Inference Algorithms Analysis

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

ABSTRACT

Boolean network tomography establishes relationships between the congestion status of network links and those of end-to-end paths. These relationships are used by inference algorithms to determine which network links are congested at any point in time.

Our work shows that no state-of-the-art inference algorithm can determine accurately the congested links from end-to-end measurements in all possible realistic scenarios. We assert that it is more beneficial, and accurate to determine the probability that any set of network links is congested, rather than the congestion status of any link.

We present a scalable algorithm that is able to compute the probability that any set of network links is congested, for all possible sets of links, provided certain well defined conditions hold. Unlike state-of-the art algorithms with a similar goal, our algorithm avoids iteratively searching the solution space by using a novel approach that makes our algorithm complete, scalable and fast. We discuss how our algorithm can be used by a large ISP to monitor the performance of its peers.

1. INTRODUCTION

In principle, 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 appeal of the approach is that, in principle, 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.

On the other hand, there are reasons to be skeptical about the applicability of this approach in practice. Tomographic algorithms necessarily make assumptions that cannot be verified in a real network, which means that their results may be inaccurate and, most importantly, there is no way to tell *to what extent* they are inaccurate.

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 Internet-wide BGP incidents, 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 an ideal match for Boolean Inference algorithms [4, 5, 2, 1, 3], which monitor end-to-end paths during a particular time interval (on the order of a few minutes) and infer which particular links 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. 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 Inference 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

[*]Currently at Bilkent University,Turkey.

complexity, yet brought little benefit—in the end, all Inference algorithms performed very well under certain conditions (dense topologies, randomly congested links, link independence, stationary network dynamics) and equally badly under the opposite conditions, which are the ones that interest us. We concluded that Boolean Inference cannot be solved with sufficient accuracy to be useful, at least not in the practical context of this work.

Instead, we argue that 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 would learn 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, we make two contributions:

(i) We show that, in the practical scenario where an ISP wants to monitor the behavior and performance of its peers, Boolean Inference cannot be solved accurately enough to be useful. We do not attribute this to the limitations of existing Inference algorithms; our point is that any Inference algorithm can fail under certain condition, 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. We present a new algorithm that solves this problem and show that it is accurate under weaker assumptions than those required by Boolean Inference and the network conditions imposed by our scenario—sparse topologies, link correlations, non-stationary network dynamics.

2. SETUP

2.1 Network Model

Links and Paths.. We model the network as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each node $v_i \in \mathcal{V}$ represents a network element, either a host or a router, while each edge $e_k \in \mathcal{E}$ represents a logical link between two network elements. We define a "path" as a sequence of links, and we denote the set of paths in the network by $\mathcal{P}$. If a path $P_i \in \mathcal{P}$ traverses a link $e_k \in \mathcal{E}$, then we write $e_k \in P_i$. A path never crosses a link more than once, i.e., there are no loops. All links participate in at least one path, i.e., there are no unused links.

We define the "path coverage" function ψ, which maps a set of links $A \subseteq \mathcal{E}$ to the set of paths that traverse at least one of these links:

$$\psi(A) = \{\ P_i \in \mathcal{P} \mid P_i \ni e_k \text{ for some } e_k \in A\ \}. \quad (1)$$

Similarly, we define the "edge coverage" function ε, which maps a set of paths $L \subseteq \mathcal{P}$ to the set of edges traversed by these paths:

$$\varepsilon(L) = \{\ e_k \in \mathcal{E} \mid e_k \in P_i \text{ for some } P_i \in L\ \}. \quad (2)$$

Congestion.. We divide time into even slots called "snapshots," such that each experiment involves a finite sequence of N snapshots.

We model the congestion behavior of link e_k during an experiment as a stationary random process: We define e_k's "packet-loss rate" during the n-th snapshot of the experiment as the fraction of packets that are not delivered to their next link out of all the packets that arrive at e_k during the snapshot. We say that e_k is "congested" (resp. "good") during the n-th snapshot, if its packet-loss rate is above (resp. below or equal to) a threshold t_l (we follow the model proposed in [2], where all links have the same link-congestion threshold t_l). For each link e_k and each snapshot n, we define a random variable $X_{e_k}(n)$ as follows:

$$X_{e_k}(n) = \begin{cases} 1, & \text{if } e_k \text{ is congested during the } n\text{-th snapshot} \\ 0, & \text{otherwise.} \end{cases}$$

For a given link e_k, all random variables $X_{e_k}(n), n = 1..N$, are identically distributed; hence, for simplicity, we use X_{e_k} to denote any of them. This implies that the congestion probability of each link remains the same throughout the experiment. The scenario in which link e_k is always congested (resp. good) is a special, degenerate case of our model, where X_{e_k} is always 1 (resp. 0). We denote the state of all links in the network by $\mathbf{X} = \{X_{e_k}\}_{e_k \in \mathcal{E}}$.

Similarly, we model the congestion behavior of path P_i during an experiment as a stationary random process: We define P_i's "packet-loss rate" during the n-th snapshot of the experiment as the fraction of packets that are not delivered to their destination out of all packets sent on P_i during that snapshot. We say that P_i is "congested" (resp. "good") during the n-th snapshot, if its packet-loss rate is above (resp. below or equal to) a threshold $t_p = 1 - (1 - t_l)^d$, where d is the number of links traversed by P_i. For each path P_i and each snapshot n, we define a random variable $Y_{P_i}(n)$ as follows:

$$Y_{P_i}(n) = \begin{cases} 1, & \text{if } P_i \text{ is congested during the } n\text{-th snapshot} \\ 0, & \text{otherwise.} \end{cases}$$

For a given path P_i, all random variables $Y_{P_i}(n), n = 1..N$, are identically distributed; hence, for simplicity, we use Y_{P_i} to denote any of them. Finally, Y_{P_i} has expected value $E(Y_{P_i}) = \mathbb{P}(Y_{P_i} = 1)$, equal to the fraction of snapshots during which P_i is congested. In other words, if, during the experiment, we observe that P_i is congested during half of the snapshots, we model this by saying that $\mathbb{P}(Y_{P_i} = 1) = 0.5$. The scenario in which path P_i is always congested (resp. good) is a special, degenerate case of our model, where Y_{P_i} is always 1 (resp. 0). We denote the state of all paths in the network by $\mathbf{Y} = \{Y_{P_i}\}_{P_i \in \mathcal{P}}$.

***Link Correlation.*.** We say that two links e_k and e_l are "independent" or "uncorrelated" when the random variables X_{e_k} and X_{e_l} are independent (or, equivalently, since they are Bernoulli random variables, uncorrelated) from one another; this corresponds to the situation where e_k's congestion status during a snapshot cannot affect e_l's congestion status during the same or any other snapshot. Otherwise, we say that the two links are "correlated."

We assume that each link may be correlated with a specific set of other links, and we assume that we know which links may be correlated with one another. Based on this knowledge, we group links into "correlation sets," such that two links from the same correlation set may be correlated with one another, but not with links from other correlation sets. More formally, consider a partition $\mathcal{C} = \{C_1, C_2, ...C_{|\mathcal{C}|}\}$ of $\mathcal{E}$, such that:

$$\begin{aligned} &\forall e_k, e_l,\ k \neq l, \quad \text{s.t.} \quad e_k \in C_p,\ e_l \in C_q,\ p \neq q, \\ &\qquad e_k \text{ is uncorrelated with } e_l; \end{aligned}$$

we call each set of links $C_p \in \mathcal{C}$ a "correlation set."

