Journal of Network and Systems Management (2023) 31:73
https://doi.org/10.1007/s10922-023-09763-y

Network Tomography with Partial Topology Knowledge and Dynamic Routing

Ippokratis Sartzetakis[1,2] · Emmanouel Varvarigos[1,2]

Received: 27 October 2022 / Revised: 31 October 2022 / Accepted: 18 July 2023

Abstract

Networks are always evolving to meet the needs of progressing and novel applications. To this end, improved network capacity, latency and security are required. Monitoring is key to achieving these objectives. It is a cornerstone for the uninterrupted network operation and the applications' QoS management. Network tomography uses a subset of monitoring information, corresponding to partial view of the network state, to estimate wide-sense network performance metrics. In this paper, we present a novel machine learning (ML) formulation for Network Tomography. It is novel in that its features are designed under the assumption that: (i) the existence of certain links of the network is not known (e.g., due to security reasons), (ii) the routing is dynamic (non-deterministic), i.e., for the same origin–destination node pair, a different route may be selected depending on the state of certain links. These assumptions are typically present in modern networks. The formulation can be used to estimate both additive and non-additive performance metrics. We evaluate our proposal using different ML algorithms: neural networks (NNs), Gaussian regression and linear regression with interactions. Our simulations indicate that our ML formulation has better estimation accuracy compared to traditional algebraic or other ML approaches that cannot or do not take into account these two hypotheses. Regarding the accuracy of the examined ML algorithms, the differences mainly depend on the additive or not nature of the network metrics.

Keywords Dynamic routing · Non-deterministic routing · Machine learning · Network tomography · Partial topology knowledge

✉ Ippokratis Sartzetakis
isartz@mail.ntua.gr

Emmanouel Varvarigos
vmanos@mail.ntua.gr

[1] Electrical and Computer Engineering, National Technical University of Athens, Iroon Polytexneiou 9, 15780 Zografou, Greece

[2] ICCS, Patision 42, 10682 Athens, Greece

Published online: 03 August 2023

1 Introduction

Cloud and edge computing infrastructures play an increasingly important role in today's society. They offer a whole range of services, such as online commerce, smartphone applications, video streaming and gaming. Also, in the post-pandemic environment they support to a much greater extent essential sectors of the society in the context of remote work. Their heavy use and rapid deployment makes them more complex, heterogeneous and difficult to manage. The performance of the edge-cloud continuum determines the efficiency of the whole Information and Communication Technology (ICT) infrastructure, and the perceived Quality of Service (QoS) of the deployed applications. The heterogeneity of the networks coupled with the increased traffic volumes and rates, make their real time (dynamic) management both challenging and necessary. Multiple factors can affect network performance, such as: packet loss, abnormal delay, delay variance, bad load distribution and poor behavior of network operating systems or user applications. These factors can result in performance degradation or even hard failures (resulting in network downtime). Consequently, the network costs can significantly and unexpectedly increase. Therefore, it is critical to provide advanced network monitoring tools able to reflect and analyze changes in the network state.

As is the case with any system, a network has to be observable in order to be manageable and stable. Network tomography (NT) [1] is a fundamental tool for this purpose. NT refers to large-scale network inference. It involves estimating network (path) performance metrics based on traffic measurements at a limited subset of network nodes. Indicative monitoring data set includes (but is not limited to): throughput, delay, jitter, packet delivery ratio, congestion, bandwidth for specific paths. These end-to-end metrics are obtained from link-level data according to some associative operator. Some of these metrics are additive per link (e.g., delay, jitter). Other metrics can be transformed to an additive form (e.g., use the log function in the case of the packet delivery ratio or reliability), or they are non-additive (e.g., the congestion level or the bandwidth, depends on the worst link of a path, involving a min operator).

NT improves the observability of the network, using the same amount of information. In other words, NT can reduce the monitoring needs for the whole network. Thus, it increases efficiency, reduces equipment and operating costs and can help verify service level agreements. The more accurate is the knowledge of the network state obtained through NT, the better positioned is the Orchestration layer to maximize the efficiency of resource (e.g., network, compute, and storage) utilization in mixed cloud/fog-edge/HPC environments. For example, in real time communications it is important to estimate in advance the quality of a connection before actually establishing it over a specific path [2]. NT can provide the means for this purpose, as it can leverage known information to provide a related estimate. Moreover, certain security events and anomalies can be promptly detected and dealt with, before they significantly affect the operation of the network.

Monitoring can be passive or active. In active monitoring, certain probes are purposefully deployed to measure the required network parts. These probes create

extra traffic. Therefore they have to be carefully deployed so as to not significantly alter the original network state. In passive monitoring, (part of) the existing traffic is monitored. In all cases, the need to keep the monitoring cost low implies that the number of probes that are deployed or of existing connections that are measured, is kept as small as possible.

A major point of concern is the complexity of the monitoring solution. In large scale networks, vast amounts of monitoring data are generated, making the correlation of the data a very difficult task. Therefore, efficient algorithms need to be developed to infer the appropriate metrics from a minimum amount of data. Another point of concern is that in complex networks a path may cross different and heterogeneous subnetworks. Under these circumstances, the complete network topology is usually not completely known by a single actor. Each individual subnetwork topology is not fully advertised for security and confidentiality reasons. So, a path that is monitored or whose performance we want to estimate may cross different networks and contain an unknown number of links. This makes difficult or even impossible the use of traditional NT algebraic methods. These methods assume complete knowledge of the network topology, where the links of the network are typically represented in a matrix. A solution could be to use a ML algorithm that takes into account the origin and destination node to provide a more accurate estimate as in [3]. However this is not enough in many scenarios. In modern networks the routing between an origin and a destination node may be dynamic (non-deterministic), as it may change based on the state of congestion in the network. So, the same origin and destination node may be routed through different intermediate links, depending on their load or other administrative policies. Therefore, the knowledge of only two nodes of a path is not enough to provide an accurate estimate of meaningful performance metrics. This should also be taken into account when designing a NT algorithm.

In this paper we propose a novel ML formulation for NT. The features of the ML algorithm are designed to take into account the peculiarities of modern networks where the topology may or may not be completely known and the routing may be static or dynamic. Moreover, the features of the ML formulation can provide the required information to estimate both additive metrics, and non-additive metrics. We demonstrate through simulation experiments that the accuracy of our proposal is excellent in a variety of scenarios. In many cases it is far better than that of other ML or pure algebraic approaches. The improved accuracy can help the network operator have a better view of the network conditions and make better optimization decisions. Thus, the overall network efficiency and the applications' QoS are improved.

