HAL
open science

Unicast Inference of Additive Metrics in General Network Topologies

Mohamed Rahali, Jean-Michel Sanner, Gerardo Rubino

To cite this version:

Mohamed Rahali, Jean-Michel Sanner, Gerardo Rubino. Unicast Inference of Additive Metrics in General Network Topologies. MASCOTS 2019 - 27th IEEE International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Oct 2019, Rennes, France. pp.107-115, 10.1109/MASCOTS.2019.00021 . hal-02430585

HAL Id: hal-02430585

https://inria.hal.science/hal-02430585

Submitted on 7 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

In MASCOTS'19

Unicast inference of additive metrics in general network topologies

Mohamed Rahali[1], Jean-Michel Sanner[1], and Gerardo Rubino[2]

[1]Firstname.Lastname@b-com.com, IRT B<>COM, Rennes, France
[2]Gerardo.Rubino@inria.fr, INRIA Rennes – Bretagne Atlantique, France

***Abstract*—Internet tomography studies the inference of the internal network performances from end-to-end measurements. Unicast probing can be advantageous for such monitoring solutions due to the wide support of unicast and the easy deployment of unicast probing paths. In this work, we propose two statistical generic methods for the inference of additive metrics using unicast probing. Our solutions give more flexibility in the choice of the collection points placement, the probed paths and they are not limited to specific topologies. Firstly, we propose the *k-paths* method that extends the applicability of a previously proposed solution called *Flexicast* for tree topologies. It is based on the Expectation-Maximization (EM) algorithm which is characterized by high computational and memory complexities. Secondly, we propose the Evolutionary Sampling Algorithm (ESA) that enhances the accuracy and the computing time but following a different approach.**

I. Introduction

Future networks will be characterized by high scalability and autonomy thanks to SDN [1] technologies. This agility in the network resources management and services deployment requires efficient monitoring tools to ensure the knowledge of the network state. The evaluation of the network monitoring systems is usually based on their accuracy and solicited network resources. Finding a trade-off between these two criteria is usually a challenging research topic.

The direct sampling of link-level metrics over large-scale IP networks, using legacy, device-centric and agent-based monitors like SNMP [2], is usually costly and cumbersome. This explains why device-centric monitoring solutions are in-practice used mostly for static and small IP networks, such as LAN or enterprise networks. The monitoring of high-speed IP core networks is mainly based on traffic sampling protocols like NetFlow and sFlow [3]. A configurable traffic sampling strategy of the crossing flows is implemented, which controls the trade-off between measurement overhead and accuracy. The resulting global monitoring solution is often fragmented, difficult to use and evolving.

The use of network tomography [4] [5] to infer link level performances from end-to-end observations can be a promising solution. These techniques give a global overview of the network state without collecting link level metrics from all the equipements. Another motivation for using end-to-end measurements is having an accurate view of the network state without consuming important network resources such as bandwidth, CPU and memory. The probes may be actively injected by the origin or passively assimilated at the destination based on the exchanged user traffic. The second probing method is preferred since it reduces the consumed resources at the intermediate nodes and bandwidth usage of network links. However, it is less accurate and may be biased.

Early inference algorithms of additive metrics focus on multicast probing [6]. Several enhancements have been introduced afterward in order to increase their usefulness and performance under different network conditions, as will be detailed in Section II. Recently, considerable attention has been paid to unicast network tomography, stimulated by the emerging SDN enabling to easily install customized network paths. In such a case, network tomography may be implemented as an SDN application running over a controller that manages a programmable data plane. This architecture can potentially extend network tomography use-cases.

However, recent works on unicast tomography concentrate on "identifiability", that is, on finding necessary conditions, e.g. number and placement of traffic collector points, origin-destination pairs, and probing paths, enabling to infer link-level metrics in an exact way using unicast probing [7], [8]. These conditions are often hard to meet at a reasonable cost in practice, especially over dynamic networks. That is why we propose in this work using statistical methods applied on unicast paths for inferring additive link-level metrics.