In our analysis, we will often refer to a "correlation subset" $A \subseteq C_p$, $A \neq \emptyset$, i.e., a non-empty subset of a correlation set. We will also refer to the set of all possible correlation subsets:

$$\tilde{\mathcal{C}} = \{\ A \subseteq \mathcal{E} \mid A \neq \emptyset \text{ and } A \subseteq C_p \text{ for some } C_p \in \mathcal{C}\ \}.$$

Given a correlation subset $A \in \tilde{\mathcal{C}}$, we call $\mathbb{P}(\cap_{e_j \in A} X_{e_j} = 1)$ the "congestion probability of A", i.e., the probability that all links in A are congested. We say that a set of paths $L \subseteq \mathcal{P}$ "covers" correlation subset $A \subseteq C_p$, if and only if

$$A \subseteq \varepsilon(L) \text{ and } \varepsilon(L) \cap (C_p \setminus A) = \emptyset,$$

i.e., the paths in L traverse all links in A, but do not traverse any link in $C_p \setminus A$.

2.2 Theoretical Results

We make three assumptions that are inherited from previous work on tomography.

ASSUMPTION 1. *Stability: The set of paths $\mathcal{P}$ remains unchanged during each experiment.*

This assumption is common in all tomographic algorithms. In practice, if a path changes during a certain snapshot, we discard the measurements collected during that snapshot and stop the current experiment, i.e., we apply our algorithms only on sequences of snapshots during which $\mathcal{P}$ remains unchanged.

ASSUMPTION 2. *Separability: A path is good if and only if all the links it traverses are good. A path is congested if and only if at least one of the links it traverses is congested.*

This assumption is common in Boolean-tomography algorithms [2, 3]. It is closely related to the problem of setting the link-congestion threshold t_l, defined in Section 2.1. We use $t_l = 0.01$, which has been shown to work well for mesh topologies and introduce negligible error [2].

ASSUMPTION 3. *Stationarity: The congestion behavior of any link during an experiment can be modeled as a stationary random process.*

This assumption is inherent in our congestion model (inherited from [3]), since we use the identically distributed random variables $X_{e_k}(n), n = 1..N$, to represent link e_k's congestion status during subsequent snapshots. We would like to clarify that this does not mean that the congestion status of link e_k remains the same during the experiment, only that the *probability* of link e_k being congested remains the same.

Suppose we can perform unicast end-to-end path measurements, i.e., measure the probability that any path or combination of paths is congested. We want to determine whether, given this information, it is feasible to identify the probability that a particular set of links is congested.

THEOREM 1. *For any network graph and any partition of links into correlation sets, if Assumptions 1, 2, and 3 hold, then the probability that any set of links is congested is identifiable from end-to-end measurements, if and only if Condition 1 is satisfied.*

CONDITION 1. *Any two correlation subsets $A, B \in \tilde{\mathcal{C}}$, $A \neq B$, are not traversed by exactly the same paths, i.e., $\psi(A) \neq \psi(B)$.*

Condition 1 generalizes a fundamental condition of network tomography, that any two *links* are not traversed by exactly the same paths [6]. Intuitively, this earlier condition captured the fact that, when two links participate in exactly the same paths, there is no way to differentiate between the two links based on end-to-end observations. We are generalizing this to say that, when two *groups of correlated links* participate in exactly the same paths (and assuming we know nothing about the nature of the correlation), there is no way to differentiate between the two groups based on end-to-end observations. Indeed, in the extreme case where each link in the network is uncorrelated with any other, our condition becomes exactly the earlier condition.

3. ALGORITHM

Our goal is to determine the probability that any set of links is congested, for all possible sets of links. Since links are partitioned in correlation sets such that any two links which belong to different correlation sets are independent, it is sufficient to compute the congestion probabilities $\mathbb{P}(\cap_{e_k \in A} X_{e_k} = 1)$, for all correlation subsets $A \in \tilde{\mathcal{C}}$. Theorem 1 states that if Condition 1 holds, then we can indeed identify from end-to-end measurements the congestion probability of any correlation subset $A \in \tilde{\mathcal{C}}$.

The Probability That Some Paths Are Good. Given a set of paths $L \subseteq \mathcal{P}$, we consider the probability that all paths in L are good, then necessarily the links in $\varepsilon(L)$, i.e., all links traversed by paths in L, must be good. We group the links in $\varepsilon(L)$ into correlation sets, and we get the following equation:

$$\mathbb{P}\left(\bigcap_{P_i \subseteq L} Y_{P_i} = 0\right) = \mathbb{P}\left(\bigcap_{e_k \in \varepsilon(L)} X_{e_k} = 0\right)$$

$$= \prod_{\substack{C_p \in \mathcal{C} \\ C_p \cap \varepsilon(L) \neq \emptyset}} \mathbb{P}\left(\bigcap_{e_k \in C_p \cap \varepsilon(L)} X_{e_k} = 0\right). \tag{3}$$

If we take the logarithm of Eq. 3, we obtain a linear equation:

$$\log \mathbb{P}\left(\bigcap_{P_i \subseteq L} Y_{P_i} = 0\right) =$$

$$\sum_{\substack{C_p \in \mathcal{C} \\ A = C_p \cap \varepsilon(L),\ A \neq \emptyset}} \log\ \mathbb{P}\left(\bigcap_{e_k \in A} X_{e_k} = 0\right). \tag{4}$$

Note that $A = C_p \cap \varepsilon(L)$, where $C_p \in \mathcal{C}$, represents a correlation subset, i.e., $A \in \tilde{\mathcal{C}}$. Thus, Eq. 4 expresses the relationship between the probability that a set of paths L is good, and the probabilities that the links in each correlation subset covered by the paths in L are good. The probability that any set of paths is good, can be determined from end-to-end measurements, for all possible sets of paths.

Challenges. If we can determine the probabilities $\mathbb{P}(\cap_{e_k \in A} X_{e_k} = 0)$, for all $A \in \tilde{\mathcal{C}}$, then we can easily compute the congestion probability of any correlation subset. The straightforward approach would be to consider Eq. 4 for all possible sets of paths. If Condition 1 holds, then we obtain a system of equations of full rank which we can solve to find out the probabilities $\mathbb{P}(\cap_{e_k \in A} X_{e_k} = 0)$, for all $A \in \tilde{\mathcal{C}}$. Nevertheless, there are two challenges with this approach: First, if there are too many correlation subsets traversed exclusively by congested paths, we end up with too many probabilities to compute. For example, for a correlation set of 40 links, we might end up with 2^{40} unknown probabilities, which is not feasible in practice. Second, it is not feasible to consider Eq. 4 for all possible sets of paths, unless there are only few paths in the network. For a network with n paths, we would need to consider 2^n equations, which can easily exceed our computational capacity.