It is worth noting that this paper is an extension of [4]. We significantly extended our previous work: We added additional background on NT, we enriched the problem formulation, we evaluated our ML formulation using two additional ML algorithms, and we included additional results for a second network topology. The rest of the paper is organized as follows. In Sect. 2 we present the related previous NT work. In Sect. 3 we describe the network scenario assumed. Then we introduce the proposed ML formulation. In Sect. 4 we present the simulation experiments and evaluate the performance of our proposal. Finally, Sect. 5 concludes the paper.

2 Related Work

The term NT was first mentioned in [1]. The initial problem statement was to estimate node-to-node network traffic intensity from link measurements. The subsequent research is vast and ongoing. We therefore will provide a short summary of it here. More information can be found in the references of the mentioned work and surveys. Note that when we refer to delay in this paper, we assume that it comprises of propagation delay and queueing delay. The path delay is the sum of the delays of the links that comprise it. In [5] the authors developed statistical techniques to estimate loss rates on internal links based on losses observed by multicast receivers. They leveraged the correlations between observations to infer the performance of paths between branch points in the tree spanning a multicast source and its receivers. In [6], the authors used unicast end-to-end traffic measurements and developed techniques to estimate link delay distribution. Passive network tomography was first introduced in [7]. The authors assumed the measurement of end-to-end performance metrics of existing traffic, and researched the problem of identifying lossy links. Reference [8] focused on characterizing end-to-end additive metrics. The authors find a minimal set of k linearly independent paths that can describe the metrics of all the other paths. Reference [9] studies a variation of NT. The objective is to identify the lossiest network links using only uncorrelated end-to-end measurements. Using binary metrics, an inference algorithm is proposed that, with high likelihood, identifies the worst performing links. The authors in [10] used multiple and simple one-way measurements among pairs of nodes. Then they estimated the one-way delay between network nodes. In doing so they used a global objective function that is affected by the network topology and not just by individual measurements. The work in [11] employed an algorithm typically used in finance: the General Method of Moments. The technique proved to be favorable for inferring link delays. Reference [12] focused on the problem of identifying link level metrics from end-to-end metrics of selected paths. The authors developed a low-complexity algorithm to construct linearly independent, cycle-free paths between monitors without examining all candidate paths. The authors of [13] assumed a similar problem statement. They investigated the conditions under which the link level metrics could be acquired, depending on the network topology and the number and location of the monitors. In [14] various variations are surveyed, such as link delay inference through multicast end-to-end measurements, origin–destination matrix inference and topology identification. Survey [15] describes subsequent developments in NT. It also presents network coding and compressed sensing to improve estimation accuracy, computational complexity, amount of probing and operational cost.

Another topic is that of failure detection using NT. In this scenario, NT pertains to identifying whether a network node has failed given binary (normal or failed) end-to-end path metrics. In [16] the authors researched the conditions that need to be satisfied in order to identify a bounded number of node failures. They also quantified the maximum number of identifiable node failures and the largest node set within which failures can be localized for a given number of failures.

In [17] the authors provided upper bounds on the maximum number of identifiable failed nodes, considering a certain number of monitored paths, constraints on the network topology, the routing scheme, and the maximum path length. Recently, ML has been applied to the field of NT. The authors of [18] proposed a NT approach for non-deterministic routing where the measured flows for a given origin–destination node pair may cross different paths. In [19] a NN is trained to infer additive metrics in an SDN/NFV environment. The authors in [3] also propose the use of NN to infer metrics based only on the origin and destination node pair, and also to reconstruct the network topology. In [20], assuming in-vehicle network monitoring, NNs are again applied to estimate the performance of an unmonitored part of a network. Finally, in [21] the same principles are applied to the domain of network slicing.

In this paper we consider that the network topology information may be incomplete as in [3]. Therefore, the exact routing, the exact links that a path contains, may not be known. At the same time, the routing may be dynamic (non-deterministic) as in [18]. This means that two paths that serve the same origin and destination node may be routed differently: the may contain a different (sub)set of links. We also assume that the set of the monitored paths at each destination is a given. So, we want to make the most of the available information. We designed a set of ML features that improve the accuracy of the performance metrics estimations under these assumptions. Our contributions are:

- We present a NT approach based on ML, which accounts for incomplete knowledge of the underlying network topology and for dynamic routing.
- Using end-to-end measurements we infer link-level metrics (both additive and certain non-additive) for known links. In case a path crosses unknown parts of the topology the metrics are derived for a subset of the path links.
- Using the above knowledge, we estimate performance metrics for unestablished or unmonitored paths, even with both incomplete topology knowledge and dynamic routing. We perform simulations using two realistic network topologies.
- We consider three ML algorithms (NNs, Gaussian regression, linear regression with interactions). We compare their accuracy under the aforementioned network scenarios.

Our proposal is different to [3] in that we employ additional ML features to account for the unknown parts of the network. The features are particularly useful in cases as in [18], where the routing is dynamic. This provides better accuracy for certain scenarios as we will demonstrate in the simulations. To the best of our knowledge, there is no previous NT formulation that considers both dynamic (non-deterministic) routing and partial network topology knowledge.

3 Network Tomography

In this section we first describe the network scenario that we consider and the related formal notation. Next, we define the problem of the evaluation of additive metrics. We present a basic algebraic method that can be used to solve the problem. Next we define the problem of other associative metrics evaluation. Afterwards, we describe our proposed ML formulation, the employed features, and the ML algorithms that we used to evaluate our framework.

3.1 Network Notation

We consider a network $N = (V, L)$ where V denotes the set of nodes and L the set of known directed links (thus, both (i, j) and (j, i) are present in L if nodes i and j are connected through a unidirectional link). We assume a set P of paths already established in the network. Vector $\mathbf{x} \in \mathfrak{R}^{\mathsf{L}}$ contains the link level parameters. The routing matrix of the established paths is defined as the binary matrix $G_P \in \{0, 1\}^{\mathsf{P}\times\mathsf{L}}$, where $G_P[p, l] = 1$ when path p contains link l, and is 0, otherwise. Consider the end-to-end vector of parameters $\mathbf{y_P} \in \mathfrak{R}^{\mathsf{P}}$. Vector $\mathbf{y_P}$ can represent different performance parameters (e.g., delay or jitter, etc.). The entries of $\mathbf{y_P}$ are obtained from the link parameters according to some monotonic associative operator $\oplus$, that is $\mathbf{y_p} = \oplus_{l=1}^{L} G_P[p, l]x[l]$. By monotonic we mean that either a $\oplus$b $\geq$ a for all a, b ≥ 0, or that a $\oplus$ b $\leq$ a for all a, b ≥ 0 (but not both). Each entry of the path metrics vector is characterized by the manner it is obtained from the link parameter components (that is, by the associative operator $\oplus$). For example, two common end-to-end path metrics (delay, bandwidth) are defined in the following ways respectively:

- $\mathbf{y_p} = \sum_{l=1}^{L} G_P[p, l]x[l], x_l \geq 0$
- $\mathbf{y_p} = \min_{l=1,\ldots,L} G_P[p, l]x[l], x_l \geq 0$

3.1.1 Additive Metrics Evaluation

If the link-level vector parameters $\mathbf{x} \in \mathfrak{R}^{\mathsf{L}}$ are additive per link, vector $\mathbf{y_P}$ can be written as a linear combination of link-level vector parameters $\mathbf{x}$, so that $\mathbf{y_P} = G_P\mathbf{x}$. We assume that we want to estimate the end-to-end parameters of a set M of paths (either new or unmonitored ones), denoted by vector $\mathbf{y_M} \in \mathfrak{R}^{\mathsf{M}}$, given that we know their routing $G_M \in \{0, 1\}^{\mathsf{M}\times\mathsf{L}}$. Then, we have:

$$\mathbf{y} = \begin{bmatrix} \mathbf{y_P} \\ \mathbf{y_M} \end{bmatrix} = \begin{bmatrix} G_P \\ G_M \end{bmatrix} \mathbf{x} \tag{1}$$

Consider, for example, the network of Fig. 1, where a set of $P = \{p_1, p_2\}$ paths are already established and correspond to submatrix G_P and the known end-to-end QoS values $y_P = \{y_1, y_2\}$. The values in vector $\mathbf{y_P}$ can be different for different applications and use cases. The path to be established is denoted by $M = \{m_3\}$

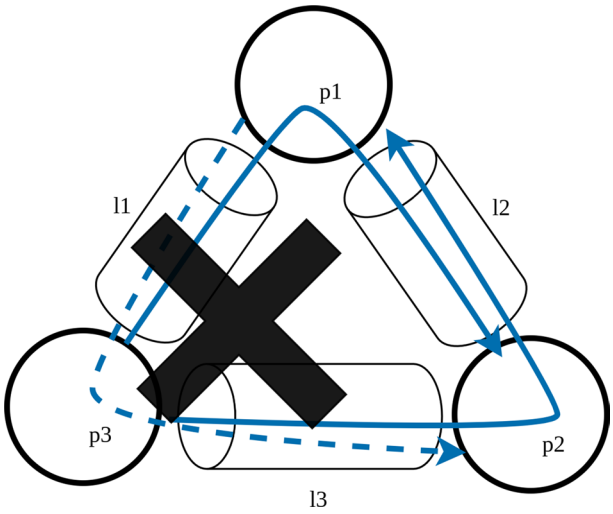

Fig. 1 A network with 2 established monitored paths and one candidate (or unmonitored), sharing one known link and one origin node. Parts of the topology (crossed) may be unknown

whose end-to-end value y_3 we want to estimate. We assume that all the (three) links the paths use are known. The routing can be described as:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{2}$$

Since the new path m_3 contains links that are already in use by other paths, it is possible to estimate its end-to-end value. This can be achieved using the Moore–Penrose inverse $(.)^+$ of G_P (complexity $O(\mathsf{P}^3)$), that is:

$$\hat{\mathbf{y}_\mathbf{M}} = G_M G_P^+ \mathbf{y_P} \tag{3}$$

3.1.2 Other Associative Metrics Evaluation

Certain metrics such as the congestion level, capacity or bandwidth of a path are non-additive per link. In these cases the worst performing link of a path determines the performance of the whole path. The case of bandwidth estimation can be expressed as:

$$\mathbf{y_M} = \min_{1<=i<=P} \left(G_M \circ \mathbf{x}\right) \tag{4}$$

where i is a matrix row, so for every row we select the minimum column element. Symbol $\circ$ denotes the element-wise product of two matrices. In the above equation the vector $\mathbf{x}$ is unknown, so the estimation of a non-additive metric from the

end-to-end metrics is not straightforward. The estimation task is non-linear. It bears some similarities to certain failure localization algorithms. These algorithms either calculate binary metrics for all the network links, or locate one failed link and estimate its value. Equation (4) can be approximated using a ML algorithm. The algorithm can be trained from end-to-end metrics to estimate the maximum (or minimum) value of a link in a path, corresponding to the calculation of the congestion level (or bandwidth) of a path respectively. In the next subsection we present a ML formulation that can be used to determine both additive and non-additive metrics.

3.2 ML Network Tomography

The algebraic formulation of Eq. (3) is normally used when the network topology is completely known. It is not expected to work well when certain links are not known. As we have already mentioned, this is the case in many of the modern networks. In this section we present a ML formulation that takes this into account. The features are chosen to summarize in a heuristic way the important characteristics of the paths with respect to the estimation problem. We organize the feature matrix Q so that each row corresponds to a path $p \in P$, while the columns represent (link or node, feature) pairs, for the links or nodes of the path and the values of the chosen features. In particular, we choose two kinds of features. The first set of features consists of all the known links of the topology (we define it as a set L). The second set of features consists of all the path nodes that can be origin or destination. More specifically, we define S as a $\mathsf{P} \times \mathsf{L}$ link-level feature matrix designed to take into account the known links that a path contains. Element S_{pl}, corresponds to path p and a known link l, and is set equal to 1 if path p contains link l, and equal to zero, otherwise. This feature is the equivalent of the routing matrix G_P. We also define the node-level feature matrix A to represent information regarding the origin and destination node of a path. In particular, A is a $\mathsf{P} \times \mathsf{V}$ node-level feature matrix. Element A_{pv} is equal to 1 if path p starts or ends at node v, and is set to zero, otherwise. Note that this formulation does not specifically take into account which node is origin and which one is destination. However, the information is indirectly captured by the ML algorithm as the set L contains both directions of a link (directed links). We also consider an additional feature to represent the ML bias term (denoted by BT). The bias term can account for monitoring errors or noise that cannot be reduced by any other means. We concatenate all the link-level feature matrices into one feature matrix defined as:

$$Q = [\, BT \;\; S \;\; A \,]_{\mathsf{P}x(1+\mathsf{L}+\mathsf{V})} \tag{5}$$

Similar to Eq. (1), Q_P corresponds to the set of features related to the end-to-end observed QoS values $\mathbf{y_P}$. The features are designed to capture as much information as possible to evaluate the metric at hand. The absence of knowledge of certain links through which the path passes (unknown links) is counterbalanced by the knowledge of the origin and destination node of a path. Moreover, this formulation is better than assuming only the origin and destination node of a path as features. The knowledge of even a small subset of the links that a path contains can greatly increase the accuracy of the estimation. After the features have been defined, an appropriate ML

algorithm can be used to find the relationship between the features and the observed output. The use of matrix A as a feature matrix, makes the problem non-linear. In the following subsections we present three machine learning (ML) algorithms that are expected to work well under the specific circumstances.