In this paper we propose two inference methods for additive link-level metrics (delay or loss rate at logarithmic scale) using unicast paths. In the first proposal, *k-paths*, we adapt the solution proposed in [9] based on the Expectation-Maximization (EM) algorithm to general topologies. The second proposal, called Evolutionary Sampling Algorithm (ESA), is a technique based on random sampling designed to avoid the high computation and memory complexity of the EM-based solutions while keeping a good accuracy in the evaluations. The adaptation proposed in *k-paths* gives us the possibility to compare our principal proposal ESA with previously proposed works in the literature.

The remainder of the paper is organized as follows. Section II overviews the existing works related to network tomography for the inference of additive metrics. Sections III and IV present respectively the problem formulation and the proposed solutions. Then, we study and compare their performances in Section V. Finally, Section VI concludes the work and outlines some perspectives for future developments.

II. Related work

Statistical inference approaches examine a set of observed measures in order to perform a mapping from path-level to link-level metrics. Generally, this is formulated as a likelihood

function that needs to be maximized. These solutions may be classified according to the probing schemes.

A. Metrics inference with multicast probing

Authors in [6] propose a solution to infer loss rates in a network topology tree using multicast probing. Basically, a designated root node actively sends multicast probes to all leaves where the monitoring traffic is collected. While this method has a good performance in term of accuracy and convergence, it requires to satisfy multiple constraints for its deployment. The support of multicast in the intermediate nodes and the needed number of hosts in the leaves to collect detailed packet traces are the two main drawbacks to handle to meet in practice.

Recently, more attention has been accorded to the applicability of multicast probing with general network topologies. In [10] and [11], the authors propose to use overlapping multicast trees to cover the network topology. Different Maximum Likelihood Estimators (MLE) are proposed to infer the loss rates in the intermediate links from the measurements performed over the multicast trees. These solutions have the same limitations as [6].

B. Metrics inference with unicast measurements

Less restrictive and lightweight unicast-based solutions are proposed in [12]. Basically, a server examines the native feedback of established TCP connections with a set of clients in order to passively infer observed metrics at each client side. Using that data and knowing the delivery topology, authors designed three inference algorithms to identify the lossy links. In spite of its easy deployment, such a solution is unable to infer link-level metrics with satisfactory precision. As such, it may be only used as a preliminary support tool for network diagnostic, as explained by the authors.

In [9], a probing framework called *Flexicast* for link delay inference in a tree is proposed. Delays (the target) are discretized and a mixed method of probing between unicast and multicast is designed. The inference problem is formulated as the maximization of a likelihood function with latent variables. The maximization technique used is the EM algorithm. Paper [13] deals with the same issue and proposes an optimized implementation of the EM algorithm. In fact, it proposes to memorize some redundant operations in order to reduce the computing complexity. A fast method to detect the worst performing links in a network topology tree is described in [14] with a likelihood inference algorithm. In [15], the authors propose a link-level inferring solution that identifies the maximal multicast probing spanning tree over a given general network topology. Therefore, multicast probing trees are created where conventional multicast MLE can be applied. The remaining links are covered with unicast probing paths.

III. General context and problem description

A. General context

Typically, a monitoring tomography solution can have three main parts: choosing the points for traffic data collection, deploying a probing strategy and inferring the internal network state. The best method to evaluate the efficiency of the first and the second steps is studying their impact on the quality of the estimations. Then, the monitoring strategy can be adapted regularly according to its efficiency and to the network topology updates. Thus, in this work, we concentrate on the inference part since it is the key of internet tomography solutions. We consider the list of probed paths as known parameters. Once the monitoring data is collected, our solution tries to infer the network state. A complete strategy for the probing scheme deployment is in the scope of our future works.