We propose a scalable algorithm that addresses these two challenges. First, it only computes a feasible number of congestion probabilities, i.e., it determines the probabilities $\mathbb{P}(\cap_{e_k \in A} X_{e_k} = 1)$, for all $A \in \tilde{\mathcal{C}}$, for which $|A| \leq$ *bound*. The *bound* is determined such that we do not exceed the computational capacity. If *bound* $= 1$, we compute only the probability that any link is congested. If *bound* $= 2$, we compute the congestion probabilities of correlation subsets that contain either one or two links, i.e, $A \in \tilde{\mathcal{C}}$ with $|A| \leq 2$. Second, our algorithm selects a minimum number of equations in order to determine the congestion probabilities we are interested in. It does not consider an equation for each possible set of paths since many of these equations are redundant.

Algorithm Description. In the following, we give a brief description of our algorithm.

First, we determine the set of unknowns $\mathcal{U}$, i.e., the set of all correlation subsets $A \in \tilde{\mathcal{C}}$, for which we want to determine the probabilities $\mathbb{P}(\cap_{e_k \in A} X_{e_k} = 1)$. The set of unknowns $\mathcal{U}$ consists of all correlation subsets $A \in \tilde{\mathcal{C}}$, such that none of the links in A is traversed by a path which is good throughout the entire measurement period. For a correlation subset $A \in \tilde{\mathcal{C}}$, for which there exists a path $P_i \in \psi(A)$, such that $\mathbb{P}(Y_{P_i} = 0) = 1$, the congestion probability is 0. If the number of correlation subsets in $\mathcal{U}$ exceeds the computational capacity, we remove from $\mathcal{U}$ the correlation subsets A, for which $|A| >$ *bound*, where *bound* is determined such that it matches our computational capacity. Thus, the set of unknowns is defined as:

$$\mathcal{U} = \{\ A \in \tilde{\mathcal{C}} \ \mid \forall P_i \in \psi(A),\ \mathbb{P}(Y_{P_i} = 1) > 0 \text{ and } |A| \leq \textit{bound}\,\}. \tag{5}$$

Note that when we use a *bound* to limit the number of correlation subsets for which we want to compute the congestion probabilities, we can no longer consider an equation for any possible set of paths. This is because some sets of paths cover correlation subsets that are not in $\mathcal{U}$. We say that "a set of paths L generates a valid equation for a correlation subset $A \in \mathcal{U}$", if L covers A, and L does not cover any other correlation subset

$B \notin \mathcal{U}$.

If Condition 1 holds for all correlation subsets $A \in \mathcal{U}$, then we can compute the congestion probability $\mathbb{P}(\cap_{e_k \in A} X_{e_k} = 1)$, for all $A \in \mathcal{U}$.

Initialization. In this step, we determine for each $A \in \mathcal{U}$, a set of paths $\mathcal{P}_A$ that generates a valid equation for A.

Since $A \in \tilde{\mathcal{C}}$, there must be a correlation set C_p, such that $A \subseteq C_p$. The set of paths $\mathcal{P}_A$ needs to satisfy three conditions: (i) any link in A must be traversed by at least one path in $\mathcal{P}_A$, (ii) the paths in $\mathcal{P}_A$ must not traverse any link in $C_p \setminus A$, and (iii) the paths in $\mathcal{P}_A$ must not cover any correlation subset $B \notin \mathcal{U}$.

For each link $e_k \in A$, we define the set of paths $\psi^*(e_k)$ that contains all paths which traverse e_k, but do not traverse any link in $C_p \setminus A$, i.e.,

$$\psi^*(e_k) = \psi(e_k) - \psi(C_p \setminus A).$$

We choose one path from $\psi^*(e_k)$ for each $e_k \in A$, and obtain thus a set of paths. We consider all such possible combinations of paths, until we find a set of paths $\mathcal{P}_A$ that satisfies condition (iii). If such a set of paths does not exist, then we are not able to determine the congestion probability of A; hence, we remove A from the set of unknowns $\mathcal{U}$. This part of the algorithm is described in Algorithm 1.

For each $A \in \mathcal{U}$, we write Eq. 4 for the set of paths $\mathcal{P}_A$. We form thus a system of linear equations where the unknowns are $\mathbb{P}(\cap_{e_k \in A} X_{e_k} = 0)$. Let $\mathbf{R}$ be the matrix associated to our system of equations, a binary matrix of dimensions $m \times n$, where $n = |\mathcal{U}|$. Each column in $\mathbf{R}$ corresponds to a correlation subset $A \in \mathcal{U}$, while each row in $\mathbf{R}$ corresponds to some set of paths $\mathcal{P}_A$, with $A \in \mathcal{U}$. The matrix has only m rows, $m \leq n$, since our algorithm can choose the same set of paths for two different correlation subsets, i.e., $\mathcal{P}_A = \mathcal{P}_B$ for $A, B \in \mathcal{U}$, $A \neq B$.

Usually, matrix $\mathbf{R}$ is not full rank. Therefore, we need to consider more equations in order to be able to solve exactly the system of equations.

Iteration. In this step, we try to find additional linearly independent equations in order to obtain a full rank matrix for our system of equations. More precisely, we need an algorithm to pick sets of paths that are most likely to generate linearly independent equations, and an efficient mechanism to verify that an equation is indeed linearly independent.

We use the fundamental theorem of linear algebra, which states that for any matrix $\mathbf{R}$,

$$null(\mathbf{R}) = (im(\mathbf{R}^T))^{\perp},$$

i.e., the null space of a matrix is orthogonal to the row space. Thus, for any row $\mathbf{r}$ to be linearly independent with the rows in $\mathbf{R}$, it is necessary and sufficient that $\mathbf{r}$ is non-orthogonal to the null space of matrix $\mathbf{R}$. We compute the null space $\mathbf{N}$ of matrix $\mathbf{R}$:

$$\mathbf{N} = \{\ \mathbf{v} \in \mathbb{R}^n \mid \mathbf{R}\mathbf{v} = \mathbf{0}\ \}.$$

This can be done in practice using singular value decomposition or QR factorization. In order to test if a row $\mathbf{r}$ is linearly independent with the rows in $\mathbf{R}$, we need to verify the condition:

$$||\mathbf{r}\ \mathbf{N}|| > 0.$$

Once we find a linearly independent row, we can extend matrix $\mathbf{R}$ with the new row:

$$\mathbf{R'} = \begin{vmatrix} \mathbf{R} \\ \mathbf{r} \end{vmatrix}.$$

However, each time we extend $\mathbf{R}$ with a new row, the null space $\mathbf{N}$ changes; $\mathbf{R'}$ has null space $\mathbf{N'} \neq \mathbf{N}$. If we want to iteratively extend matrix $\mathbf{R}$, we need to compute each time the new null space. Yet computing every time the null space from scratch using singular value decomposition or QR factorization incurs a high cost. Thus, we compute the null space $\mathbf{N'}$ based on the previous null space $\mathbf{N}$ and the new added row $\mathbf{r}$. The formula to update the null space is(Lemma 3):

$$\mathbf{N'} = \left(\mathbf{I}_n - \frac{\mathbf{N}_{*1}\mathbf{r}}{\mathbf{r}\mathbf{N}_{*1}}\right)\mathbf{N}_{*2:*} \tag{6}$$

where $\mathbf{N}$ is the null space of matrix $\mathbf{R}$, $\mathbf{I}_n$ is the identity matrix of dimension n, and $\mathbf{r}$ is the new row added to matrix $\mathbf{R}$. We use the notation $\mathbf{N}_{*1}$ to denote the first column of matrix $\mathbf{N}$, and the notation $\mathbf{N}_{*2:*}$ to denote the columns of $\mathbf{N}$ from column 2 onward.

