be skeptical about the applicability of this approach in practice. First, tomographic algorithms necessarily make assumptions that cannot be verified in a real network (Sections 2 and 3), which means that their results may be inaccurate and, most importantly, there is no way to tell *to what extent* they are inaccurate. Second, tomographic algorithms are typically designed and evaluated over generated graphs that model full router-level or Autonomous-System-level topologies; yet there is no evidence that these models capture well the topologies encountered in scenarios like the ones mentioned above, where using tomography would make sense—indeed, we will see that the topologies encountered in these scenarios can be significantly sparser (Section 3).

In this work, we look at whether and how network performance tomography could be useful in the following real scenario. A European Tier-1 ISP (we will call it the "source ISP") wants to monitor the behavior and performance of its most "important" peers, i.e., the peers through which it routes most traffic that it cannot deliver directly to the corresponding destination's ISP. In particular, for each peer, the source ISP wants to understand: when the peer is responsible for connectivity/performance problems encountered by the customers of the source ISP; how frequently the peer is congested and how its congestion level changes over the course of day or week; how well the peer reacts to exceptional situations like Border Gateway Protocol (BGP) failures, flash crowds, or distributed denial-of-service attacks. Of course, the source ISP does not have access to its peers' networks and cannot directly monitor their links; it can only perform end-to-end path measurements, i.e., monitor a number of one-way paths from its own network to various Internet end-hosts that go through the peers in question. So, the source ISP's operators asked us: can we apply network performance tomography to these end-to-end measurements to answer some or all of the above questions regarding the peers?

At first, this scenario sounded like a good match for Boolean Inference algorithms [12, 8, 6, 11], which monitor paths during a particular time interval (on the order of a few minutes) and infer which particular links on these paths were congested during that interval. In principle, the source ISP could use one of these algorithms to infer which particular links of each peer were congested during each time interval, which would help answer all of the above questions.

Yet Boolean Inference turned out to be too hard a problem in this scenario. State-of-the-art Inference algorithms performed significantly worse than expected, even when adjusted and fine-tuned to the scenario. Our initial reaction was to focus on the limitations of existing algorithms and design a new one that would overcome them; we found that each feature or twist we added to our algorithm to improve it came at the cost of significant complexity, yet brought little benefit—in the end, all Inference algorithms that we tried performed very well under certain conditions (randomly congested links, link independence, stationary network dynamics, dense topologies) and equally badly under the opposite conditions, which are the ones that interest us. So, in the scenario considered by this work, we could not solve Boolean Inference with sufficient accuracy to be useful.

Instead, we argue that, in this scenario, the "right" problem to solve is Congestion Probability Computation, i.e., compute, for each set of links in the network, the probability that the links in this set are congested. This is less information than what would be provided by Boolean Inference: the source ISP learns only *how frequently* each set of links of each peer are congested over a long period of time (hours or so), as opposed to *which particular* set of links of each peer are congested during each particular time interval (of minutes or so). On the other hand, we will show that, in practice, this information is more useful, because it can be obtained accurately under weaker assumptions and more challenging network conditions.

After reviewing existing results on Boolean performance tomography and the assumptions that they rely on (Section 2), we make two contributions:

(i) We experimentally show that, in the scenario where an ISP wants to monitor the behavior and performance of its peers, state-of-the-art Boolean-Inference algorithms (including our own) are not accurate enough to be useful (Section 3). We argue that this is not due to the limitations of the particular algorithms, but that any Inference algorithm can fail under certain conditions, and there is no practical way of knowing whether and when these conditions occur.

(ii) We argue that, in this scenario, it is more useful to solve an easier problem, i.e., compute the probabilities that different sets of links are congested (Section 4). We present a new algorithm that solves this problem and experimentally show that it is accurate under weaker assumptions than those required by Boolean Inference and the network conditions imposed by our scenario—link correlations, non-stationary network dynamics, sparse topologies (Section 5).

We present related work in Section 6 and conclude in Section 7.

2. BACKGROUND

In this section, we summarize existing results related to Boolean network tomography and the assumptions that they rely on.

We use the following network model: The network is a directed graph, where the vertices represent network elements (end-hosts or routers) and the edges represent logical links between network elements. We denote by E^* the set of all links in the network, and by e_i the i-th link based on an arbitrary ordering. A *path* is a sequence of links starting from and ending at an end-host. We denote by P^* the set of all paths in the network, while p_i is the i-th path based on an arbitrary ordering. There are no loops in the network, i.e., any given link participates in any given path at most once, and the set of paths P^* remains unchanged during each measurement period.

We divide time into even intervals, such that each experiment involves a finite sequence of T intervals. We model the congestion status of link e_i during an experiment as a stationary random process. We say that link e_i is *good* (resp. *congested*) during an interval, if it drops less than or equal to (resp. more than) a fraction f of the packets it receives during that interval. The status of link e_i during interval t is modeled with a random variable $X_{e_i}(t)$:

$$X_{e_i}(t) = \begin{cases} 1, & \text{if } e_i \text{ is congested during interval } t \\ 0, & \text{otherwise.} \end{cases}$$

For a given link e_i, all random variables $X_{e_i}(t), t = 1..T$, are identically distributed, and X_{e_i} denotes any one of them. Links $\{e_i, e_j, e_k, \dots\}$ are said to be *independent* when the random variables $\{X_{e_i}, X_{e_j}, X_{e_k}, \dots\}$ are mutually independent; otherwise, the links are *correlated*. Similarly, we model the congestion status of path p_i during an experiment as a stationary random process. We say that path p_i is *good* (resp. *congested*) during an interval, if it drops less than or equal to (resp. more than) a fraction f^d of the packets sent along path p_i during that interval, where d is the number of links traversed by p_i [8]. The status of path p_i during interval t is modeled with a random variable $Y_{p_i}(t)$:

$$Y_{p_i}(t) = \begin{cases} 1, & \text{if } p_i \text{ is congested during interval } t \\ 0, & \text{otherwise.} \end{cases}$$

For a given path p_i, all random variables $Y_{p_i}(t), t = 1..T$, are identically distributed, and Y_{p_i} denotes any one of them.

All the algorithms that we will discuss rely on the following assumptions and conditions:

ASSUMPTION 1. *Separability: A path is good if and only if all the links it traverses are good.*

ASSUMPTION 2. *E2E Monitoring: Based on end-to-end measurements, we can determine whether a path is good during a particular time interval.*

CONDITION 1. *Identifiability: Any two links are not traversed by the same paths.*

We use the term "assumption" to refer to a statement whose correctness is impossible to test given the set of all links E^* and the set of all paths P^*; we use the term "condition" to refer to a statement whose correctness *can* be tested given E^* and P^*.

The **Boolean Inference** problem is the following: given the network graph, a particular time interval t, and the set of congested paths $P^c(t)$ during interval t, infer the set of congested links $E^c(t)$ during that interval [8]. This problem is ill-posed: given any network graph [1] and a particular outcome (set of congested paths) $P^c(t)$, there may be multiple possible solutions (set of congested links) $E^c(t)$ that could have led to this outcome. For example, suppose that, in Fig. 1, all three paths are congested during a time interval; there are 8 possible sets of congested links that could have led to this outcome: $\{e_1, e_3\}$, $\{e_1, e_4\}$, $\{e_2, e_3\}$, $\{e_1, e_2, e_3\}$, $\{e_1, e_2, e_4\}$, $\{e_1, e_3, e_4\}$, $\{e_2, e_3, e_4\}$, and $\{e_1, e_2, e_3, e_4\}$.