3.3 Neural Networks

A NN is a set of connected functions that interpret a set of inputs into a desired kind of output. A NN can be trained to learn a function f through which a set of features explain a respective set of observations. In our case we have:

$$\mathbf{y_P} = f(Q_P) \tag{6}$$

A fully connected NN consists of an input layer, one (or more) hidden layer(s), and an output layer. Each node of one layer is fully connected to all the nodes of the next layer. This enables data propagation from one layer to another. Each layer multiplies the input by a weight matrix and adds a bias term. The hidden layers can be wider (have more nodes) than the input and output. In these cases, the NN may be able to learn more complex relationships between the features and thus have better accuracy. A trial-and-error approach is usually required to find the most suitable architecture. NNs have the potential to represent complex functions and thus estimate both additive and certain non-additive metrics. In our case the specific problem at hand is not very complicated. Therefore, a relatively shallow NN architecture is expected to have good estimation accuracy. In general, a given trained model is expected to perform well without the need for retraining, unless the network state changes significantly. More information about NNs can be found in [22].

3.4 Gaussian Process Regression

A Gaussian process (GP) is a set of random variables. Any subset of these variables is jointly Gaussian. A GP is a stochastic process that can be used to model and approximate nonlinear continuous functions. An application of a GP is regression. The function f of Eq. 6 is modeled as a GP defined by a mean and a covariance (also called kernel) function. The mean function equals to the average of all functions in the distribution for a specific input. The kernel function models the dependence of function values for different inputs. In contrast with other ML models, a GP regression model is generally non-parametric. Thus the number of model parameters is not fixed. For example in a NN the number of hidden-layers must be specified beforehand. In GP the parameters grow as the amount of data increases. However, there still are some parameters that can be optimized, such as length scales. More details about GP regression can be found in [23].

3.5 Linear Regression with Interactions

Linear regression fits a set of inputs to a respective set of outputs using a linear model. An interaction occurs when a variable can affect the output in a different

way depending on the value of another variable. For example, assume a simple two-node network with one link. Then $S = [s_1]$ and $A = [a_1 \ a_2]$. The objective of a simple linear regression model is to calculate the most appropriate thetas (ϑ), such that:

$$y_P = \vartheta_0 + \vartheta_1 \times s_1 + \vartheta_2 \times a_1 + \vartheta_3 \times a_2 \tag{7}$$

Interactions can be added in the sense of product pairs. Then:

$$\begin{aligned} y_P = \vartheta_0 + \vartheta_1 \times s_1 + \vartheta_2 \times a_1 + \vartheta_3 \times a_2 + \vartheta_{12}(s_1 \times a_1) \\ + \vartheta_{13}(s_1 \times a_2) + \vartheta_{23}(a_1 \times a_2) \end{aligned} \tag{8}$$

In our case, the feature vectors S and A depend on each other, since certain links are directly connected (or not) to specific nodes. As we will see in the simulations, interaction effects help to provide an accurate estimate in a variety of scenarios. Reference [24] studies specifically regression with interactions.

In the next section we present the evaluation of different architectures and compare our proposed tomography framework to alternatives.

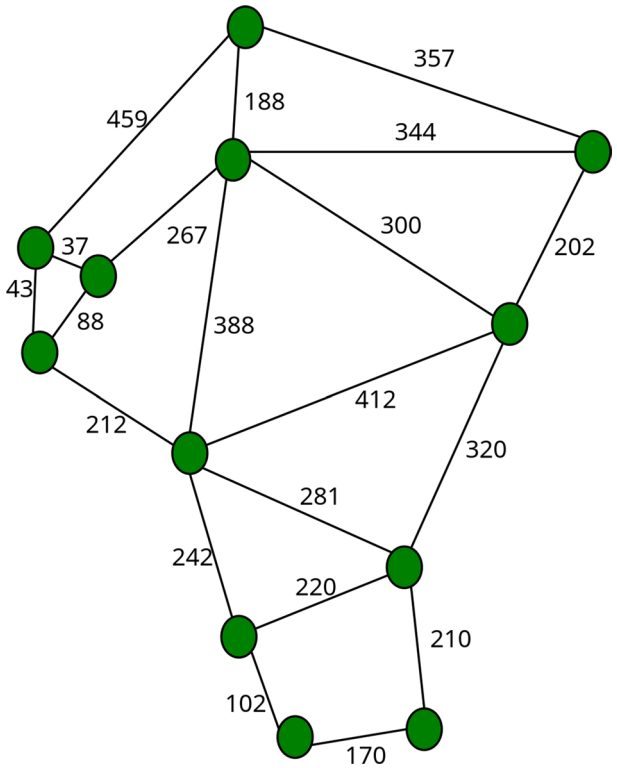

Fig. 2 The DT topology

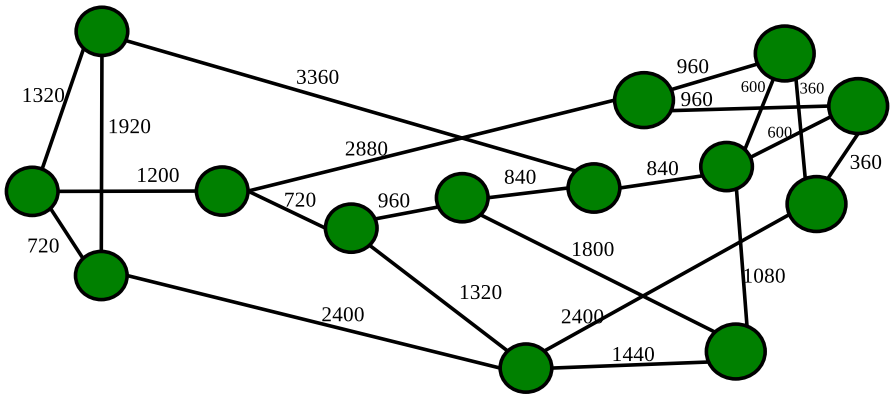

Fig. 3 The NSFNET topology

4 Results

To evaluate our proposed formulation, we performed a number of simulation experiments. We assumed the Deutsche Telecom (DT) topology of Fig. 2 with 12 nodes and 20 unidirectional links. We also considered the National Science Foundation Network (NSFNET) topology of Fig. 3 with 14 nodes and 22 unidirectional links.