We have seen how to iteratively add linearly independent rows to matrix $\mathbf{R}$. We now need a way to determine the sets of paths that are most likely to generate linearly independent rows since it is not feasible to consider all possible sets of paths. The null space $\mathbf{N}$ is formed by vectors $\mathbf{v} \in \mathbb{R}^n$ that are orthogonal to matrix $\mathbf{R}$, i.e.,

$$\mathbf{R}\mathbf{v} = \mathbf{0}.$$

Since each column in $\mathbf{R}$ corresponds to a correlation subset $A \in \mathcal{U}$, it follows that each row in $\mathbf{N}$ can be associated with a correlation subset $A \in \mathcal{U}$. We sort the rows in $\mathbf{N}$ by the number of non-zero elements in each row. We pick the row that has the most non-zero elements, we find its associated correlation subset $A \in \mathcal{U}$, and we try to obtain a valid equation for A by considering sets of paths that cover A. The intuition behind is that, by considering a set of paths that cover A, the row $\mathbf{r}$ corresponding to this set of paths affects more vectors in the null space, increasing the chances that $||\mathbf{r}\mathbf{N}|| > 0$. If $\mathbf{r}$ is linearly independent with the other rows in $\mathbf{R}$, we extend $\mathbf{R}$ by $\mathbf{r}$, otherwise we consider the next row in $\mathbf{N}$ that has the most non-zero elements and we reiterate.

In order to find a set of paths that is likely to generate a linearly independent equation for A, we consider the paths in $\psi^*(A)$, i.e., all paths that traverse links in A, but do not traverse any link in $C_p \setminus A$, where C_p is the correlation set of A:

$$\psi^*(A) = \psi(A) - \psi(C_p \setminus A).$$

If the set of paths $\psi^*(A)$ does not yield a valid linearly independent equation, we iteratively consider each correlation subset S covered by the paths in $\psi^*(A)$. For each correlation subset S covered by the paths in $\psi^*(A)$, we test wheather the set of paths $\psi^*(A) - \psi(S)$ generates a linearly independent equation. If this is not the case, we consider all links $e_k \in S$, and test weather the set of paths $\psi^*(A) - \psi(e_k)$ generates a linearly independent equation. This part of the algorithm is described in Algorithm 2.

This last step that tries to find a valid linearly independent equation for a correlation subset is a heuristic, but when the Condition 1 is satisfied, we are always able to obtain a full rank matrix.

Finally, once we have obtained a full rank matrix, we use norm minimization to solve the system of equations and determine the probabilities $\mathbb{P}(\cap_{e_k \in A} X_{e_k} = 0)$, for all $A \in \tilde{\mathcal{C}}$.

A complete description of the algorithm is given in Algorithm 3.

4. BOOLEAN INFERENCE

In this section, we discuss how to infer which links in the network are congested at any moment in time from end-to-end measurements. More precisely, given the states of paths at any moment in time, and the probability that any set of links is congested, for all possible sets of links, we want to determine the state of each link at that particular moment of time.

The input parameters to our problem are:

- the congestion probabilities for all correlation subsets: $\Gamma = \{ \mathbb{P}(\bigcap_{e_k \in A} X_{e_k} = 1) \}_{A \in \tilde{\mathcal{C}}}$. These are sufficient to determine the probability that any set of links is congested.
- the states of the paths at a given moment in time $\mathbf{y} = \{ y_{P_i} \}_{P_i \in \mathcal{P}}$.
- the network topology that describes which links are traversed by each path

Our goal is to determine the most likely link states $\mathbf{x} = \{x_{e_j}\}_{e_j \in \mathcal{E}}$ that can explain the path states $\mathbf{y}$.

4.1 Bayesian Inference

In Bayesian Inference, our problem can be stated as

$$\arg\max_{\mathbf{x}} \mathbb{P}_\Gamma(\ \mathbf{X} = \mathbf{x} \mid \mathbf{Y} = \mathbf{y}\ ), \tag{7}$$

where $\mathbb{P}_\Gamma(.)$ is the probability given the prior Γ. By Bayes' rule, Eq. 7 becomes:

$$\arg\max_{\mathbf{x}} \mathbb{P}_\Gamma(\mathbf{X} = \mathbf{x} | \mathbf{Y} = \mathbf{y}) = \arg\max_{\mathbf{x}} \frac{\mathbb{P}_\Gamma(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) \mathbb{P}_\Gamma(\mathbf{X} = \mathbf{x})}{\mathbb{P}_\Gamma(\mathbf{Y} = \mathbf{y})}.$$

Since $\mathbb{P}_\Gamma(\mathbf{Y} = \mathbf{y})$ does not depend on $\mathbf{x}$, we are left to solve the maximization problem:

$$\arg\max_{\mathbf{x}} \mathbb{P}_\Gamma(\mathbf{Y} = \mathbf{y} | \mathbf{X} = \mathbf{x}) \mathbb{P}_\Gamma(\mathbf{X} = \mathbf{x}). \tag{8}$$

The states of any two paths are conditionally independent given the states of all links. Hence, we can write:

$$\mathbb{P}_\Gamma(\mathbf{Y} = \mathbf{y} \mid \mathbf{X} = \mathbf{x}) = \prod_{P_i \in \mathcal{P}} \mathbb{P}_\Gamma(Y_{P_i} = y_{P_i} | \mathbf{X} = \mathbf{x}).$$

We consider the probability $\mathbb{P}_\Gamma(Y_{P_i} = y_{P_i} | \mathbf{X} = \mathbf{x})$ for any path $P_i \in \mathcal{P}$. From Assumption 2, it follows that if $y_{P_i} = 0$, then $\mathbb{P}_\Gamma(Y_{P_i} = 0 | \mathbf{X} = \mathbf{x}) = 1$ if and only if all links traversed by path P_i are good; otherwise $\mathbb{P}_\Gamma(Y_{P_i} = 0 | \mathbf{X} = \mathbf{x}) = 0$. Similarly, if $y_{P_i} = 1$, then $\mathbb{P}_\Gamma(Y_{P_i} = 1 | \mathbf{X} = \mathbf{x}) = 1$ if and only if path P_i traverses at least one congested link; otherwise $\mathbb{P}_\Gamma(Y_{P_i} = 1 | \mathbf{X} = \mathbf{x}) = 0$. Therefore, in order for the probability $\mathbb{P}_\Gamma(\mathbf{Y} = \mathbf{y} \mid \mathbf{X} = \mathbf{x})$ to have the maximum value, all links on good paths must be good, and each congested path must traverse at least one congested link. Let $\mathcal{P}_c$ be the set of congested paths at that particular moment in time:

$$\mathcal{P}_c = \{P_i \in \mathcal{P} \mid y_{P_i} = 1\}.$$

Since all paths in $\mathcal{P} \setminus \mathcal{P}_c$ are good, then all links covered by these paths, i.e., $\varepsilon(\mathcal{P} \setminus \mathcal{P}_c)$, must be good. We denote by $\mathcal{E}^*$ the set of links that are not traversed by any good path, which we call the candidate links :

$$\mathcal{E}^* = \mathcal{E} \setminus \varepsilon(\mathcal{P} \setminus \mathcal{P}_c) \tag{9}$$