All inference algorithms are subject to certain common sources of inaccuracy. First, the four assumptions mentioned above do not always hold in practice; for example, a network operator typically detects whether a path is good during a time interval through probing, which may incur false negatives and false positives. Moreover, since Boolean Inference is an ill-posed problem, no algorithm can solve it exactly (identify the congested links $E^c(t)$ without false negatives or positives) for any $P^c(t)$. However, it is possible to compute an approximation of $E^c(t)$ that is close to the actual solution when certain additional assumptions hold. Hence, what distinguishes different inference algorithms from one another is the set of additional assumptions that each of them introduces. One popular assumption is:

ASSUMPTION 3. *Homogeneity: All links are equally likely to be congested.*

A different problem, related to Boolean Inference, is **Congestion Probability Computation** (from now on, just "Probability Computation" for brevity): given the network graph and the set of congested paths $P^c(t)$, $t = 1..T$, over T consecutive time intervals, compute the *congestion probability* of each set of links, i.e., the probability that all the links in that set are congested [11]. Probability Computation may be a well-posed or ill-posed problem, depending on the assumptions we make.

ASSUMPTION 4. *Independence: All links are independent.*

Under Assumption 4, Probability Computation is well-posed, and there exists an algorithm that solves it [11]: it first computes the congestion probability of each individual link e_i, i.e., $\mathbb{P}(X_{e_i} = 1)$; since links are assumed to be independent, it then derives the congestion probability of each set of links, e.g., $\mathbb{P}(X_{e_i} = 1, X_{e_j} = 1) = \mathbb{P}(X_{e_i} = 1)\, \mathbb{P}(X_{e_j} = 1)$.

[1]Except for the trivial case when end-hosts are directly interconnected.

Symbol	Definition
e_i	The i-th link.
E^*	The set of all links.
$E^c(t)$	The set of congested links during interval t.
X_{e_i}	Random variable associated with link e_i.
p_i	The i-th path.
P^*	The set of all paths.
$P^c(t)$	The set of congested paths during interval t.
Y_{p_i}	Random variable associated with path p_i.
$\mathcal{C}^*$	The set of all correlation sets.

Table 1: Defined symbols.

ASSUMPTION 5. *Correlation Sets: Links are grouped into known* correlation sets*, such that links from the same correlation set may be correlated, but they are always independent from links in other correlation sets.*

We denote by $\mathcal{C}^*$ the set of all correlation sets in the network. We define a *correlation subset* to be a non-empty subset of a correlation set.

CONDITION 2. *Identifiability++: Any two correlation subsets are not traversed by the same paths.*

For example, in Fig. 1, Case 1, this condition holds; in Case 2 it fails, because the sets of links $\{e_1, e_4\}$ and $\{e_2, e_3\}$ belong to different correlation sets, and are both traversed by the same paths, i.e., $\{p_1, p_2, p_3\}$.

Under Assumption 5, if Condition 2 holds, Probability Computation is well-posed [9]; however, there exists no algorithm that solves it completely. The closest result is a heuristic that computes the congestion probability of each individual link [9]. However, as we will see, its accuracy can be significantly improved.

A key question is how does one separate links into correlation sets, i.e., how does one know in advance which links may be correlated with each other. In this paper, we consider the scenario where a source ISP wants to use tomography to monitor its peers, and there is no practical way for it to know which links of each peer may be correlated. Hence, we define one correlation set per Autonomous System (AS), i.e., all links that belong to one AS are assigned to a separate correlation set, and we use an AS-level graph. In short, since we do not know which links of each AS are correlated, we assume that *all* links that belong to the same AS may be correlated. To define correlation sets in this manner, we need to map each link in the network graph to an AS, but no additional information, e.g., correlation factors between different links.

Bayesian Inference is a way to perform Boolean Inference by using Probability Computation as a first step [11]. In particular, this approach poses Boolean Inference as a Maxi-

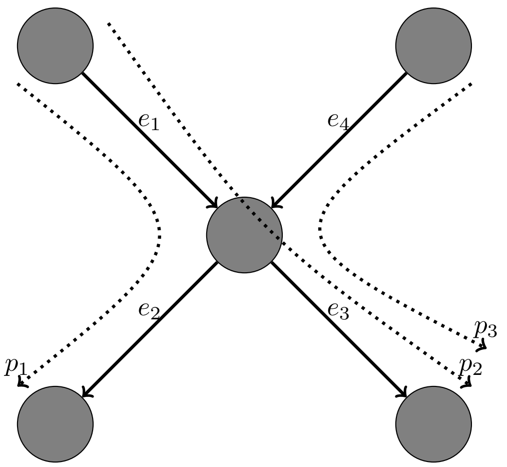


Figure 1: A toy topology.
Links $E^* = \{e_1, e_2, e_3, e_4\}$. Paths $P^* = \{p_1, p_2, p_3\}$. We consider two cases throughout the paper. In Case 1, the correlation sets are $\mathcal{C}^* = \{\{e_1\}, \{e_2, e_3\}, \{e_4\}\}$. In Case 2, the correlation sets are $\mathcal{C}^* = \{\{e_1, e_4\}, \{e_2, e_3\}\}$.

mum Likelihood Estimation (MLE) problem: of all the possible solutions to the Boolean Inference problem, it looks for the one that occurred with the highest probability. Since Probability Computation provides the probability that any set of links is congested, it also provides the probability that any particular solution occurred, e.g., in Fig. 1, the probability that the set of congested links is $\{e_1, e_3\}$ is equal to $\mathbb{P}(X_{e_1} = 1, X_{e_2} = 0, X_{e_3} = 1, X_{e_4} = 0)$.

We state all defined symbols in Table 1.

3. INFERENCE LIMITATIONS

In this section, we look at three inference algorithms for mesh networks: (i) *Sparsity* (originally called Tomo[2]) [6], an adaptation of Duffield's inference algorithm for trees [8] to mesh networks; (ii) *Bayesian-Independence* (originally called CLINK) [11]; and (iii) *Bayesian-Correlation*, a new algorithm that we developed for this work [10]. We experimentally show that neither of them performs accurate inference in the scenario that we are considering. Our point is not that these algorithms are not good (we pick them precisely because they represent the state of the art). Instead, we argue that any inference algorithm is bound to be accurate in some scenarios and inaccurate in others, and there is no evidence that the scenarios favored by one algorithm occur more frequently than those favored by the others.

3.1 Intuition

First, we qualitatively explain through toy examples the sources of inaccuracy in each algorithm.

Sparsity. The gist behind this algorithm is that a few congested links are responsible for many congested paths;

[2]We use new names for the existing algorithms, in order to better distinguish them from each other.

$$\begin{aligned}
\mathbb{P}(Y_{p_1} = 0) &= \mathbb{P}(X_{e_1} = 0)\,\mathbb{P}(X_{e_2} = 0).\\
\mathbb{P}(Y_{p_2} = 0) &= \mathbb{P}(X_{e_1} = 0)\,\mathbb{P}(X_{e_3} = 0).\\
\mathbb{P}(Y_{p_3} = 0) &= \mathbb{P}(X_{e_4} = 0)\,\mathbb{P}(X_{e_3} = 0).\\
\mathbb{P}(Y_{p_2} = 0, Y_{p_3} = 0) &= \mathbb{P}(X_{e_1} = 0)\,\mathbb{P}(X_{e_3} = 0)\,\mathbb{P}(X_{e_4} = 0).\\
\mathbb{P}(Y_{p_1} = 0, Y_{p_2} = 0) &= \mathbb{P}(X_{e_1} = 0)\,\mathbb{P}(X_{e_2} = 0)\,\mathbb{P}(X_{e_3} = 0).\\
\mathbb{P}(Y_{p_1} = 0, Y_{p_3} = 0) &= \mathbb{P}(X_{e_1} = 0)\,\mathbb{P}(X_{e_2} = 0)\,\mathbb{P}(X_{e_3} = 0)\\
&\quad \mathbb{P}(X_{e_4} = 0).
\end{aligned}$$

(a) Bayesian-Independence.

$$\begin{aligned}
\mathbb{P}(Y_{p_1} = 0) &= \mathbb{P}(X_{e_1} = 0)\,\mathbb{P}(X_{e_2} = 0).\\
\mathbb{P}(Y_{p_3} = 0) &= \mathbb{P}(X_{e_3} = 0)\,\mathbb{P}(X_{e_4} = 0).\\
\mathbb{P}(Y_{p_1} = 0, Y_{p_2} = 0) &= \mathbb{P}(X_{e_1} = 0)\,\mathbb{P}(X_{e_2} = 0, X_{e_3} = 0).\\
\mathbb{P}(Y_{p_2} = 0, Y_{p_3} = 0) &= \mathbb{P}(X_{e_1} = 0)\,\mathbb{P}(X_{e_3} = 0)\\
&\quad \mathbb{P}(X_{e_4} = 0).\\
\mathbb{P}(Y_{p_1} = 0, Y_{p_2} = 0, Y_{p_3} = 0) &= \mathbb{P}(X_{e_1} = 0)\,\mathbb{P}(X_{e_4} = 0)\\
&\quad \mathbb{P}(X_{e_2} = 0, X_{e_3} = 0).
\end{aligned}$$

(b) Bayesian-Correlation, for Case 1.