B. Network model and notations

The network topology is modeled by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ is the set of network vertices and $\mathcal{L}$ is the set of network links. Let us denote by $\mathcal{P}$ the set of given probed paths. The cardinalities of sets $\mathcal{V}$, $\mathcal{L}$, $\mathcal{P}$ are respectively V, L, P, and we denote $\mathcal{V} = \{1, 2, \ldots, V\}$, $\mathcal{L} = \{1, 2, \ldots, L\}$, $\mathcal{P} = \{1, 2, \ldots, P\}$. Each path π is represented by a boolean vector of size L, where $\pi(\ell)$ is equal to 1 if link ℓ belongs to π. A is a boolean matrix with rows composed of the probed paths vectors (dimensions $P \times L$). Thus, $A(p, \ell)$ is equal to 1 if path p uses link ℓ.

Y denotes the vector of the observed path metrics. Hence, its size is P and the pth element $Y(p)$ represents the metric on path p. X denotes the vector of the unknown link metrics. Thus, its size is L and its ℓth element $X(\ell)$ is the metric value corresponding to link ℓ.

For efficiency reasons, we discretize the set of values that the additive considered metric can take, and we assume known an upper bound of those values. Referring to the target metric on network links, we denote by B this bound, and we consider that the set of possible values is $\{0, \Delta, 2\Delta, \ldots, J\Delta = B\}$ (that is, $\Delta = B/J$).

In our approach, X is a random variable with a discrete probability distribution α, seen as an $L \times J$ matrix. Hence, $\alpha(\ell, j)$ represents the probability of observing value $j\Delta$ on link ℓ. Our objective is then to get an accurate approximation of the probability distribution α. Observe that the elements of every row of matrix α sum up 1.

C. The linear model

Recall that the objective is to estimate the link metrics from end-to-end measurements. The problem can be modeled by the following system of linear equations:

$$AX = Y. \tag{1}$$

This linear system is in the practical cases undetermined (that is, $P < L$ –and often $P \ll L$). So, there is an infinite number of vectors X satisfying (1). The point here is that we have more information available, we know the network topology leading to matrix A, that makes that the random components of vector X will exhibit, in general, interdependencies. This information can be used to find an appropriate approximation to solutions to the previous system of equations. This can be achieved by means of statistical methods, that can

provide good approximations of the distribution of vector X, for instance maximizing the likelihood of the observed data.

Hence, statistical methods can be adopted to find the probability distribution of the link metrics that maximizes the likelihood of the observed data.

IV. Tomography algorithms for general network topologies

A. Introduction

This section introduces our network tomography solutions for general network topologies using unicast probing. We start by the *k-paths* method, that as previously stated, is a direct extension of the proposal in [9]. In this paper it also serves to provide performance indications about our ESA algorithm.

In [9], the authors propose a solution for additive metrics inference in tree topologies using unicast probing. The measurements are performed between the root of the tree and the leaves. This problem corresponds to the previously introduced linear inverse problem given by (1). The EM algorithm is used to find the most likely solution that matches the system. The principle in this approach is to divide the linear system into multiple smaller sub-systems. Each sub-system is equivalent to a sub-tree. The division is managed to have correlated information based on common links, available at the leaves of each sub-tree. The algorithm then solves each sub-tree. At the end, the EM uses these solutions as inputs to solve the global system.

An immediate observation here is that switching from a tree topology to an arbitrary graph reveals other optimization problems related to the placement of the traffic collection points and the probed paths. In fact, in a tree topology, the traffic is injected at the root node and collected at the leaves. More sophisticated probing strategies are needed with general network topologies.

B. k-paths unicast probing schemes

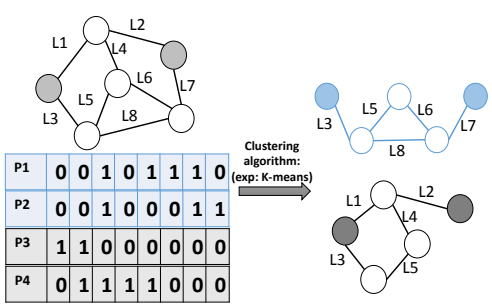


Fig. 1: Network topology splitting. For clarity in the picture, we denoted $L1, L2, \ldots$ the links, $P1, P2, \ldots$ the paths.