We must find a set of congested links $\chi \subseteq \mathcal{E}^*$, such that each path in $\mathcal{P}_c$ traverses at least one link in χ. Therefore, in order to find the solution of Eq. 8, we need to solve the aforementioned set cover problem, and at the same time the maximization problem:

$$\arg\max_{\chi \subseteq \mathcal{E}^*} \mathbb{P}_\Gamma \left((\bigcap_{e_j \in \chi} X_{e_j} = 1) \bigcap (\bigcap_{e_j \in \mathcal{E} \setminus \chi} X_{e_j} = 0) \right)$$

We define the state of a correlation set as the set of all congested links in that correlation set. For example, the state S^p of correlation set C_p consists of all congested links in C_p, i.e., $S^p = \{e_j \in C_p | X_{e_j} = 1\}$. We group links in correlation sets, and we obtain:

$$\arg\max_{\chi \subseteq \mathcal{E}^*} \prod_{C_p \in \mathcal{C}} \mathbb{P}_\Gamma(\ S^p = C_p \cap \chi\ ). \tag{10}$$

Given the set of congested links χ, we define the weight of correlation set C_p as:

$$\omega_{C_p}(\chi) = \mathbb{P}_\Gamma(\ S^p = C_p \cap \chi\ ).$$

Consequently, Eq. 10 becomes:

$$\arg\max_{\chi \subseteq \mathcal{E}^*} \prod_{C_p \in \mathcal{C}} \omega_{C_p}(\chi)$$

Finally, if we take the logarithm of the above equation, we get:

$$\arg\max_{\chi \subseteq \mathcal{E}^*} \sum_{C_p \in \mathcal{C}} \log \omega_{C_p}(\chi). \qquad (11)$$

In conclusion, we need to find the set of congested links $\chi \in \mathcal{E}^*$ such that each path in $\mathcal{P}_c$ traverses at least one link in χ, and at the same time solve Eq. 11. Unfortunately, the above optimization problem is NP-complete, therefore, we propose a greedy algorithm to find a feasible solution.

4.2 Bayesian Inference with Complete Information

We now present an algorithm that uses the congestion probabilities of all correlation subsets, in order to determine the congested links in the network at any moment in time. The algorithm takes as input the set of congested paths $\mathcal{P}_c$ at a specified moment in time, and the congestion probabilities of all correlation subsets, i.e., $\forall A \in \tilde{\mathcal{C}}$, $\mathbb{P}(\cap_{e_j \in A} X_{e_j} = 1)$. The goal is to infer the most likely set of congested links χ that have caused the paths in $\mathcal{P}_c$ to be congested.

Initialy, we determine the set of candidate links $\mathcal{E}^*$ from Eq. 9, and we compute the state probabilities $\mathbb{P}(S^p = A)$, for all $A \subseteq C_p$, and all correlation sets $C_p \in \mathcal{C}$, as described in Lemma 4.

In the first phase of the algorithm, we assign to each correlation set C_p the most probable state S_{C_p} given the measurements, i.e.,

$$S_{C_p} = \arg\max_{A \subseteq C_p, A \subseteq \mathcal{E}^*} \mathbb{P}(S^p = A).$$

Next, we compute the set of congested paths that are not explained by the chosen states of the correlation sets:

$$\mathcal{L} = \mathcal{P}_c \setminus \bigcup_{C_p \in C} \psi\left(S_{C_p}\right).$$

If $\mathcal{L} = \emptyset$, then the algorithm outputs $\chi = \bigcup_{C_p \in \mathcal{C}} S_{C_p}$ and exits. Otherwise, the algorithm goes to the second phase.

In the second phase of the algorithm, we consider all correlation subsets $A \subseteq C_p$, $\forall C_p \in \mathcal{C}$, such that $S_{C_p} \subset A$. For each such correlation subset A, we compute its score using the formula:

$$\text{score}_{A \subseteq C_p} = \frac{\log \frac{1}{W_{C_p}(A)}}{|\mathcal{L} \cap \psi\,(A)\,|},$$

and we select the correlation subset $A \subseteq C_p$ that has the lowest score. Intuitively, this is the correlation subset A which achieves the highest state probability $\mathbb{P}(S^p = A)$ and that covers the most remaining congested paths $|\mathcal{L} \cap \psi\,(A)\,|$. We change the state of correlation set C_p to $S_{C_p} = A$ and update the set of remaining congested paths: $\mathcal{L} = \mathcal{L} \setminus \psi\,(A)$. If $\mathcal{L} \neq \emptyset$, we reiterate from the second part of the algorithm.

The first part of the algorithm which assigns to each correlation set the most probable state, solves the optimization problem given by Eq. 11. If all congested paths can be explained by the chosen states of the correlation sets, then the algorithm finds the exact solution for the optimization problem described Section 4.1. Otherwise, we select more links to congest until all congested paths are explained. A detailed description of this algorithm is given by Algorithm 4.

4.3 Bayesian Inference with Incomplete Information

In some situations, we are not able to determine the congestion probability of all correlation subset. For example, when each correlation subset is not traversed by a different set of paths and/or when the network topology has a large correlation set. In these cases, we need to deal with incomplete information, i.e., we know the congestion probabilities of only some of the correlation subsets. Since we are not able to determine the state probabilities for each correlation set, we cannot apply the first phase of the inference algorithm described above. Therefore, we only apply the second phase of the algorithm that has complete inromation. We consider all correlation subsets for which we were able to determine the probability $\mathbb{P}(\cap_{e_j \in A} X_{e_j} = 1)$, and for each of these correlation subset we compute its score as follows:

$$\text{score}_{A \subseteq C_p} = \frac{\log \frac{1}{\mathbb{P}(\cap_{e_j \in A} X_{e_j} = 1)}}{|\mathcal{L} \cap \psi\,(A)\,|}.$$

We iteratively pick the correlation subset that yields the lowest score until all congested paths are explained. A detailed description of this algorithm is given by Algorithm 5.

5. INFERENCE ANALYSIS

In this section, we analyze in depth several state-of-the-art inference algorithms.

First, we ask the question: What is the condition that needs to be fulfilled by a network such that, regardeless of which links in the network are congested, we are always able to determine precisely the state of links from end-to-end measurements? The necessary and suficient condition that needs to be fulfilled is that any subset of links in the network must be traversed by a different set of paths, e.g.,

$$\forall A, B \subseteq \mathcal{E}, \psi\left(A\right) \neq \psi\left(B\right). \qquad (12)$$

Unfortunately, the only network that satisfies Condition 12 is the trivial network, where end-hosts are directly interconnected (Lemma 1). Therefore, all algorithms that try to solve the bayesian inference problem need to make some assumptions. Condition 12 is similar with Condition 1, the necessary and suficient condition needed in order to identify the probability that any set of links in the network is congested. The difference is that Condition 12 needs to hold in a network where the set of hosts and the set of routers are disjoint, while Condition 1 needs to hold within a correlation set where all nodes are routers. In order for the Condition 1 to hold in a correlation set, each node in the correlation set must be a border router, e.g, it needs to be traversed by paths that entry or exit the correlation set via this node.