Figure 2: Equations formed by Boolean-Inference algorithms for the example of Fig. 1.

hence, the algorithm which assumes Homogeneity (Assumption 3), "favors" links that participate in more congested paths, i.e., the larger the number of congested paths in which a link participates, the more likely it is to be labeled as congested. For example, in the toy topology of Fig. 1, if the congested paths are $\{p_1, p_2, p_3\}$, Sparsity will infer that the congested links are $\{e_1, e_3\}$ (because each of them participates in two congested paths).

Sparsity works best in scenarios where congestion is concentrated in a few links; this is not the case, for instance, when there exists a lot of congestion at the edge of the network, i.e., many links adjacent to end-hosts are congested at the same time. For example, in Fig. 1, if links e_2 and e_3 are both congested, that will cause the congested paths to be $\{p_1, p_2, p_3\}$, and Sparsity will pick solution $\{e_1, e_3\}$, i.e., it will miss one congested link and falsely blame one good link.

Bayesian-Independence. This is a Bayesian Inference algorithm, i.e., of all the possible solutions, it picks the one that occurs with the highest probability. Hence, this algorithm consists of two steps: (i) *Probability Computation*, which monitors the network and learns the probability with which each solution occurs. (ii) *Probabilistic Inference*, which looks at the status of paths during each particular time interval and determines which set of links were most likely congested during that interval, based on the output of the previous step; this is an NP-complete problem [11], so, this step uses an approximate algorithm. For example, in Fig. 1, if the congested paths are $\{p_1, p_2, p_3\}$, Bayesian-Independence will consider all 8 possible solutions and pick the one that occurs with the highest probability.

The Probability Computation step monitors the status of paths, learns the probability that each set of paths is congested and, from these, computes the probability that each link is congested; under the Independence assumption (Assumption 4), it then computes the probability of each solution. We illustrate with the example of Fig. 1: First, the method computes the probability that p_1 is good, which is equal to the probability that e_1 and e_2 are both good since it assumes links are independent. It forms the first equation in Fig. 2(a). In the same way, it computes the probability that each path and each pair of paths is good and forms the remaining equations in Fig. 2(a). The resulting system has four unknowns (one for each link) and four linearly independent equations, hence, gives us the probability that each link is good. Assuming Independence, we can therefore easily compute the probability of each particular solution, e.g., the probability of solution $E^c = \{e_1, e_3\}$ is $\mathbb{P}(X_{e_1} = 1)\ \mathbb{P}(X_{e_2} = 0)\ \mathbb{P}(X_{e_3} = 1)\ \mathbb{P}(X_{e_4} = 0)$.

Bayesian-Independence needs the Independence assumption, in order to form equations by combining probabilities related to different links. As previous work has already pointed out [9], this assumption does not always hold, which causes Bayesian-Independence to compute some probabilities incorrectly, leading to incorrect inference. For example, suppose that, in Fig. 1, links e_1 and e_4 are always good, while e_2 and e_3 are perfectly correlated (either both are congested or both are good). This means that $\mathbb{P}(X_{e_2} = 0, X_{e_3} = 0) \neq \mathbb{P}(X_{e_2} = 0)\ \mathbb{P}(X_{e_3} = 0)$, and the last two equations in Fig. 2(a) are wrong. As a result, Bayesian-Independence incorrectly determines that $\{e_1, e_3\}$ is the solution with the highest probability and always picks it over the correct one, $\{e_2, e_3\}$.

A more subtle source of inaccuracy in the Probabilistic Inference step is the following: Bayesian-Independence determines whether link e_i was congested during a *particular* time interval based on the probability that link e_i is congested during *any* time interval. More formally, Bayesian-Independence approximates the value of random variable $X_{e_i}(t)$ with its expected value $\mathbb{E}[X_{e_i}(t)] = \mathbb{P}(X_{e_i} = 1)$. We illustrate with an example. Suppose that the Probability Computation module observes the network in Fig. 1 for an hour and computes, among others, the following probabilities:

$$\mathbb{P}(X_{e_1} = 1, X_{e_2} = 0, X_{e_3} = 0, X_{e_4} = 0) = 0.8.$$
$$\mathbb{P}(X_{e_1} = 1, X_{e_2} = 1, X_{e_3} = 0, X_{e_4} = 0) = 0.1.$$

This means that, during the one hour of monitoring, $\{e_1\}$ was the only congested link in the network for 80% of the time, while $\{e_1, e_2\}$ were the only congested links in the network for 10% of the time. Now suppose that during the last 1-minute interval within this hour, the congested paths are $\{p_1, p_2\}$; the Probabilistic Inference module determines that there are two possible solutions for this interval, $\{e_1\}$ and $\{e_1, e_2\}$, and picks the first one (because it has a higher probability associated with it). In essence, Probabilistic Inference determines that this solution is more likely to have occurred *during the last minute*, because it occurred more frequently *over the last hour*.

In practice, we cannot tell whether this approximation works, unless we have "insider information" on network conditions. For example, consider a link that is normally congested very rarely, and Probability Computation correctly computes a low congestion probability for it; suppose this link incurs a technical failure or comes under a flooding attack and becomes severely congested for a few time intervals; unless we already know when this failure/attack occurs and how long it lasts, Probabilistic Inference will not pick this link as congested (because it has a low congestion probability associated with it). So, even if Probability Computation correctly computes for what fraction of time a link is congested, Probabilistic Inference cannot use this information correctly, because it operates at a different time scale.

Finally, the Probabilistic Inference step uses an approximate algorithm to pick the solution that occurred with the highest probability, which means that it does not always pick the right one.

To summarize, Bayesian-Independence introduces three additional sources of inaccuracy: the Independence assumption (used in both steps), the fact that it approximates the value of random variable $X_{e_i}(t)$ with its expected value (in the Probabilistic Inference step), and the use of an approximate algorithm to pick the solution that occurred with the highest probability (also in the Probabilistic Inference step).

Bayesian-Correlation. In an effort to remove one source of inaccuracy, we developed a new inference algorithm that takes into account link correlations. It is similar to Bayesian-Independence (it also consists of a Probability Computation and a Probabilistic Inference step), however, in the former step, instead of Independence, it assumes Correlation Sets (Assumption 5). For instance, in the example of Fig. 1, Case 1, it treats $\mathbb{P}(X_{e_2} = 0, X_{e_3} = 0)$ as an extra unknown, as opposed to mistakenly breaking it into $\mathbb{P}(X_{e_2} = 0)\ \mathbb{P}(X_{e_3} = 0)$, and forms the equations in Fig. 2(b). The resulting system has 5 unknowns (one for each link plus one for the pair of correlated links $\{e_2, e_3\}$) and 5 linearly independent equations, hence, gives us the probability that each set of links is good. Once we know these probabilities, we can compute the probability of each solution [9].