In the *k-paths* method, the main idea of [9] is followed, which is dividing the global linear system into multiple sub-systems with smaller sizes. However, we propose different ideas for splitting linear system (1) and for the resolution of the obtained sub-systems that are not limited here to tree topologies. Basically, we gather the different paths in clusters maximizing the number of shared links between the paths inside them, and minimizing the number of links in each one of them.

For these purposes, a clustering algorithm is used based on reducing a distance (for example, Euclidian, or Hamming) inside the obtained clusters. The algorithm takes as its input the list of the probing paths vectors encoded in matrix A and tries to gather them with maximum distance similarity as described in Fig. 1. The *k-means* algorithm can be used for this task. Then, a recursive solver based on a Depth-First-Search is used to explore the solutions space and find the possible ones. Last, the EM algorithm starts from the obtained solutions and sweep them to find the best one for the global linear equation system.

Some remarks before providing the details. From a global point of view, what we want is a procedure such that given a single observation Y, it builds an approximated value of the link-level metrics X. In the process, from this single observation we build a potentially huge set $\mathcal{X}$ of possible vectors X (the first step), from which a first approximation of the distribution of X is computed, then refined following the EM algorithm. If this set $\mathcal{X}$ is too big, we can sample it to continue the computations with a subset of candidates having a more reasonable size. This will not be explored in deep here; the considered examples allow to deal with the space of all potential candidates.

A last remark concerning targeting α, that is, the distribution of X, and not, say, $\text{Exp}(X)$. In this paper, we illustrate the approach with the problem of approximating X from the observation of Y, but more complex applications are possible, and we mention one at the beginning of Section V where numerical results are exhibited. So, we will stay general in the description of the global approach.

1) Recursive solver using Depth-First-Search: The purpose of the recursive solver is to find all possible solutions for a given linear equation system taking into consideration that each variable takes integer values. A pseudo-code of the solver and the introduced variables are presented in Algorithm 1. Let us consider, for instance, the example shown in Fig. 2.

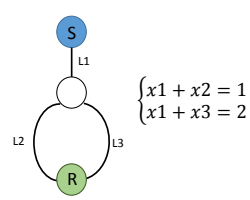


Fig. 2: Probing scheme example

To further simplify things, assume that each variable can take only two values, 0 and 1. Fig. 3 shows the solutions space tree. Each node represents a variable and the edges issued from it are its possible values. See that there are two paths in the example, and that we have $Y = [1, 2]$. The algorithm tries to distribute the end-to-end values among the link metrics

variables using recursive calls. Thus, each recursive call has two parameters: the link index ℓ and the remainder of the end-to-end measurements Y. In a passage by variable X_ℓ corresponding to link ℓ, the link metric value, denoted j in the pseudo-code, is subtracted from all the paths p that include link ℓ ($\ell \in p$, $A(p,\ell) = 1$).

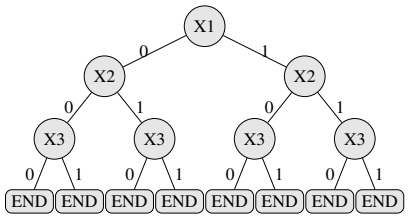


Fig. 3: Solution space example

If the reminder on one path is strictly less than the metric value, the algorithm does not make a recursive call (line 10 in Algorithm 1). This condition avoids exploring the branches of the solution space tree that could not lead to possible solutions. When the "End" node is reached and the remainder Y is equal to 0, the path taken by the successive recursive calls which corresponds to one solution is saved. The path made by the recursive calls is memorized in vector $\mathcal{S}$. In fact, at each recursive call, the value taken by the link metric X_ℓ is added to $\mathcal{S}$ ($\mathcal{S}$.add(j)). At the end of the call, this value is removed ($\mathcal{S}$.delete-last-value).