We analyse four state-of-the-art inference algorithms. The first one is the algorithm proposed by Duffield [2], which works on tree topologies. The performance of this algorithm depends highly on which links in the network are congested. In fact, there exists no topology where regardless of the congested links, this algorithm always performs well (excluding the scenario where there is only one congested link in the network). For example, the algorithm misses to detect any congested link that is located in the subtree rooted at another congested link, or when all child links of a node are congested, the algorithm blames the parent link and marks the child links are good. The performance of this algorithm is improved when: there are only a few congested links in the network, and these links have low probability of being congested. In general, this algorithm will always have a low false positives rate since it finds te smallest set of links that can explain the congestion. We call this algorithm, the Duffield algorithm.

The second algorithm is a generalization of the first algorithm for mesh topologies, its description is given in [1]. It inherits the shortcomings of its predecesor: there exists no topology where regardless of the congested links, this algorithm always performs well (Lemma 2). However, this algorithm doesn't always have a low false positives rate since when there are more links traversed by the same number of congested paths, it will mark all of them as congested. We call this algorithm, the Sparsity algorithm.

The third algorithm we analyze is the algorithm described in [3]. This algorithm has well defined conditions under which it performs well: (i) the links in the network are independent (ii) each link in the network

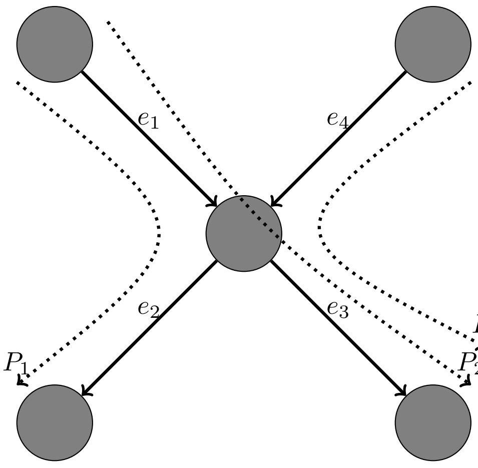

Figure 1: A toy topology.
Links $\mathcal{E}^* = \{e_1, e_2, e_3, e_4\}$. Paths $\mathcal{P}^* = \{P_1, P_2, P_3\}$.

is traversed by a different set of paths (iii) the congestion probabilities are below 0.5 and (iv) the congestion probabilities of network links remains constant throughout the measurement. When one or more of the above condition is not met, the algorithm can have a bad performance. We call this algorithm, the Independence algorithm.

The final algorithm we analyse is our algorithm presented in Section 4. Our algorithm performs well under the following conditions: (i) the correlation set are correctly known (ii) Condition 1 holds and (iv) the congestion probabilities of network links remains constant throughout the measurement. We call our algorithm, the Correlation algorithm.

In the following, we consider a toy example (Figure 1) where depending on which links are congested and which links are correlated, each of the four algorithms can have a bad performance.

First, we consider the case when both links e_2 and e_3 are congested. It follows that all three paths in the network P_1,P_2 and P_3 are congested. Since the Sparsity algorithm chooses the links that are traversed by more congested paths, the congested links inferred by the Sparsity will be e_1 and e_2, which leads to a detection rate of 0.5 and a false positive rate of 0.5. Note that if the conditions for the Independence algorithm and for the Correlation algorithm hold in this case, then both of this algorithms will have a detection rate of 1.0, and a false positive rate of 0.0.

Second, we consider the case when links e_2 and e_3 are congested and perfectly correlated, that is, they are both congested or both good. Since the Independence algorithm assumes all links are independent, it will not take into account the correlation of links e_2 and e_3. As a consequence, it will compute badly the congestion probabilities of links. In particular, all links in the network will have the same congestion probability. The algorithm will infer that links e_1 and e_3 are congested since these are the links traversed by more congested paths. This gives us a detection rate of 0.5 and a false positive rate of 0.5. In this scenario, the Sparsity algorithm will have on average the same detection rate/false positive rate as the Independence algorithm. However, the Correlation algorithm will achieve perfect accuracy.

Finally, we consider the case when links e_2 and e_3 are congested and perfectly correlated, and links e_1 and e_4 are correlated, but not congested. In this case, Condition 1 doesn't hold for the two correlation sets $\{e_2, e_3\}$ and $\{e_1, e_4\}$. Consequently, we don't have enough equations to compute accurately the congestion probabilities of links. The probability that links $\{e_2, e_3\}$ are congested is the same with the probability that links $\{e_1, e_4\}$ are congested. Therefore, the algorithm will choose at random one of them, and on average, the detection rate is 0.5 and the false positives rate is 0.5. In this scenario, both the Sparsity algorithm and the Independence algorithm will perform on average worse or equal to the Correlation algorithm.

In conclusion, we have shown that the boolean inference problem is an ill-posed problem, and that no inference algorithm can produce accurate results, in all possible realistic congestion scenarios.

6. 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).

7. REFERENCES

[1] 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.

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

[3] 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.

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

[5] 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.

[6] Y. Vardi. Network Tomography: Estimating Source-Destination Traffic Intensities. *Journal of the American Statistical Association*, 91:365–377, 1996.

APPENDIX

A. BOOLEAN INFERENCE, AN ILL-POSED PROBLEM

Boolean Inference is an ill-posed problem. There exists no network (apart from the trivial network where end-hosts are directly interconnected), where regardeless of which links are congested, we are always able to determine precisely the state of links from end-to-end measurements. Condition 12 states the necessary and suficient condition that needs to be fulfilled in order to be able to determine precisely the state of links from end-to-end measurements, i.e., any subset of links in the network must be traversed by a different set of paths. Unfortunately, the only network that satisfies Condition 12 is the trivial network(Lemma 1).

LEMMA 1. *Condition 12 holds only in a trivial network, where end-hosts are directly interconnected.*

PROOF. Given any non-trivial network, there exists at least one node that is a router. We denote by $\mathcal{E}_{in}$ the set of ingress links of the router, and by $\mathcal{E}_{out}$, the set of egress links of the router. From our network model, the two sets of links are disjoint $\mathcal{E}_{in} \cap \mathcal{E}_{out} = \emptyset$. Since all path that enter the router, must also exit the router, it follows that $\psi(\mathcal{E}_{in}) = \psi(\mathcal{E}_{out})$. Thus Condition 12 is not fulfilled since there exist two subsets of links $\mathcal{E}_{in}$ and $\mathcal{E}_{out}$ traversed by the same paths. $\square$

B. THE SPARSITY ALGORITHM

LEMMA 2. *Given any non-trivial topology, the Sparsity algorithm may fail to identify the congested links.*