However, taking link correlations into account comes at the price of introducing extra unknowns, and we can compute all of them if and only if the Identifiability++ condition (Condition 2) holds. For instance, in the example of Fig. 1, Case 2, it is impossible to compute the probability that $\{e_1, e_4\}$ are both good or the probability that $\{e_2, e_3\}$ are both good. The intuition is the following: both these sets of links are traversed by the same set of paths, $\{p_1, p_2, p_3\}$; this makes it impossible to distinguish one pair from the other based on path observations and to compute the probability that each pair is good. So, the Probability Computation step of Bayesian-Correlation cannot always compute the probability of all solutions, because the Identifiability++

	Spar.	Bayesian-Indep.		Bayesian-Corr.	
		Step 1	Step 2	Step 1	Step 2
Separability	X	X	X	X	X
E2E Mon.	X	X	X	X	X
Homogeneity	X				
Independence	X	X	X		
Correlation Sets				X	X
Identifiability	X	X	X		
Identif.++				X	X
Other approx./ heuristic	X		X		X

Table 2: Sources of inaccuracy for Boolean Inference algorithms: assumptions, conditions, and approximations/heuristics.

condition does not always hold. As a result, the Probabilistic Inference step does not have all the information it needs to pick the likeliest solution; in the particular example considered above, if the congested paths are $\{p_1, p_2, p_3\}$, it picks at random one of the solutions $\{e_1, e_4\}$ or $\{e_2, e_3\}$.

To summarize, Bayesian-Correlation introduces four additional sources of inaccuracy: the Correlation Sets assumption and—like Bayesian-Independence—the fact that it approximates the values of random variables with their expected values and the use of an approximate algorithm to pick the solution that occurred with the highest probability (in the Probabilistic Inference step).

Conclusion. Each inference algorithm introduces its own sources of inaccuracy (summary in Table 2), and there is no basis for arguing that one algorithm covers more cases than the others.

3.2 Experiments

We now look at the performance of the three Inference algorithms under various scenarios (Fig. 3). We assume that Separability, E2E Monitoring, and Correlation Sets always hold, because this is the weakest set of assumptions under which we can solve Boolean Inference; the rest of the assumptions and conditions in Table 2 may or may not hold, depending on the scenario.

Metrics. We consider two metrics: during a particular time interval, the *detection rate* of an algorithm is the fraction of congested links that the algorithm correctly identified as congested; the *false positive rate* of an algorithm is the fraction of links incorrectly identified as congested out of all links inferred as congested by the algorithm. Each detection rate and false-positive rate we show is an average over 1000 time intervals.

Topologies. We use two kinds of topologies: the *Sparse* topologies are real topologies, given to us by the source ISP; the *Brite* topologies are synthetic topologies.

Each Sparse topology was obtained in the following way. The operator of the source ISP performed traceroutes from a few end-hosts located inside her network toward a large number of external end-hosts; she discarded all incomplete traceroutes. In this way, she collected a router-level graph (where each vertex corresponds to an IP router and each edge corresponds to an IP-level link). Moreover, she mapped each IP router to an Autonomous System (AS) and created an AS-level graph, where each vertex corresponds to a border router and each edge corresponds to an inter-domain link between border routers of peering ASes, or an intra-domain path between two border routers of the same AS. The source ISP wants to monitor its peers at the AS level (it is not interested in each peer's internals), hence, we use the AS-level graph as the network topology. The router-level graph tells us how the links in the AS-level graph are correlated—if a router-level link becomes congested, then all the AS-level links that share this router-level link become congested at the same time.

Each Brite topology also consists of a router-level and an AS-level graph, each derived using the corresponding module of the Brite topology generator [1].

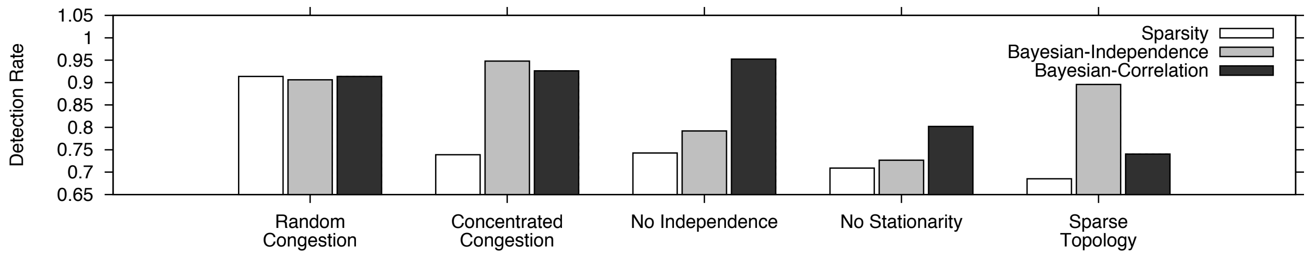
We show results for a representative Sparse topology of about 2000 links and a representative Brite topology of about 1000 links, each of them with 1500 paths—the results for other topologies were similar. The Identifiability++ condition holds only for the Brite topologies.

Simulator. In the beginning of each experiment, we determine the probability that each (AS-level) link is congested and the degree of correlation between congested links (depending on whether they share underlying router-level links). In the experiments that we present here, only 10% of the links are assigned a non-zero congestion probability, which is chosen at random between 0 and 1. Which particular 10% of the links have a non-zero probability of congestion differs, depending on the scenario we are simulating.

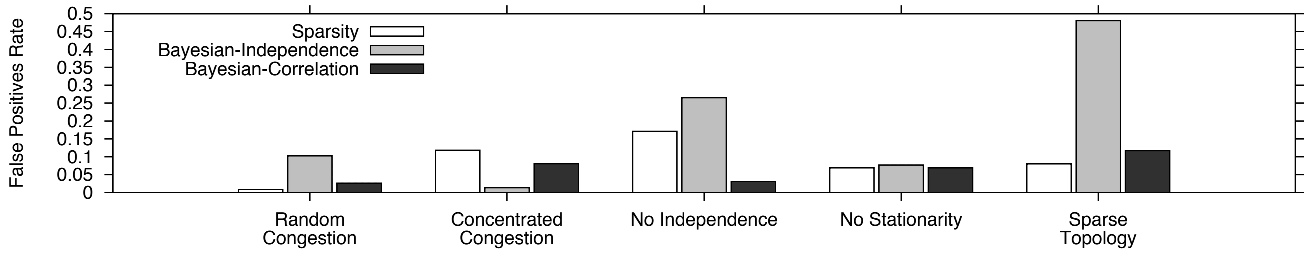
Each experiment consists of multiple time intervals. In the beginning of each interval, we flip a biased coin for each link, to determine whether the link will be good or congested, such that we respect the individual and joint probabilities of congestion determined in the beginning of the experiment; if we determine that a link will be good (resp. congested) in this interval, we randomly assign to it a packet-loss rate between 0 and 0.01 (resp. 0.01 and 1), according to the loss model in [12] (and similar to the loss models in [13, 11]). In each interval, packets are sent along each path; for each packet that arrives at a given link, we flip a biased coin to determine whether it will be dropped or not, such that we respect the packet-loss rate assigned to the link in the beginning of the interval.

Random Congestion (Brite). In this scenario, the 10% of the links that have a non-zero congestion probability are chosen at random. As we see in Fig. 3, all Inference algorithms perform equally well: on average, they identify 90% of the congested links and miss fewer than 2% of them (except from Bayesian-Independence that misses 10%).

The intuition is the following. The Brite topology models a full AS-level topology, hence, it is relatively "dense," i.e., paths tend to criss-cross. This is good for Inference algorithms, because the denser the topology, the higher the rank



(a) Detection Rate.



(b) False Positive Rate.

Figure 3: Performance of inference algorithms under various realistic congestion scenarios, when 10% of the network links have a positive probability of being congested.

of the resulting system of equations and the fewer the possible solutions to each observation—which means that the heuristic/approximate aspect of each algorithm is exercised less. Bayesian-Independence performs slightly worse, because it assumes that links are independent, whereas, in several time intervals, some of the congested links happen to be correlated (share an underlying router-level link).

Concentrated Congestion (Brite). In this scenario, the 10% of the links that have a non-zero probability of congestion are chosen to be located toward the edge of the network, i.e., there is no congestion at the core. As we see in Fig. 3, Sparsity's detection rate drops to 75%, while its false-positive rate rises to 10%. This happens because Sparsity assumes Homogeneity and picks links that are traversed by many congested paths, hence it is more likely to pick solutions that involve links located close to the core of the network. This result does not imply that Sparsity is worse than the other algorithms—just that it performs worse in this particular scenario.