In the figures, the length of each link in km is also depicted. We considered that the required estimations metrics are: (i) the delay, which is additive per link, and (ii) the bandwidth, which is non-additive and depends on the worst link of the path (min operation over the links of the path). We set the delay of each link to be numerically equal to its length. So, the delay of a path is equal to the sum of the delays of its links. Without loss of generality, we set the bandwidth of each link to be numerically equal to its length. The bandwidth of a path equals to the minimum bandwidth of the links that it contains. For the DT topology, we assumed various different loads of 100, 200, 300, 400, 500 connections with uniformly chosen source-destination nodes. For the NSFNET topology we assumed up to 1000 connections. In the simulations we will explain why NFSNET requires a larger amount of connections. For each source-destination pair we used a k-shortest path algorithm with $k = 1, 2, 3$ to decide the routing. When $k > 1$ the specific route for each source-destination pair was chosen uniformly over these k paths. We also assumed an increasing number of unknown links. For a given link that is considered unknown, we removed its related value from the routing matrix G, and from the feature matrix S.

Simulations were performed in MATLAB using a quad core CPU@3GHZ. We used 90% of the established paths for training, and 10% for testing. We exclude from the testing set any path that contains a link that is not used at all in the training set. This case typically arises in small training datasets. In a practical setting, this scenario could appear for example in a new network where there are insufficient related data. In these cases active monitoring can be used to obtain the necessary information. We employed a trial-and-error approach to find the most suitable architecture and hyperparameters that exhibit good accuracy. We run 100 independent iterations

and averaged the results for each case. The training of the NN was performed using 2000 epochs. The ReLu activation function and the Mean Square Error (MSE) loss function were employed. Three total layers were used. The size of the first two was equal to two times the number of features. The size of the third layer was equal to the number of features. The linear interaction algorithm contained a linear term for each predictor, an intercept and all products of pairs of the predictors, with no squared terms. The Gaussian regression algorithms used a rational quadratic kernel (covariance) function and a constant basis function: a vector of basis coefficients whose length is the number of predictors is added to the model.

4.1 Additive Metrics Evaluation

First, we evaluated the three different ML algorithms that employed the features described in section III.B. In Fig. 4 we present the estimation accuracy results for the estimation of the delay in the DT topology. We plot the accuracy in terms of mean absolute error (MAE) as a function of the number of established paths in the network. In Fig. 4a, $k = 1$, so each source-destination pair can only be served by one path (static routing). The accuracy of the three algorithms is very similar and increases as the number of paths increases (more data leads to better training). As k increases (Fig. 4b and c), a source-destination pair may be routed over different paths (non-deterministic or dynamic routing). This means that the algorithms should be able to correlate a larger amount of information to provide an accurate output (i.e., the origin and destination pair by themselves do not contain the required information for an accurate estimation). Again, all algorithms seem to have similar performance. However, a larger amount of established paths is required to achieve good accuracy. The reason is that the algorithms need additional information to understand the dynamic routing (the relationship between the features and the outputs). Overall, linear regression with interactions has better accuracy than the other algorithms. One possible explanation is that the estimation of the delay is additive per link. Linear regression is expected to work well under these circumstances. The interactions help to interpret the additional features of the origin and destination node when needed (i.e., in cases of dynamic routing). NNs and Gaussian regression are particularly good in the approximation of non-linear relationships. As we will see in the results of the b/w estimation they achieve better accuracy.

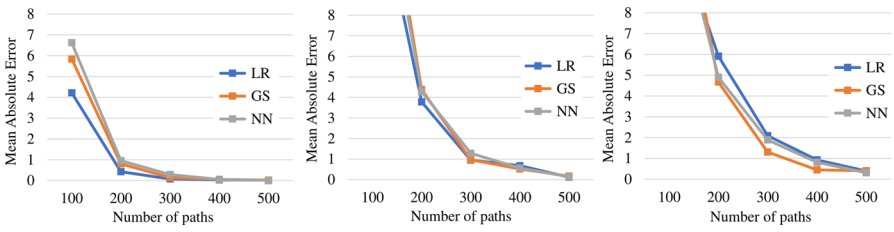

Fig. 4 Accuracy comparison (delay metric) of three ML algorithms for different number of paths and **a** k = 1, **b** k = 2, **c** k = 3 (DT topology)

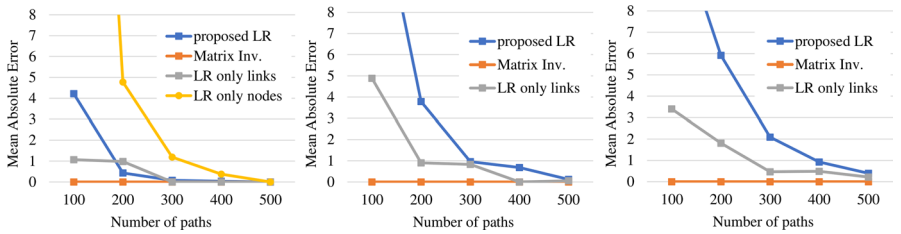

Fig. 5 Accuracy (for the delay metric) of the proposed ML formulation and alternatives for different number of paths and **a** k = 1, **b** k = 2, **c** k = 3 (DT topology)

We then compared the accuracy of four algorithms: (i) our proposed ML formulation, (ii) a ML formulation where the features are only the links of the paths, (iii) a ML formulation where the features are only the origin and destination nodes of the paths, and (iv) a traditional algebraic matrix inversion solution. All ML formulations used linear regression with interactions, that was shown to have better accuracy. Note than when we mention confidence interval, we refer to the 95% confidence interval. Figure 5 presents the MAE of the four examined algorithms for different number of measured paths in the DT topology. In Fig. 5a, where $k = 1$, we notice that a simple matrix inversion can achieve good accuracy even with a relatively small number of measured paths. The algorithms that are based on LR require a larger number of established paths to be trained so as to have good accuracy. The LR that uses only the links as features, achieves slightly better accuracy earlier than the proposed LR (in this scenario). The reason is that the proposed LR has more features thus requiring a larger training set. The LR that only uses the origin and destination node requires the largest training set. The reason is that the algorithm needs to be trained with all possible combinations of origin and destination nodes in order to be able to provide an accurate estimate. For 500 established paths, all algorithms have MAE approximately 10^{-4}, the maximum error is less than 10^{-3} delay units and the confidence interval is in the order of $\pm 10^{-4}$. In Fig. 5b and c, where we have $k > 1$, the simple matrix inversion method still achieves the best accuracy in all cases (the MAE and the maximum error are similar to the case where $k = 1$). The LR that relies only on the origin and destination node does not have enough information to provide an accurate estimate. Its accuracy is not depicted in these figures,