PROOF. In any non-trivial topology, there exists two subsets of links A and B such that $\psi(A) = \psi(B)$(Lemma 1). Consider the scenario when $\mathbb{P}(\cap_{e_j \in A} X_{e_j} = 1) > 0$, and $\mathbb{P}(\cap_{e_j \in B} X_{e_j} = 1) > 0$, that is, the links in A, and respectively B have a positive probability of being congested. All other links in the network are good $\mathbb{P}(\cap_{e_j \in C} X_{e_j} = 1) = 0$, for all $C \subseteq \mathcal{E}$. We have thus, three possible scenarios: the links in A are congested, while all other links in the network are good, the links in B are congested, while all other links in the network are good, and the links in $A \cup B$ are congested, while all other links in the network are good. Each of these scenarios produces the same outcome visible from end-to-end measurements, namely, the paths in $\psi(A)$ are congested, while all other paths in the network are good. The Sparsity algorithm systematically picks the same solution, missing the two other possible scenarios. □

C. UPDATING THE NULL SPACE BASIS

LEMMA 3. *A matrix $\boldsymbol{R}$ of dimension $m \times n$ and rank k, is expanded with a row $\boldsymbol{r}$ that increases the rank of the matrix,*

$$\boldsymbol{R'} = \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{r} \end{bmatrix},$$

where rank$(\boldsymbol{R'}) = k+1$. Then, a basis of the null space of matrix $\boldsymbol{R'}$ can be computed from:

$$\boldsymbol{N'} = \left(\boldsymbol{I}_n - \frac{\boldsymbol{N}_{*1} \times \boldsymbol{r}}{\boldsymbol{r} \times \boldsymbol{N}_{*1}} \right) \times \boldsymbol{N}_{*2:(n-k)},$$

where $\boldsymbol{I}_n$ is the identity matrix of dimension n, $\boldsymbol{N}$ is a basis of the null space of matrix $\boldsymbol{R}$, $\boldsymbol{N}_{*1}$ is the first column of matrix $\boldsymbol{N}$, and $\boldsymbol{N}_{*2:(n-k)}$ is the matrix formed by columns 2 to $(n-k)$ of matrix $\boldsymbol{N}$.

PROOF. The null space of matrix $\mathbf{R}$ has dimension $n-k$, while the null space of matrix $\mathbf{R'}$ has dimension $n-k-1$. A basis $\mathbf{N'}$ of the null space of $\mathbf{R'}$ must satisfy two conditions: (i) each column in $\mathbf{N'}$ must be orthogonal with each row in $\mathbf{R'}$ (the null space of a matrix is the orthogonal complement of its row space), and (ii) the rank of $\mathbf{N'}$ must be $n-k-1$.

The null space of matrix $\mathbf{R'}$ is a subspace of the null space of $\mathbf{R}$. We denote by $\mathbf{N}$ a basis of the null space of $\mathbf{R}$. For any basis $\mathbf{N'}$ of the null space of $\mathbf{R'}$, we can write each column of $\mathbf{N'}$ as a linear combination of the columns of $\mathbf{N}$:

$$\mathbf{N'} = \mathbf{N} \times \mathbf{T}, \tag{13}$$

where $\mathbf{T}$ is transformation matrix of dimension $(n-k) \times (n-k-1)$. Eq. 13 ensures that each column in $\mathbf{N'}$ is orthogonal with each row in $\mathbf{R}$, but for condition (i) to hold, we also require that:

$$\mathbf{r} \times \mathbf{N'} = \mathbf{0}. \tag{14}$$

We combine Eq. 13 and Eq. 14, and we obtain:

$$\mathbf{r} \times \mathbf{N} \times \mathbf{T} = \mathbf{0}.$$

We use the notation $\mathbf{q} = \mathbf{r} \times \mathbf{N}$, and the above equation becomes:

$$\mathbf{q} \times \mathbf{T} = \mathbf{0}. \tag{15}$$

Since we assume we know a basis $\mathbf{N}$ of the null space of matrix $\mathbf{R}$, and row $\mathbf{r}$ is also known, we cand determine $\mathbf{q}$. In fact, Eq. 15 is an undetermined system of linear equations with $n-k$ equations, and $(n-k) \times (n-k-1)$ unknowns, i.e., the entries in the transformation matrix $\mathbf{T}$. We consider a transformation matrix $\mathbf{T}$ of the following form:

$$\mathbf{T} = \begin{bmatrix} \alpha \\ \mathbf{I}_{n-k-1} \end{bmatrix}, \tag{16}$$

where $\mathbf{I}_{n-k-1}$ is the identity matrix of dimension $n-k-1$, and α a row of dimension $n-k-1$. We choose this form for the transformation matrix $\mathbf{T}$ to ensure that the rank of $\mathbf{T}$ is $n-k-1$, and consequently, that $\mathbf{N'}$ has full column rank, satisfying thus condition (ii). We use the transformation matrix defined in Eq. 16, to solve the system of linear equations in Eq. 15. After some algebric manipulations, we obtain:

$$\alpha = -\frac{\mathbf{r} \times \mathbf{N}_{*2:(n-k)}}{\mathbf{r} \times \mathbf{N}_{*1}}, \tag{17}$$

where $\mathbf{N}_{*1}$ is the first column of matrix $\mathbf{N}$, and $\mathbf{N}_{*2:(n-k)}$ is the matrix formed by columns 2 to $(n-k)$ of matrix $\mathbf{N}$. In conclusion, we have determined a transformation matrix $\mathbf{T}$, which we can use in Eq 13 to discover a basis $\mathbf{N'}$ of the null space of matrix $\mathbf{R'}$:

$$\mathbf{N'} = \left(\mathbf{I}_n - \frac{\mathbf{N}_{*1} \times \mathbf{r}}{\mathbf{r} \times \mathbf{N}_{*1}} \right) \mathbf{N}_{*2:(n-k)},$$

where $\mathbf{I}_n$ is the identity matrix of dimension n, $\mathbf{N}$ is a basis of the null space of matrix $\mathbf{R}$, $\mathbf{N}_{*1}$ is the first column of matrix $\mathbf{N}$, and $\mathbf{N}_{*2:(n-k)}$ is the matrix formed by columns 2 to $(n-k)$ of matrix $\mathbf{N}$. □

D. COMPUTING THE STATE PROBABILITIES OF A CORRELATION SET

LEMMA 4. *Given a correlation set C_p, if the congestion probabilities $\mathbb{P}(\cap_{e_j \in A} X_{e_j} = 1)$ are known for all $A \subseteq C_p, A \neq \emptyset$, then the state probabilities of correlation set C_p, i.e., $\mathbb{P}(S^p = B), \forall B \subseteq C_p$, are identifiable.*

PROOF. We must show that we can determine the probabilities $\mathbb{P}(S^p = B)$ for all $B \subseteq C_p$. First, we define a partial ordering $\mathcal{T}$ over all subsets $B \subseteq C_p$, where the subsets are sorted in decreasing number of their links $|B|$. We will prove by induction on the partial ordering $\mathcal{T}$ that we can indeed compute all these probabilities.

First Step: Consider the first subset B in the partial ordering $\mathcal{T}$. In this case, we have $B = C_p$ and $\mathbb{P}(S^p = C_p) = \mathbb{P}(\cap_{e_j \in C_p} X_{e_j} = 1)$, which is already known from the hypothesis.