No Independence (Brite). In this scenario, the 10% of the links that have a non-zero probability of congestion are chosen such that each of them is correlated with at least one other. As we see in Fig. 3, Bayesian-Independence's detection rate drops below 80%, while its false-positive rate rises to 25%; this happens because its Probability Computation part assumes Independence, hence learns the probability of each set of links incorrectly.

No Stationarity (Brite). This scenario is similar to the previous one, plus the congestion probabilities of links (the 10% of them, that is) change every few time intervals. As we see in Fig. 3, it is the turn of Bayesian-Correlation's detection rate to drop below 80%; this happens because its Probabilistic Inference step assumes stationarity, i.e., it assumes that a solution is more likely to have occurred during the last time interval, just because it occurred more frequently throughout the entire experiment.

Sparse Topology. This scenario is similar to the first one (the 10% of the links that have a non-zero congestion probability are chosen at random), but is applied to the Sparse as opposed to the Brite topology. As we see in Fig. 3, all Inference algorithms suffer. The fact that Bayesian-Independence has a 90% detection rate should not be mistaken for success: it achieves this by aggressively marking links as congested, which results in a 45% false-positive rate.

The intuition is the following. The Sparse topology was created by running traceroutes from the source ISP to various Internet end-hosts. However, most traceroutes returned incomplete/inconclusive results and had to be discarded, which resulted in a "sparse" view, where few paths intersect one another. This is bad for Inference algorithms, because the sparser the topology, the lower the rank of the resulting system of equations—which means that each algorithm has to rely more on its heuristic/approximate aspect to pick a solution. Note that we did not in any way engineer this scenario to make the Inference algorithms fail as we did in the previous scenarios—we did not introduce extra link correlations or non-stationarity.

We should clarify that the Sparse topology is the most complete topology that the ISP operator was able to collect with the resources (the monitoring points) she had at her disposal. One might argue that, if the operator had done a better job and collected a more complete (less sparse) topology, the Inference algorithms would have performed better. This is true, however, in our experience from working with the operator, piecing together a topology from traceroutes is a complex task—some routers respond to a traceroute probe through a different interface than the one where the probe was received, some routers do not respond to traceroute probes at all, while load-balancing interferes with traceroute results. Hence, we think it is fair to assume that operators are typically not able to collect complete topologies.

Conclusion. Any Inference algorithm can perform badly under certain network conditions, and there is no evidence that such conditions do not occur in practice. Moreover, all Inference algorithms perform badly on Sparse topologies—in particular, each Inference algorithm performs worse on Sparse topologies under easy conditions (random congestion) than on Brite topologies under worst-case conditions (congestion at the edges for Sparsity, link correlations for Bayesian-Independence, and non-stationarity for Bayesian-Correlation).

4. SHIFTING GOALS

Until now, Probability Computation has been viewed as a step to enable Boolean Inference; we argue that, in the scenario considered in this work, Probability Computation is a useful problem to solve in its own right, and that it makes more sense to solve this problem than Boolean Inference.

If accurately solved, Boolean Inference would provide the source ISP with the status of each link in the network during each time interval. This information would enable the source ISP to attribute blame to a peer for a particular connectivity/performance problem faced by the source ISP's customers and/or request compensation in case an SLA has been violated. However, as we saw in the last section, given the Sparse topology, state-of-the-art Inference algorithms yield a detection rate as low as 68% and a false-positive rate as high as 47%; attributing blame or extracting compensation is practically impossible based on this level of accuracy.

Probability Computation provides less information than Boolean Inference: if accurately solved, it would provide the source ISP with the congestion probability of each set of links in the network, i.e., *how frequently* each set of links are congested, but not *which particular* links were congested *when*.

On the other hand, we have an algorithm (Step 1 of Bayesian-Correlation) that solves Probability Computation with fewer sources of inaccuracy than Boolean Inference algorithms:

▷ It assumes Separability, E2E Monitoring, and Correlation Sets—a weaker set of assumptions than those assumed by Sparsity and Bayesian-Independence.

▷ Unlike the Bayesian Inference algorithms (Bayesian-Independence and Bayesian-Correlation), our algorithm does not need to solve any NP complete problem.

▷ Unlike the Bayesian Inference algorithms, our algorithm does not need to approximate the value of any random variable with its expected value: if we compute that $\mathbb{P}(X_{e_i} = 1) = 0.8$ over T time intervals, we interpret this as "e_i was congested for 80% of the T time intervals." In contrast, the Bayesian Inference algorithms use the same information to infer during which particular intervals e_i was congested. When network conditions change over time, the Bayesian Inference algorithms may make the wrong decision (§3.1); our result, however, still holds, because it concerns the *average* behavior of the link over the T time intervals, and not the diagnosis of the congested links over one time interval.

The biggest challenge in solving Probability Computation is complexity: there are as many unknowns as sets of links that belong to the same AS; in a real network, there may be billions of such sets, and it could take an impractical amount of time to solve the corresponding system of equations. Hence, we design our algorithm to accurately compute a *configurable* subset of the computable probabilities, depending on the available resources. For instance, we can configure our algorithm to compute only the congestion probability of each individual link, or the congestion probability of each set of one, two, or three links. This allows us to control the complexity of the algorithm and obtain useful information in a timely manner, even if we do not wait to solve Probability Computation completely (i.e., we do not learn the congestion probability of all sets of links).

In the next section, we support these claims with experimental results.

5. PROBABILITY COMPUTATION

In this section, we present an algorithm that solves the Probability Computation problem, assuming Separability, E2E Monitoring, and Correlation Sets (Assumptions 1, 2, and 5). In particular, our algorithm computes the congestion probability of each correlation subset for which the following is true: there exists no other correlation subset in the network that is traversed by the same paths. When the Identifiability++ condition holds, this is true for all correlation subsets, which means that our algorithm computes the congestion probability of each set of links in the network.

After introducing a basic building block of the algorithm (§5.1) and our terminology and notation (§5.2), we describe the algorithm itself (§5.3), and evaluate it experimentally (§5.4).

5.1 A Basic Building Block

Informally, our algorithm forms a system with enough linearly independent equations to compute as many congestion probabilities as possible. Each equation corresponds to a different set of paths, in the following way: consider a set of paths P; by the Separability assumption, if all paths in P are good, then all links traversed by the paths in P (denoted by

Links (P)) are good; hence, we can write:

$$\mathbb{P}\left(\bigcap_{p\in P} Y_p = 0\right) = \mathbb{P}\left(\bigcap_{e\in Links(P)} X_e = 0\right)$$
$$= \prod_{\mathcal{C}\in\mathcal{C}^*} \mathbb{P}\left(\bigcap_{e\in Links(P)\cap\mathcal{C}} X_e = 0\right). \qquad (1)$$

In Fig. 1, Case 1, if we apply Eq. 1 to path set $\{p_1\}$, we obtain:

$$\mathbb{P}(Y_{p_1} = 0) = \mathbb{P}(X_{e_1} = 0, X_{e_2} = 0)$$
$$= \mathbb{P}(X_{e_1} = 0)\ \mathbb{P}(X_{e_2} = 0). \qquad (2)$$

If we apply Eq. 1 to path set $\{p_1, p_2\}$, we obtain:

$$\mathbb{P}(Y_{p_1} = 0, Y_{p_2} = 0) = \mathbb{P}(X_{e_1} = 0, X_{e_2} = 0, X_{e_3} = 0)$$
$$= \mathbb{P}(X_{e_1} = 0)\ \mathbb{P}(X_{e_2} = 0, X_{e_3} = 0). \qquad (3)$$

We could form as many such equations as there are sets of paths in the network. A naïve approach would be to consider all $2^{|P^*|}$ possible sets of paths in the network, form a system of that many equations, reduce this to a system of linearly independent equations[3], and solve the latter. However, processing $2^{|P^*|}$ equations is practically infeasible for any topology with more than a few tens of paths (our Sparse topology has roughly 1500 paths). We address this challenge by using a novel technique that forms the minimum number of equations needed, without considering all possible sets of paths.