The first call of the solver is done from the root node X_1: *Solver*$(1, Y = [1, 2])$. In this example, the variable X_1 can take two values, 0 and 1. The algorithm makes two recursive calls:

- *Solver*$(2, Y = [1, 2])$, when X_1 is equal to 0,
- *Solver*$(2, Y = [0, 1])$, When X_1 is equal to 1.

2) Expectation-Maximization (EM) algorithm: The proposed EM algorithm is an adapted version of the EM algorithm described in [9]. The objective is to find the probability distribution α from the probing paths matrix A and the end-to-end observations Y. To obtain an approximation of α, a matrix of latent variables M must be beforehand estimated. The number $M_{\ell,j}$ represents the expected number of times the metric takes value $j\Delta$ on link ℓ. Matrix M has the same dimensions as α.

Expectation step (E-step): The purpose of this step is computing an estimation of the latent variables matrix M from the initial values of α. For a better understanding of the problem formulation, we introduce additional notations described in Table I.

Thus, $M_{\ell,j}$ can be computed using (2):

$$M_{\ell,j} = \sum_{c \in C^\ell} \sum_{\substack{x \in \mathcal{X}^{c,y^c}, \\ x[\ell]=j\Delta}} \frac{\Pr(x)}{\Pr(y^c)} \times N_{c,y^c}, \tag{2}$$

Algorithm 1 Recursive solver

INPUT: linear equation system $AX = Y$.
OUTPUT: $\mathcal{X}$, all possible solutions to the linear system.

1: $\mathcal{X} \leftarrow \emptyset$ ▷ Initialize the solutions set
2: $\mathcal{S} \leftarrow \emptyset$ ▷ Vector to memorize the recursive calls
3: **function** SOLVER(ℓ, Y)
4: **if** $\ell > L$ **then**
5: **if** Y = 0 **then**:
6: $\mathcal{X} \leftarrow \mathcal{S}$ ▷ Save Solution
7: **else**
8: Return
9: **for** `j ∈ range(0,J)` **do**
10: **if** $\forall p \in \mathcal{P}$, with $A[p,\ell] = 1$, $Y[p] \geq j\Delta$ **then**
11: $Y' = Y$
12: **for** $p \in \mathcal{P}$, $A(p,\ell)$ = 1 **do**
13: Y'[p] = Y'[p] - jΔ
14: $\mathcal{S}$.add(jΔ)
15: Solver(ℓ + 1, Y')
16: $\mathcal{S}$.delete-last-value
17: First call of the solver:
18: $Solver(1, Y)$

TABLE I: **List of used parameters**

variable	description
C	The set of clusters
C^ℓ	The subset of clusters where there is at least one path that includes ℓ
y^c	Vector of end-to-end observations for the cluster of paths c
$\mathcal{X}^{c,y^c}$	The solutions to the linear equation system corresponding to c and y^c
N_{c,y^c}	The number of probing experiences where y^c is observed in cluster c

where the probability of each solution vector $x \in \mathcal{X}^{c,y^c}$ is computed with (3),

$$\Pr(x) = \prod_{\ell \in c} \alpha(\ell, x[\ell]/\Delta), \tag{3}$$

and the probability of y^c is the addition of the probabilities of all the possible solutions as described in (4):

$$\Pr(y^c) = \sum_{x \in \mathcal{X}^{c,y^c}} \Pr(x). \tag{4}$$

Maximization step (M-step): the new probability distribution that maximizes the likelihood of the observed data are computed from M as described by (5):

$$\alpha_{\ell,j} = \frac{M_{\ell,j}}{\sum_{j=0}^{J} M_{\ell,j}}. \tag{5}$$

The two steps are repeated until the convergence of α. More details about the used EM algorithm are available in [9].