Induction Step: Next, we assume that the probability $\mathbb{P}(S^p = D)$ is known for all $|D| > |B|, D \subseteq C_p$. We want to determine $\mathbb{P}(S^p = B)$. We know that:

$$\mathbb{P}(\cap_{e_j \in B} X_{e_j} = 1) = \sum_{B \subseteq D} \mathbb{P}(S^p = D)$$

Hence, it follows that:

$$\mathbb{P}(S^p = B) = \mathbb{P}(\cap_{e_j \in B} X_{e_j} = 1) - \sum_{B \subset D} \mathbb{P}(S^p = D).$$

Since the probabilities $\mathbb{P}(S^p = D)$ are known for all $|D| > |B|$ (from the induction hypothesis), we can compute $\mathbb{P}(S^p = B)$ from the above equation. $\square$

E. PSEUDOCODE

Algorithm 1 Finding a set of paths $\mathcal{P}_A$ that generates a valid equation for $A \in \mathcal{U}$.

```
Require: $A = \{e_1, e_2, ..., e_{|A|}\}$, $A \subseteq C_p$
Ensure: $\mathcal{P}_A$ generates a valid equation for $A$

  for $e_k \in A$ do
    $\psi^*(e_k) = \psi(e_k) - \psi(C_p \setminus A)$
    if $\psi^*(e_k) = \emptyset$ then
      return $\emptyset$
    end if
  end for

  for $P_1 \in \psi^*(e_1)$ do
    for $P_2 \in \psi^*(e_2)$ do
$\vdots$
      for $P_{|A|} \in \psi^*(e_{|A|})$ do
        $\mathcal{P}_A = \{P_1, P_2, ..., P_{|A|}\}$
        if $\{B \in \tilde{\mathcal{C}} \mid \mathcal{P}_A \textit{ covers } B\} \subseteq \mathcal{U}$ then
          return $\mathcal{P}_A$
        end if
      end for
      $\vdots$
    end for
  end for

  return $\emptyset$
```

Algorithm 2 Finding a valid linearly independent equation for $A \in \mathcal{U}$.

```
Require: $A \in \mathcal{U}$, $A \subseteq C_p$
Ensure: a valid linearly independent equation for $A$

  $\psi^*(A) = \psi(A) - \psi(C_p \setminus A)$
  if $\psi^*(A) = \emptyset$ then
    return $\emptyset$
  end if

  if isLIEq($\psi^*(A)$)=1 then
    return $\psi^*(A)$
  end if

  for $S \in \{B \in \tilde{\mathcal{C}} \mid \psi^*(A) \textit{ covers } B\}$ do
    if isLIEq( $\psi^*(A) \setminus \psi(S)$ )=1 then
      return $\psi^*(A) \setminus \psi(S)$
    end if
    for $e_k \in S$ do
      if isLIEq( $\psi^*(A) \setminus \psi(e_k)$ )=1 then
        return $\psi^*(A) \setminus \psi(e_k)$
      end if
    end for
  end for

  return $\emptyset$


  isLIEq( $\mathcal{L} \subseteq \mathcal{P}$ ):

  if $\{B \in \tilde{\mathcal{C}} \mid \mathcal{L} \textit{ covers } B\} \not\subseteq \mathcal{U}$ then
    return 0
  end if

  $\mathbf{r} \leftarrow$ row vector generated by $\mathcal{L}$

  if $||\mathbf{r}\,\mathbf{N}|| > 0$ then
    return 1
  end if


  return 0
```

Algorithm 3 The Algorithm that computes the congestion probabilities

```
compute the set of unknowns $\mathcal{U}$ from Eq. 5
$\mathbf{R} = []$

for $A \in \mathcal{U}$ do
  $\mathcal{P}_A$ = Algorithm_1( $A$)
  $\mathbf{r} \leftarrow$ row vector generated by $\mathcal{P}_A$
  $\mathbf{R} = \begin{vmatrix} \mathbf{R} \\ \mathbf{r} \end{vmatrix}$
end for

if $\mathbf{R}$ is full rank then
  return $\mathbf{R}$
end if

$\mathbf{N} = null(\mathbf{R})$

while True do
  sort the rows in $\mathbf{N}$ by the number of non-zero elements, and put the correlation subset associated to each row in $\mathcal{T}$

  for $A \in \mathcal{T}$ do
    $\mathcal{L}$ =Algorithm_2( $A$)

    if $\mathcal{L} \neq \emptyset$ then
      $\mathbf{r} \leftarrow$ row vector generated by $\mathcal{L}$
      $\mathbf{R} = \begin{vmatrix} \mathbf{R} \\ \mathbf{r} \end{vmatrix}$
      update $\mathbf{N}$ using Eq. 6
      break
    end if

    if $\mathbf{R}$ is full rank then
      return $\mathbf{R}$
    end if

  end for
end while
```

Algorithm 4 Inference Algorithm with Complete Information

```
$\mathcal{E}^* = \mathcal{E} \setminus \varepsilon(\mathcal{P} \setminus \mathcal{P}_c)$
compute $\mathbb{P}(S^p = A)$, for all $A \subseteq C_p$, and all $C_p \in \mathcal{C}$
for $C_p \in \mathcal{C}$ do
  $S_{C_p} = \underset{A \subseteq C_p, A \subseteq \mathcal{E}^*}{\arg\max} \; \mathbb{P}(S^p = A)$
end for
$\mathcal{L} = \mathcal{P}_c \setminus \bigcup_{C_p \in C} \; \psi\left(S_{C_p}\right)$
while $\mathcal{L} \neq \emptyset$ do
  for $C_p \in \mathcal{C}$ do
    for $A \subseteq C_p$ do
      if $S_{C_p} \subset A$ then
        $\text{score}_{A \subseteq C_p} = \frac{\log \frac{1}{W_{C_p}(A)}}{|\mathcal{L} \cap \psi(A)|}$
      end if
    end for
  end for
  select $A \subseteq C_p$ with the lowest $\text{score}_{A \subseteq C_p}$
  $S_{C_p} = A$
  $\mathcal{L} = \mathcal{L} \setminus \psi(A)$
end while
return $\chi = \bigcup_{C_p \in \mathcal{C}} \; S_{C_p}$
```

Algorithm 5 Inference Algorithm with Incomplete Information

```
$\mathcal{E}^* = \mathcal{E} \setminus \varepsilon(\mathcal{P} \setminus \mathcal{P}_c)$
for $C_p \in \mathcal{C}$ do
  $S_{C_p} = \emptyset$
end for
$\mathcal{L} = \mathcal{P}_c$
while $\mathcal{L} \neq \emptyset$ do
  for $C_p \in \mathcal{C}$ do
    for $A \subseteq C_p$ do
      if $A \subseteq \mathcal{E}^*$ and $\mathbb{P}(\cap_{e_j \in A} X_{e_j} = 1)$ is known then
        $\text{score}_{A \subseteq C_p} = \frac{\log \frac{1}{\mathbb{P}(\cap_{e_j \in A} X_{e_j} = 1)}}{|\mathcal{L} \cap \psi(A)|}$
      end if
    end for
  end for
  select $A \subseteq C_p$ with the lowest $\text{score}_{A \subseteq C_p}$
  $S_{C_p} = S_{C_p} \cup A$
  $\mathcal{L} = \mathcal{L} \setminus \psi(A)$
end while
return $\chi = \bigcup_{C_p \in \mathcal{C}} \; S_{C_p}$
```