5.2 Definitions and Notation

▷ The *path coverage* function *Paths* (E) maps a set of links E to the set of paths that traverse at least one of these links. In Fig. 1, *Paths* $(\{e_1, e_2\}) = \{p_1, p_2\}$, *Paths* $(\{e_1, e_3\}) = \{p_1, p_2, p_3\}$.

▷ The *link coverage* function *Links* (P) maps a set of paths P to the set of links traversed by at least one of these paths. In Fig. 1, *Links* $(\{p_1\}) = \{e_1, e_2\}$, *Links* $(\{p_1, p_2\}) = \{e_1, e_2, e_3\}$.

▷ A *correlation subset* $\mathcal{E}$ is a non-empty subset of a correlation set, i.e., $\mathcal{E} \subseteq \mathcal{C}$ for some correlation set $\mathcal{C}$. We often refer to "all the possible correlation subsets" in the network; in Fig. 1, Case 1, these are $\{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}, \{e_2, e_3\}$; in Case 2, they are $\{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}, \{e_2, e_3\}, \{e_1, e_4\}$.

▷ We define the *complement* of a correlation subset $\mathcal{E}$ that belongs to a correlation set $\mathcal{C}$ as $\bar{\mathcal{E}} = \mathcal{C} \setminus \mathcal{E}$. In Fig. 1, Case 1, $\overline{\{e_1\}} = \emptyset$, $\overline{\{e_2\}} = \{e_3\}$, and $\overline{\{e_3\}} = \{e_2\}$, $\overline{\{e_4\}} = \overline{\{e_2, e_3\}} = \emptyset$.

▷ A correlation subset $\mathcal{E}$ is *potentially congested* if none of its links is traversed by a path that is always good, i.e., every path that traverses a link in $\mathcal{E}$ is congested during at least one time interval. In Fig. 1, Case 1, suppose path p_3 is always good, whereas the other two paths are not; this means that links e_3 and e_4 are always good, hence, the potentially congested correlation subsets are $\{e_1\}$ and $\{e_2\}$. The congestion probability of any correlation subset that is not potentially congested is 0.

▷ If P is a set of paths and $\hat{\mathcal{E}}$ is an ordering of all the potentially congested correlation subsets, we define the vector $Row(P, \hat{\mathcal{E}})$ as follows: we apply Eq. 1 to the set of paths P and we form a vector $\mathbf{r}$, where

$$\mathbf{r}_i = \begin{cases} 1, & \text{if the } i\text{-th correlation subset in } \hat{\mathcal{E}} \text{ appears} \\ & \text{in the equation for path set } P \\ 0, & \text{otherwise.} \end{cases}$$

$Row(P, \hat{\mathcal{E}})$ is equal to $\mathbf{r}$.

▷ If $\hat{P}$ is an ordering of a set of path sets and $\hat{\mathcal{E}}$ is an ordering of all the potentially congested correlation subsets, we define the matrix $Matrix(\hat{P}, \hat{\mathcal{E}})$ as follows: the i-th row of the matrix is equal to $Row(P_i, \hat{\mathcal{E}})$, where P_i is the i-th path set in $\hat{P}$. For example, in Fig. 1, Case 1, suppose all correlation subsets are potentially congested; given an ordering of these correlation subsets $\hat{\mathcal{E}} = \langle\{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}, \{e_2, e_3\}\rangle$ and an ordering of path sets $\hat{P} = \langle\{p_1\}, \{p_1, p_2\}\rangle$,

$$Matrix(\hat{P}, \hat{\mathcal{E}}) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

This matrix corresponds to the system of equations 2 and 3.

5.3 The Algorithm

To compute the congestion probability of each set of links, it is sufficient to compute the congestion probability of each potentially congested correlation subset—or, equivalently, for each potentially congested correlation subset $\mathcal{E}$, compute the probability that all links in that subset are good [9], i.e.,

$$\mathbb{P}\left(\bigcap_{e\in\mathcal{E}} X_e = 0\right).$$

For instance, in Fig. 1, Case 1, if all the correlation subsets are potentially congested, it is sufficient to compute $\mathbb{P}(X_{e_1} = 0)$, $\mathbb{P}(X_{e_2} = 0)$, $\mathbb{P}(X_{e_3} = 0)$, $\mathbb{P}(X_{e_4} = 0)$, and $\mathbb{P}(X_{e_2} = 0, X_{e_3} = 0)$.

To compute these probabilities, our algorithm (Alg. 1) forms as many linearly independent equations as it can by applying Eq. 1 to different path sets. The input to the algorithm is an ordering $\hat{\mathcal{E}}$ of all the potentially congested correlation subsets that can appear in such equations.[4] The output is an ordering of path sets $\hat{P}$ to which we apply Eq. 1 to form a system of equations.

First, we form an initial list of path sets $\hat{P}$ (lines 1 to 5). We ensure that each correlation subset $\mathcal{E} \in \hat{\mathcal{E}}$ is traversed by at least one of the path sets in $\hat{P}$, namely path

[3]If we consider the logarithm of Eq. 1, we obtain a linear equation.

[4]When Identifiability++ does not hold, there may exist correlation subsets that cannot appear in any equation. We give an example in our technical report [10].

Algorithm 1 *Selection of Path Sets*

```
Input:      $\hat{\mathcal{E}}$:  a list of potentially congested
                 correlation subsets
Variables:  $\hat{P}$:  a list of path sets
            $P$:  a path set
            $\mathcal{E}$:  a correlation subset

1: $\hat{P} \leftarrow \langle\rangle$
2: for all $\mathcal{E} \in \hat{\mathcal{E}}$ do
3:     $P \leftarrow Paths\left(\mathcal{E}\right) \setminus Paths\left(\bar{\mathcal{E}}\right)$
4:     $\hat{P} \leftarrow \hat{P} + P$
5: end for

6: $\mathbf{R} \leftarrow Matrix(\hat{P}, \hat{\mathcal{E}})$
7: $\mathbf{N} \leftarrow NullSpace\left(\mathbf{R}\right)$

8: repeat
9:     $\mathbf{r} \leftarrow \mathbf{0}$
10:    for all $\mathcal{E} \in SortByHammingWeight(\hat{\mathcal{E}}, \mathbf{N})$ do
11:        for all $P \subseteq Paths\left(\mathcal{E}\right) \setminus Paths\left(\bar{\mathcal{E}}\right)$ do
12:            $\mathbf{r} \leftarrow Row(P, \hat{\mathcal{E}})$
13:            if $||\mathbf{r} \times \mathbf{N}|| > 0$ then
14:                $\hat{P} \leftarrow \hat{P} + P$
15:                go to line 21
16:            else
17:                $\mathbf{r} \leftarrow \mathbf{0}$
18:            end if
19:        end for
20:    end for
21:    $\mathbf{N} \leftarrow NullSpaceUpdate\left(\mathbf{N}, \mathbf{r}\right)$
22: until N has no columns left or $\mathbf{r} = \mathbf{0}$

23: return $\hat{P}$

Notation:
$\mathcal{A} \setminus \mathcal{B}$:   subtract set $\mathcal{B}$ from set $\mathcal{A}$
$\hat{P} + P$:   add path set $P$ to list of path sets $\hat{P}$
```

set $Paths\left(\mathcal{E}\right) \setminus Paths\left(\bar{\mathcal{E}}\right)$ (lines 2 and 3). We illustrate with an example. Suppose that, in Fig. 1, Case 1, all correlation subsets are potentially congested and we pick ordering $\hat{\mathcal{E}} = \langle\{e_1\}, \{e_2\}, \{e_3\}, \{e_4\}, \{e_2, e_3\}\rangle$. After line 5 has been executed, $\hat{P}$ consists of the path sets in the last column of the following table:

$\mathcal{E}$	$\bar{\mathcal{E}}$	$Paths\left(\mathcal{E}\right)$	$Paths\left(\bar{\mathcal{E}}\right)$	$Paths\left(\mathcal{E}\right) \setminus Paths\left(\bar{\mathcal{E}}\right)$
$\{e_1\}$	$\emptyset$	$\{p_1, p_2\}$	$\emptyset$	$\{p_1, p_2\}$
$\{e_2\}$	$\{e_3\}$	$\{p_1\}$	$\{p_2, p_3\}$	$\{p_1\}$
$\{e_3\}$	$\{e_2\}$	$\{p_2, p_3\}$	$\{p_1\}$	$\{p_2, p_3\}$
$\{e_4\}$	$\emptyset$	$\{p_3\}$	$\emptyset$	$\{p_3\}$
$\{e_2, e_3\}$	$\emptyset$	$\{p_1, p_2, p_3\}$	$\emptyset$	$\{p_1, p_2, p_3\}$

If we apply Eq. 1 to each of the path sets in $\hat{P}$, we obtain the system of equations shown in Fig. 2(b). In this particular example, the corresponding matrix $Matrix(\hat{P}, \hat{\mathcal{E}})$ has full column rank, which means that we can solve our system and compute, for each correlation subset, the probability that all links in that subset are good, hence also the congestion probability of each set of links in the network. In general, however, the resulting system of equations is under-determined, and we continue with the second part of the algorithm.

We augment the initial list of path sets $\hat{P}$ by iteratively

Algorithm 2 *NullSpaceUpdate*

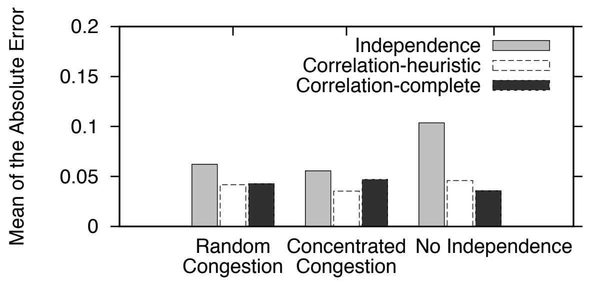
```
Input:  N:  a matrix of size $n \times p$
        r:  a row vector of $n$ elements

1: return $\left(\mathbf{I}_n - \frac{\mathbf{N}_{*1} \times \mathbf{r}}{\mathbf{r} \times \mathbf{N}_{*1}}\right) \mathbf{N}_{*2:p}$

Notation:
$\mathbf{I}_n$:        the identity matrix of size $n$
$\mathbf{N}_{*1}$:     the 1-st column of matrix N
$\mathbf{N}_{*2:p}$:   the matrix formed by taking columns
                       2 to $p$ of N
```

adding path sets, such that we increase the rank of the associated matrix $Matrix(\hat{P}, \hat{\mathcal{E}})$ (lines 6 to 22). More specifically, we first compute the matrix **R** associated with the initial list of path sets $\hat{P}$ (line 6), as well as a matrix **N**, whose columns span the null space of **R** (line 7); the latter can be done using standard techniques, like singular value decomposition or QR factorization. Next, we iteratively identify a path set P such that adding $\mathbf{r} = Row(P, \hat{\mathcal{E}})$ to the system matrix, increases the latter's rank, and we add P to $\hat{P}$ (lines 12 to 15). Every time we add a new path set to $\hat{P}$, we update the matrix **N**, such that its columns always span the null space of $Matrix(\hat{P}, \hat{\mathcal{E}})$ (line 21). We stop the iteration when **N** is left with 0 columns or there are no more path sets left to consider, i.e., the loop in line 10 finishes (line 22).

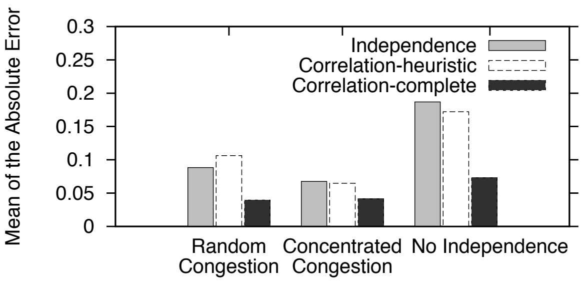
The first point worth noting is that identifying a new set of paths P such that $Row(P, \hat{\mathcal{E}})$ increases the rank of the system matrix is not straightforward. If such a set of paths exists, our algorithm finds it, because it iterates over all sets of paths (lines 10 and 11) and tests whether each of them satisfies the corresponding condition (lines 12 and 13). However, to save time, the algorithm orders the sets of paths such that it first tries those that are more *likely* to satisfy the condition (this is the role of the *SortByHammingWeight* function). Intuitively, if the i-th element of vector **r** is non-zero and the i-th row of matrix **N** has many non-zero elements, then $||\mathbf{r} \times \mathbf{N}|| > 0$ is likely to be true. Thus, our algorithm picks the row of **N** with the largest number of non-zero elements (the largest Hamming weight); suppose that this row corresponds to a correlation subset $\mathcal{E}$ (line 10). Then, it looks for any path set P that traverses $\mathcal{E}$ (line 11), and picks the first one that satisfies the condition (lines 12 to 15). The *SortByHammingWeight* helps us pick the correlation subset $\mathcal{E}$—it outputs an ordering of the correlation subsets in $\hat{\mathcal{E}}$ such that the first element in that ordering corresponds to the row of matrix **N** with the largest Hamming weight.

The second point worth noting about the algorithm is that computing the null space of a matrix with thousands of rows takes a significant amount of time, and doing this at every iteration would render the algorithm practically useless. Instead, the *NullSpaceUpdate* function (Alg. 2) updates the null space incrementally, i.e., given the null space computed in the previous iteration, it efficiently updates the null space.

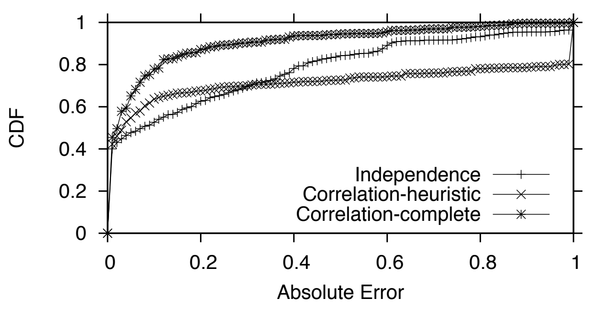
Complexity. The complexity of our algorithm is $\mathcal{O}(n_1^3 + n_1^2 \cdot 2^{n_2} \cdot n_3)$, where $n_1 = |\hat{\mathcal{E}}|$ is the number of potentially



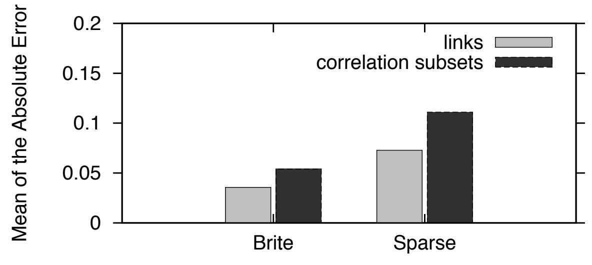
(a) The mean of the absolute error under various congestion scenarios, when computing the congestion probability of individual links. Brite topologies.



(b) The mean of the absolute error under various congestion scenarios, when computing the congestion probability of individual links. Sparse topologies.



(c) CDF of the absolute error for the "No Independence" congestion scenario, when computing the congestion probability of individual links. Sparse topologies.



(d) The mean of the absolute error of the Correlation-complete algorithm in the "No Independence" scenario, when computing the congestion probability of individual links *and* correlation subsets.

Figure 4: Performance of Probability Computation algorithms for Brite and Sparse topologies.

congested correlation subsets, $n_2 = \max_{\mathcal{E} \in \hat{\mathcal{E}}} |Paths(\mathcal{E})|$ is the maximum number of paths that traverse the same potentially congested correlation subset, n_3 is the nullity of the initial system matrix $Matrix(\hat{P}, \hat{\mathcal{E}})$. We express complexity as a function of these three parameters, because any one of them can dominate the other two, depending on the topology and the congestion scenario. The proof can be found in our technical report [10].