3) Complexity analysis: The proposal is composed of three main steps: decomposing the global linear equation system (using *k-means*), solving the sub-systems and applying the EM algorithm. We study the complexity of each one separately.

In the first step, the *k-means* algorithm is used to split the linear equation system. The complexity of the algorithm is $O(PL\,k\,n)$, where n is the number of iterations needed by the *k-means* algorithm and k is the number of clusters.

The second step is solving the sub-systems. The split of the linear system into multiple sub-systems enables to reduce the size of the problem by reducing the number of equations and variables. For each sub-system c, the solution space is explored using the recursive solver. The worst case occurs when all the solution space is explored, so, we have $O(J^{L_c})$; in this expression, L_c denotes the number of links in sub-system c. The complexity of this step is $O(J^{L_c}k)$, that is, exponential in L_c. Without dividing the global equation system, the worst case complexity is $O(J^L)$.

Finally, the EM algorithm computes the probability distribution α of the global solution X. The EM algorithm consists of a number of iterations of the E-step and the M-step subroutines. In the E-step, for each subset of paths c, there are $\left|\mathcal{X}^{c,y^c}\right|$ outcomes from the solver. For each outcome we need $O(L_cL)$ multiplications to compute its probability plus additions and divisions to compute the latent variable M. For each subset c, the complexity of the E-step can be approximated by $O(|\mathcal{X}^{c,y^c}|\,L_cL)$. The M-step complexity is $\mathcal{O}(JL)$ which can be ignored in comparison with the E-step. Hence, the complexity of one iteration of the EM algorithm can be approximated by $O(|\mathcal{X}^{c,y^c}|\,L_cL\,k)$.

C. Evolutionary Sampling Algorithm (ESA)

The common idea between [9] and *k-paths* is splitting the linear equation system into multiple sub-systems and solving them. Then, the EM algorithm computes the best global solution. The last two steps (solving the sub-systems and the EM algorithm) have high memory and computation requirements. Thus, these methods are intractable with large topologies or when a fine granularity (a large J) is required.

Hence, we propose the Evolutionary Sampling Algorithm (ESA) which is a stochastic population-based algorithm. Basically, a population is randomly chosen to verify a certain constraint. The goal is to improve the population to find the best solution within the population over iterations.

Thus, instead of solving exactly the linear equation system (or sub-systems), the basic equality is changed into an inequality with an error ε to minimize as shown in inequality (6):

$$||AX - Y|| < \varepsilon. \tag{6}$$

In (6), $||\cdot||$ denotes a standard matrix norm, X is a random variable and α is its probability distribution. As we will see, the objective is to find the best α that minimizes the error ε along the iterations. Distribution α is initialized randomly. The first step of the algorithm is searching the best α that satisfies the inequality. For this purpose, we proceed as follows. Firstly, an important number of solution vector samples (the initial population) are generated according to the probability density α, denoted $\mathcal{X}^{rand}$. Then, the samples that satisfy inequality (6) are selected; $\mathcal{X}^{\text{selected}}$ denotes the selected samples. The new probability distribution α is computed from $\mathcal{X}^{\text{selected}}$. The probability of observing value j on link ℓ, $\alpha(\ell, j)$, is computed by dividing the number of samples where link ℓ takes value $j\Delta$ by the total number of selected samples $|\mathcal{X}^{\text{selected}}|$ as described by (7).

$$\alpha(\ell, j) = \frac{|x \in \mathcal{X}^{selected}, x[\ell] = j\Delta|}{|\mathcal{X}^{selected}|}. \tag{7}$$

This process is repeated until the convergence of α or until reaching a maximum fixed number of iterations. We consider that the probability density α has converged if some criteria comparing previous and new α is satisfied, as in many iterative processes. We don't discuss this kind of details in this text.