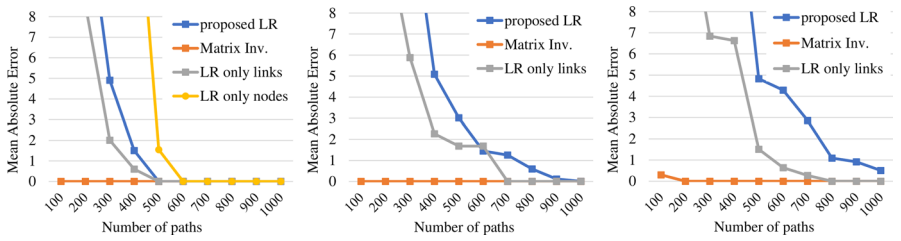

Fig. 6 Accuracy (for the delay metric) of the proposed ML formulation and alternatives for different number of paths and **a** k = 1, **b** k = 2, **c** k = 3 (NSFNET topology)

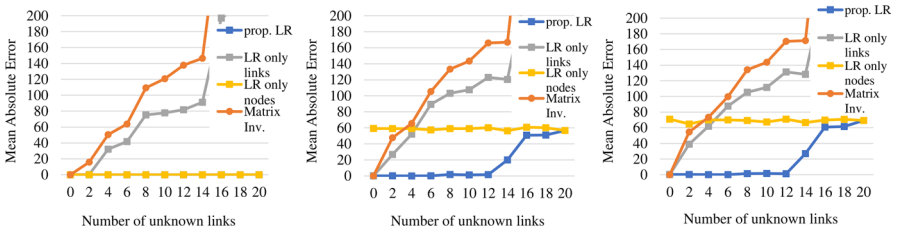

Fig. 7 Accuracy (delay metric) of the algorithms for different number of unknown links and **a** k = 1, **b** k = 2, **c** k = 3 (DT topology)

but is examined in Fig. 7. The other algorithms require larger number of established paths to achieve comparable accuracy, since the non-deterministic routing requires more training data in order for the LR to understand it. In the case of 500 paths, the MAE of both algorithms are approximately the same, and it is in the order of 10^{-1} for $k = 2$ and 3×10^{-1} for $k = 3$. The maximum error is approximately 200 and 300 and the confidence interval is in the order of $\pm$ 1 and $\pm$ 2 for $k = 2$ and $k = 3$, respectively.

In Fig. 6 we can see again the accuracy (in terms of MAE) of our proposal compared to alternatives as in Fig. 5, but for the NSFNET topology. We can see similar trends as in Fig. 5. The difference here is that a significantly larger amount of established paths compared to the DT is required to achieve good accuracy. This can attributed to a few reasons. The NSFNET topology has 2 additional links and 2 additional nodes. This means that additional paths are required to understand both the deterministic routing ($k = 1$), and the dynamic routing ($k > 1$) respectively. For the case of the DT topology, when $k = 1$, there are 56 unique paths between all the nodes. In the NSFNET topology, there are 82 unique paths. When $k = 3$ there are 168 and 246 paths for the DT and NSFNET respectively. Finally, the NSFNET topology has a number of significantly longer links than the DT topology. This means that the MAE will by definition be larger, even when all other comparison parameters are the same. Nevertheless, the accuracy of the proposed algorithm in the NSFNET topology is still good. When $k = 1$ approximately 700 paths are required to achieve similar accuracy to the DT (using 500 paths). When $k > 1$, approximately 1000 paths are required for the accuracy to be comparable to that of the DT topology (again using 500 paths). The reason as we mentioned above is the additional routing combinations that the NSFNET topology has.

In Fig. 7 we assumed 500 established paths in the DT topology and we examine the MAE of the algorithms for an increasing number of unknown links. In Fig. 7a, k is equal to 1. Note that the blue line of the proposed LR is under the yellow one of the LR that employs as features only nodes. The matrix inversion and the neural network that relies only on the links contained on a path, exhibit the worst accuracy as the number of unknown links increases. The reason is that these algorithms do not have the appropriate information to compensate for the missing topology knowledge. The proposed LR has a bit better MAE than the LR that uses only the origin and destination node as features. The MAE and the maximum error of our proposal

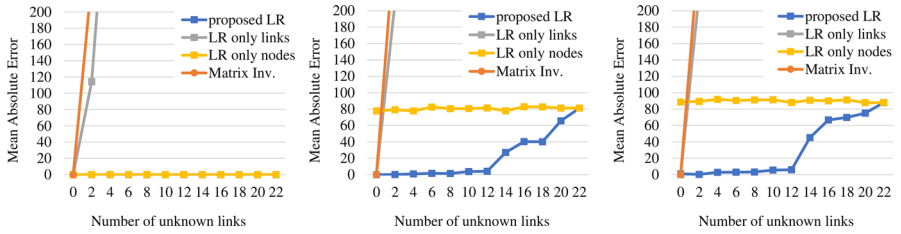

Fig. 8 Accuracy (delay metric) of the algorithms for different number of unknown links and **a** k = 1, **b** k = 2, **c** k = 3 (NSFNET topology)

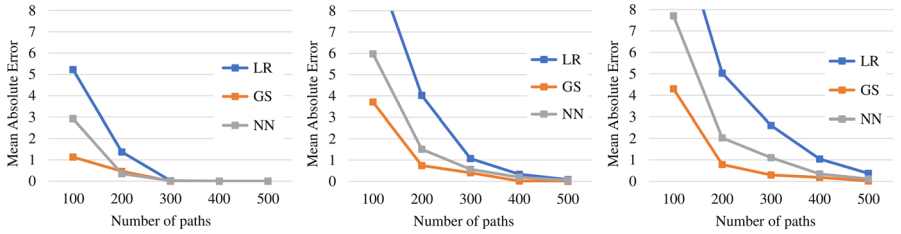

Fig. 9 Accuracy comparison (b/w metric) of three ML algorithms for different number of paths and **a** k = 1, **b** k = 2, **c** k = 3

is consistent for almost all the cases of unknown links. The algorithm is able to compensate for the unknown network links by using the origin and destination node to estimate the delay of each path. In Fig. 7b and c we have $k > 1$. We notice that the proposed LR still achieves good accuracy even with 12 unknown links. After that, its accuracy deteriorates rapidly, and with 20 unknown links (all the links of the network), its accuracy is (as expected) equal to the LR whose features are only the origin and destination node. For both $k = 1$ and $k = 2$, the MAE is approximately 2 until the unknown links are 12. The maximum error is above 200 in almost all cases and the confidence interval is $\pm$ 2. As we can see, the combination of the features of the proposed LR work really well in this scenario. The two classes of features can work together and provide estimates even for a large number of unknown links, and under non-deterministic routing ($k > 1$). This is particularly useful in modern network scenarios where the topology of the network may not be completely known due to security reasons and also connections with the same origin and destination nodes may be routed differently over the physical topology.

In Fig. 8 we assumed 1000 established paths in the NSFNET topology, and we again examined the accuracy for an increasing number of unknown links. As we can see, the algorithms have in this case the same behavior as in the case of the DT topology. The only difference is that additional established paths are required to achieve similar accuracy, for the same reasons as described in Fig. 6.