5.4 Experiments

We now look at the performance of our algorithm (we label it *Correlation-complete*) and compare it to the two most related pieces of work: (i) *Independence* [11], which is the Probability Computation step of the Bayesian-Independence algorithm; (ii)*Correlation-heuristic* [9], an earlier heuristic that, under the Correlation Sets assumption, computes the probability that each individual link is congested.

We reconsider all congestion scenarios described in Section 3.2: Brite (Fig. 4(a)) and Sparse (Fig. 4(b)) topologies, where 10% of the links have a non-zero probability of being congested. The links with a non-zero congestion probability are chosen either at random (Random Congestion), or chosen to be located toward the edge of the network (Concentrated Congestion), or chosen such that each of them is correlated with at least one other link (No Independence). In addition, in each of these scenarios, the congestion probability of each link changes every few time intervals, i.e., we add the "No Stationarity" scenario on top of each of the above scenarios described in Section 3.2.

First, we evaluate how accurately each algorithm computes the congestion probability of each individual link. For each link, we determine the absolute error between the actual congestion probability (the one assigned by the simulator) and the one inferred by each algorithm; we show the mean of the absolute error for all potentially congested links, i.e., all links which are not traversed by any good path.

For the Brite topologies (Fig. 4(a)), in the "Random Congestion" and "Concentrated Congestion" scenarios, all algorithms perform well, with a mean absolute error below 0.07; for the "No Independence" scenario, the error of the Independence algorithm doubles compared to that of the alternatives (because it ignores link correlations).

For the Sparse topologies (Fig. 4(b)), the performance of the Correlation-heuristic and Independence algorithms degrades, because these algorithms create a significantly larger number of equations than ours, which introduces more noise when solving the system. For the "No Independence" scenario, the mean of the absolute error of the Independence algorithm is 3 times larger than that achieved by our algorithm. Since this scenario is the most challenging for all algorithms, we also show the cumulative distribution function (CDF) of the absolute error for each algorithm (Fig. 4(c)). For a perfect algorithm, this CDF would be a single point at $x = 0, y = 100\%$, i.e., the algorithm would compute each congestion probability with an absolute error of 0. In general, the earlier the CDF hits the $y = 100\%$ line, the better the performance of the corresponding algorithm. Correlation-heuristic is able to compute accurately (with an absolute error below 0.1) the congestion probability of only 65% of the links, while the Independence algorithm is more uniformly inaccurate for 50% of the links. Our algorithm does better, with an absolute error less than 0.1 for 80% of the links.

We also show how accurately our algorithm computes the congestion probabilities of different *sets* of links, not just individual links (Fig. 4(d)). Knowing these probabilities reveals which links within each peer are actually correlated; this can be useful for computing "disjoint" paths to some destination, i.e., paths that are not likely to fail at the same time. Our algorithm performs well: even in the "No Independence" scenario, it accurately computes the congestion probability of all correlation subsets for Brite topologies, or a significant number (depending on available resources) of correlation subsets for Sparse topologies, with a mean absolute error of 0.1 or less.

Conclusion. The Probability Computation problem can be solved accurately, even on sparse topologies with link correlations and non-stationary network conditions.

6. RELATED WORK

Network performance tomography, which infers link characteristics from end-to-end path measurements, is an ill-posed inverse problem that has been well studied in the last decade.

The initial methods relied on temporal correlation to infer the loss rate of network links, either by sending multicast probe packets (which are perfectly correlated on multicast links) [4, 3, 2], or by sending unicast probe packets back to back (which are strongly correlated on shared links), as an emulation of multicast packets [5, 7]. Multicast is not widely deployed, and groups of unicast packets require substantial development and administrative costs, hence it is not easy to rely on temporal correlations.

The set of methods [12, 8, 11] that followed use only unicast end-to-end flows for the simpler goal of identifying the congested links (i.e., identifying if the link loss rate or delay exceeds some threshold, instead of computing their actual value). As different assignments are possible, these "Boolean" network-tomographic methods use additional information or assumptions. In Sections 2 and 3, we discussed these assumptions and their practical impact in detail.

We show that, in the scenario of an ISP that wants to monitor the performance of its peers, it is more useful to compute the probabilities that links are congested rather than identify the congested links. We propose an algorithm that computes these probabilities without assuming link independence (as opposed to the method in [11]) and achieves significantly higher accuracy for sparse topologies than the heuristic proposed in [9].

7. CONCLUSION

We considered a real scenario where network performance tomography could be useful: a Tier-1 ISP wants to monitor the congestion status of its peers. In principle, this could be achieved using Boolean Inference; in practice, in turned out that, in this scenario, Boolean Inference cannot be solved accurately enough to be useful. We argued that it makes more sense to solve the Congestion Probability Computation problem—compute how frequently each peer's links are congested as opposed to infer which particular links are congested when. We presented an algorithm that solves this problem accurately under weaker assumptions than those required by Boolean Inference and more challenging network conditions (sparse topologies, link correlations, and non-stationary network dynamics).

Acknowledgments
This work was supported by grant ManCom 2110 of the Hasler Foundation, Bern, Switzerland, and by an Ambizone grant from the Swiss National Science Foundation. We would like to thank our shepherd Rocky K.C. Chang and the anonymous reviewers for their constructive feedback.

8. REFERENCES

[1] Boston University Representative Internet Topology Generator. `http://www.cs.bu.edu/brite/`.

[2] A. Adams, T. Bu, T. Friedman, J. Horowitz, D. Towstey, R. Caceres, N. Duffield, F. L. Presti, S. B. Moon, and V. Paxson. The Use of End-to-end Multicast Measurements for Characterizing Internal Network Behavior. *IEEE Communications Magazine*, May 2000.

[3] T. Bu, N. Duffield, F. L. Presti, and D. Towsley. Network Tomography on General Topologies. In *Proceedings of the ACM SIGMETRICS Conference*, 2002.

[4] R. Caceres, N. G. Duffield, J. Horowitz, and D. Towsley. Multicast-based Inference of Network-Internal Loss Characteristics. *IEEE Transactions on Information Theory*, 45:2462–2480, 1999.

[5] M. Coates and R. Nowak. Network Loss Inference Using Unicast End-to-End Measurement. In *Proceedings of the ITC Specialist Seminar on IP Traffic Measurement, Modeling and Management*, 2000.

[6] A. Dhamdhere, R. Teixeira, C. Drovolis, and C. Diot. Netdiagnoser: Troubleshooting network unreachabilities usind end-to-end probes and routing data. In *Proceedings of ACM Conext*, 2007.

[7] N. Duffield, F. L. Presti, V. Paxson, and D. Towsley. Inferring Link Loss Using Striped Unicast Probes. In *Proceedings of the IEEE INFOCOM Conference*, 2001.

[8] N. G. Duffield. Network Tomography of Binary Network Performance Characteristics. *IEEE Transactions on Information Theory*, 52(12):5373–5388, December 2006.

[9] D. Ghita, K. Argyraki, and P. Thiran. Network Tomography on Correlated Links. In *Proceedings of the ACM IMC Conference*, 2010.

[10] D. Ghita, K. Argyraki, and P. Thiran. Rethinking boolean network tomography. Technical report, EPFL, 2011.

[11] H. X. Nguyen and P. Thiran. The Boolean Solution to the Congested IP Link Location Problem: Theory and Practice. In *Proceedings of the IEEE INFOCOM Conference*, 2007.

[12] V. N. Padmanabhan, L. Qiu, and H. J. Wang. Server-based Inference of Internet Performance. In *Proceedings of the IEEE INFOCOM Conference*, 2003.

[13] H. H. Song, L. Qiu, and Y. Zhang. NetQuest: A Flexible Framework for Large-Scale Network Measurement. In *Proceedings of the ACM SIGMETRICS Conference*, 2006.