The second step is to tune the error according to the result of the first step. If the probability density has converged, the tolerated error ε is reduced and the process starts again, to try a more ambitious target. Otherwise, we iterate the process with a larger error. To minimize the error, ε is multiplied by an adaptation coefficient denoted β in the pseudo-code, taking values between 0 and 1. This coefficient is set up at the initialization step before running the algorithm. The previous value of ε is memorized in a variable noted ε' at each adaptation. If α does not converge in the first step, ε is initialized again with ε'. Then, β is increased using (8):

$$\beta = \beta + \frac{1-\beta}{2} = \frac{1+\beta}{2}. \tag{8}$$

The two steps are repeated until ε stops changing (that is, until β gets very close to 1. This process is globally described in Algorithm 2 and Fig. 4.

Some comments on the beginning of the algorithm, when we initialize some variables. Concerning α, we can sample its values from some distribution, or we can assume the Uniform one: $\alpha(\ell, j) = (J+1)^{-1}$. Concerning ε, we first normalize X and Y by dividing them by B, and we take ε rather large (for instance, $\varepsilon = 1/2$). The coefficient β is chosen not close to 1, typically $\beta = 1/2$ as well.

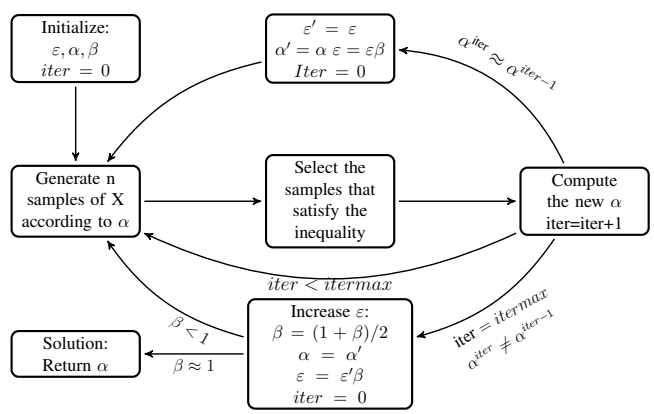


Fig. 4: ESA algorithm

Algorithm 2 Evolutionary Sampling Algorithm (ESA)

INPUT: Linear equation system $AX = Y$
OUTPUT: α

Initialize α (α^0), ε and β
iter = 0 ▷ iteration index
— Step 1:
while iter $\leq$ itermax **do**
  iter = iter +1
  Generate n random samples of solutions according to distribution α.
  Select the ones that respect $||AX - Y|| < \varepsilon$; put them in set $\mathcal{X}^{\text{selected}}$.
  Compute the new α (α^{iter}) from the selected samples:
    $\alpha^{\text{iter}}(\ell, j) = \frac{|x \in \mathcal{X}^{\text{selected}},\, x[\ell]=j|}{|\mathcal{X}^{\text{selected}}|}$
  if $\alpha^{iter} \approx \alpha^{iter-1}$ **then**
    Break ▷ α has converged
— Step 2:
if α has converged **then**
  $\varepsilon' = \varepsilon$
  $\alpha' = \alpha$
  $\varepsilon = \varepsilon\ \beta$
  iter = 0
  Repeat from Step 1
else
  $\beta = \beta + (1 - \beta)/2$
  $\alpha = \alpha'$
  if $\beta \approx 1$ **then**
    Return α ▷ ε has converged
  else
    $\varepsilon = \varepsilon'\beta$
    iter = 0
    Repeat from Step 1