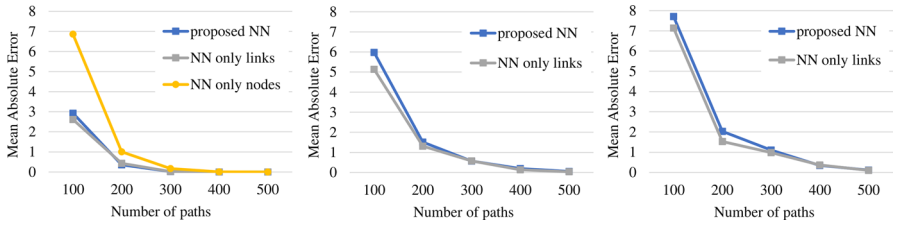

Fig. 10 Accuracy (for the b/w metric) of the proposed ML formulation and alternatives for different number of paths and **a** k = 1, **b** k = 2, **c** k = 3 (DT topology)

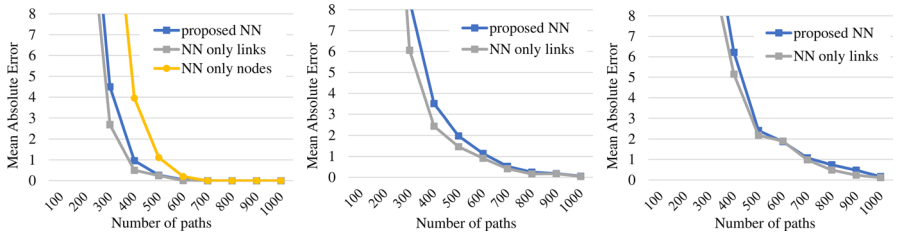

Fig. 11 Accuracy (for the b/w metric) of the proposed ML formulation and alternatives for different number of paths and **a** k = 1, **b** k = 2, **c** k = 3 (NSFNET topology)

4.2 Non-additive Metrics Evaluation

In this subsection we focus on the estimation of bandwidth, which is a non-additive metric.

In Fig. 9 we can see the accuracy comparison of the three ML algorithms using our proposed ML formulation, as in the case of Fig. 4. Here we can see that the LR's accuracy is a bit worse than that of GS and NN. Nevertheless, the accuracy of the three algorithms converge for 500 paths. The bandwidth estimation is a non linear estimation task and this is probably why LR performs worse compared to the case of the delay estimation. Overall, NN and GS have a bit better accuracy than LR even for 500 paths. Therefore, we chose NN for the subsequent experiments.

In Fig. 10, we compare the accuracy of the proposed formulation to two other alternatives. In Fig. 10a, where $k = 1$, we can see that all algorithms achieve excellent accuracy with a relatively low number of measured paths. The MAE for 500 paths is less than 10^{-3} for all the algorithms, the maximum error is less than 10^{-2} and the confidence interval $\pm 10^{-5}$. In Fig. 10b and c where $k > 1$, the accuracy of the NN that uses only the origin and destination nodes is again not good, since it still lacks the necessary information to provide an accurate estimate. The other two algorithms have similar performance. Their MAE is in the order of 10^{-2} and 10^{-1}, the maximum error is 32 and 100 and the confidence interval $\pm 10^{-1}$ and $\pm 5 \times 10^{-1}$ for $k = 2$ and $k = 3$ respectively.

In Fig. 11 we examine again the bandwidth estimation accuracy of our proposal compared to alternatives as in Fig. 10, but for the NSFNET topology. We notice similar trends as in the case of the delay estimation in DT and NSFNET: additional

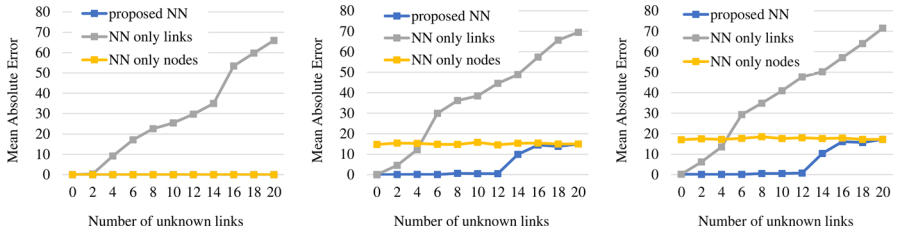

Fig. 12 Accuracy (b/w metric) of the algorithms for different number of unknown links and **a** k = 1, **b** k = 2, **c** k = 3 (DT topology)

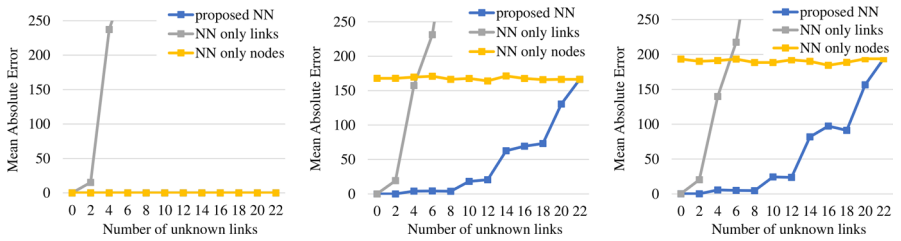

Fig. 13 Accuracy (b/w metric) of the algorithms for different number of unknown links and **a** k = 1, **b** k = 2, **c** k = 3 (NSFNET topology)

paths are required to achieve good estimation accuracy. For 1000 paths the MAE is in the order of 10^{-5}, 10^{-2}, 10^{-1}, the maximum error is 10^{-4}, 250 and 300 and the confidence interval $\pm 2 \times 10^{-5}$, $\pm 8 \times 10^{-1}$, ± 2 for $k = 1, k = 2$ and $k = 3$ respectively.

Finally, in Figs. 12 and 13 we considered 500 and 1000 established paths in DT and NSFNET topology respectively. We again examined the MAE of the three algorithms for an increasing number of unknown links. For all the algorithms we notice behavior similar to the case of Figs. 7 and 8. In the case of the DT topology, for $k = 1$ the MAE, the maximum error and the confidence interval of the proposed NN is in the order of 10^{-5}, 10^{-3} and $\pm 10^{-5}$ respectively. For $k > 1$ the accuracy deteriorates after 12 unknown links. Until then the proposed ML formulation can provide accurate estimates (MAE: 10^{-1}, max. error: 50 and conf. interval: $\pm 10^{-1}$). In the case of NSFNET topology, for $k = 1$ the MAE and the maximum error are in the same order as in the DT topology (albeit with different number of established paths). For $k > 1$ the accuracy is good until 8 unknown links. After that, the accuracy deteriorates rapidly. For $k = 3$ the deterioration is even more significant. Still, the accuracy is good compared to the alternatives. For a larger amount of established paths, the performance could be even better.

Overall, the results indicate that our proposed ML formulation can achieve good estimation accuracy for both additive and the non-additive metrics examined and for different topologies. When compared to other ML formulations, our

proposal has substantially better performance in almost all cases. In the case of delay estimation, a simple matrix inversion has better accuracy when the network topology is completely known. When the topology knowledge is incomplete, our proposal is significantly more accurate.

5 Conclusions

In this paper we presented a novel ML formulation for Network Tomography under the assumptions of partial topology knowledge and dynamic (non-deterministic) routing. We designed ML features that suit these assumptions that are present in modern networks. We evaluated three different ML algorithms that employ the same features: NNs, Gaussian regression and linear regression with interactions. The results indicate significant improvement in estimation accuracy for both additive and non-additive metrics when compared to other approaches. The improvement is prominent in cases where the number of unknown links is large and there are many different routing decisions for a given source-destination pair. Linear regression with interactions had the best performance in the case of delay estimation. Gaussian regression and NNs had better accuracy for the task of bandwidth estimation. Future work includes the localization of failures.

Acknowledgements This work was partially supported by the Horizon 2020 SERRANO project (Grant Agreement: 101017168).

Author Contributions IS wrote the main manuscript text and prepared the figures. EV also wrote the manuscript text, supervised and directed the work.

Funding Open access funding provided by HEAL-Link Greece.

Data Availability Data available upon request.

Declarations

Conflict of interest The authors declare no competing interests.

Ethical Approval Not applicable.

References

1. Vardi, Y.: Network tomography: estimating source–destination traffic intensities from link data. J. Am. Stat. Assoc. **91**, 365–377 (1996)
2. Rødbro, C.A.,Chou, P.A., Dogan, Ü.: Prediction of Bandwidth and Additive Metrics for Large Scale Network Tomography. Microsoft Research Technical Report, MSR-TR-2016-57
3. Ma, L., Zhang, Z., Srivatsa, M.: Neural network tomography. arXiv preprint (2020). arXiv:2001.02942
4. Sartzetakis, I., Varvarigos, E.: Machine learning network tomography with dynamic routing and partial topology knowledge. In: Proceedings of the IEEE Globecom Conference, Rio de Janeiro, Brazil (December 2022)
5. Cáceres, R., Duffield, N.G., Horowitz, J., Towsley, D.F.: Multicast-based inference of network-internal loss characteristics. IEEE Trans. Inf. Theory **45**(7), 2462–2480 (1999)
6. Duffield, N.G., Horowitz, J., Presti, F.L., Towsley, D.: Network delay tomography from end-to-end unicast measurements. In: Proceedings of the Thyrrhenian International Workshop on Digital Communications. Springer, Berlin (2001)
7. Padmanabhan, V.N., Qiu, L., Wang, H.J.: Passive network tomography using Bayesian inference. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement, Marseille, France (November 2002)
8. Chen, Y., Bindel, D., Katz, R.H.: Tomography-based overlay network monitoring. In: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement, Miami Beach, USA (October 2003)
9. Duffield, N.: Network tomography of binary network performance characteristics. IEEE Trans. Inf. Theory **52**(12), 5373–5388 (2006)
10. Gurewitz, O., Cidon, I., Sidi, M.: One-way delay estimation using network-wide measurements. IEEE Trans. Inf. Theory **52**(6), 2710–2724 (2006)
11. Chen, A., Cao, J., Bu, T.: Network tomography: identifiability and Fourier domain estimation. IEEE Trans. Signal Process. **58**(12), 6029–6039 (2010)
12. Ma, L., He, T., Leung, K.K., Towsley, D., Swami, A.: Efficient identification of additive link metrics via network tomography. In: Proceedings of the 33rd IEEE International Conference on Distributed Computing Systems, Philadelphia, USA (July 2013)
13. Ma, L., He, T., Leung, K.K., Swami, A., Towsley, D.: Inferring link metrics from end-to-end path measurements: identifiability and monitor placement. IEEE/ACM Trans. Network. **22**(4), 1351–1368 (2014)
14. Castro, R., Coates, M., Liang, G., Nowak, R., Yu, B.: Network tomography: recent developments. Stat. Sci. **19**(3), 499–517 (2004)
15. Kakkavas, G., Gkatzioura, D., Karyotis, V., Papavassiliou, S.: A review of advanced algebraic approaches enabling network tomography for future network infrastructures. Future Internet **12**(2), 20 (2020)
16. Ma, L., He, T., Swami, A., Towsley, D., Leung, K.K.: Network capability in localizing node failures via end-to-end path measurements. IEEE/ACM Trans. Network. **25**(1), 434–450 (2017)
17. Bartolini, N., He, T., Khamfroush, H.: Fundamental limits of failure identifiability by Boolean network tomography. In: Proceedings of the IEEE INFOCOM Conference on Computer Communications, Atlanta, USA (May 2017)
18. Tagyo, R., Ikegami, D., Kawahara, R.: Network tomography using routing probability for undeterministic routing. IEICE Trans. Commun. **E104.B**(7), 837–848 (2021)
19. Rahali, M., Sanner, J., Rubino, G.: TOM: a self-trained tomography solution for overlay networks monitoring. In: Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, USA (January 2020)
20. Ibraheem, A., Sheng, Z., Parisis, G., Tian, D.: Neural network based partial tomography for in-vehicle network monitoring. In: Proceedings of the IEEE International Conference on Communications Workshops (ICC), Virtual Conference (June 2021)
21. Rkhami, A., Hadjadj-Aoul, Y., Rubino, G., Outtagarts, A.: On the use of machine learning and network tomography for network slices monitoring. In: Proceedings of the IEEE 22nd International Conference on High Performance Switching and Routing (HPSR), Paris, France (June 2021)
22. Bishop, C.M., Nasrabadi, N.M.: Pattern Recognition and Machine Learning. Springer, New York (2006)

23. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT, Cambridge (2005)
24. Aiken, L.S., West, S.G., Reno, R.R.: Multiple Regression: Testing and Interpreting Interactions. Sage, London (1991)

Ippokratis Sartzetakis received the Ph.D. degree in resource allocation for elastic optical networks in 2019 from the National Technical University of Athens (NTUA), Athens, Greece. He is currently a Postdoctoral Researcher with NTUA. His research interests include edge-cloud computing, network resource management, and machine learning.

Emmanouel Varvarigos received the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1992. Since 2015 he is a Full Professor with the School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece. His research activities are in the areas of optical communication networks, optical networking algorithms and protocols, optical interconnects for Data Centers, network protocols, 5G networks, and network services.