1) Complexity analysis: The ESA algorithm is composed of two parts denoted Step 1 and Step 2. The complexity of the first step has the strongest weight. It is mainly determined by the number of random solutions $|\mathcal{X}^{\text{rand}}|$. The most costly operation is the multiplication of the matrix of random solutions $\mathcal{X}^{\text{rand}}$ by matrix A. The complexity of this operation is $O(|\mathcal{X}^{\text{rand}}|LP)$. The complexity of the other operations like the generation of $\mathcal{X}^{\text{rand}}$ and computing the new value of α with (7) depend on $|\mathcal{X}^{\text{rand}}|$ and the granularity J. Their complexity can be approximated by $O(|\mathcal{X}^{\text{rand}}|J)$. These operations are repeated until the convergence of α. Let n_1 denote the number of iterations for the convergence of α. The two steps are repeated until the error bound ε is considered as having converged; let us denote by n_2 the number of global iterations needed. Both n_1 and n_2 are mainly influenced by the initialized parameters like the selectivity criteria ε and convergence conditions of α and β. The global complexity of the ESA can be approximated by

$$O\Big(|\mathcal{X}^{\text{rand}}|(LP + J)n_1 n_2\Big).$$

V. Performance Analysis

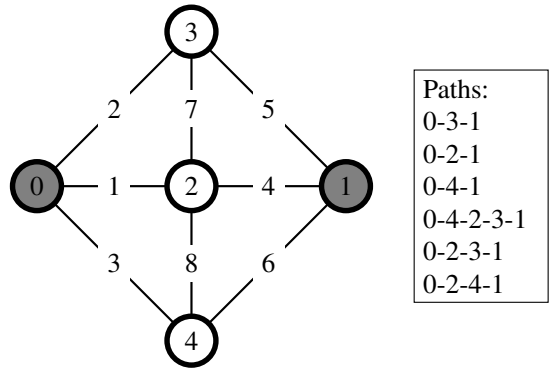


Fig. 5: Topology-A

The output of the two algorithms we proposed in this paper, *k-paths* and ESA, is the potential distribution α of the unknown link metric X. This information can be useful for multiple use cases in network monitoring. One possible application is the troubleshooting of network failure using end-to-end measurement. In fact, it is possible to compute from distribution α the probability that the metric value on a link ℓ exceeds a fixed threshold $V_{\text{fail}} \in [0, B]$, $V_{\text{fail}} = j_{\text{fail}}\Delta$, using (9):

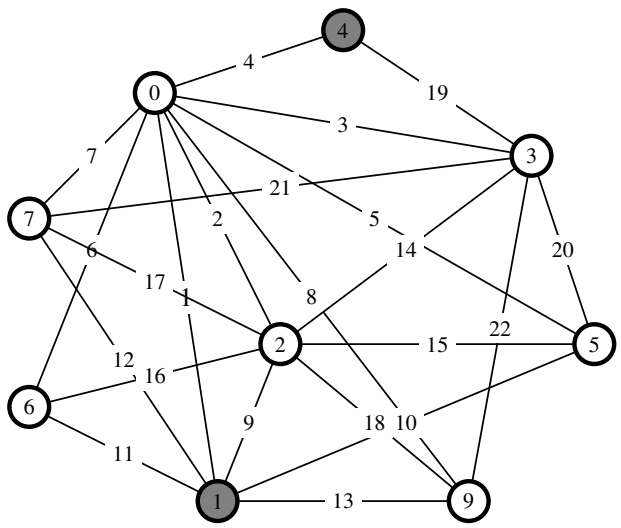
$$\Pr(X_\ell > V_{\text{fail}}) = \sum_{j=j_{\text{fail}}+1}^{J} \alpha_{\ell,j}. \tag{9}$$

Then, an alarm can be activated if the failure probability on a link exceeds a fixed threshold, say, if $\Pr(X_\ell > V_{\text{fail}}) > p^*$. Thus our solutions can be extended to a binary classification task for a link failure detection. The two thresholds V_{fail} and p^* can be tuned to have a good rate of true positive and true negative in the binary classification task. This can be used for the initial troubleshooting of the network, in order to detect the potential failure root causes using only end-to-end measurements. Then, more sophisticated monitoring protocols can be applied to remote additional information about the potential failed links or equipment. Coming back to the paper, we preferred to illustrate our method with the simpler case of just approximating the value of the metric, but this dependability application has been already explored and will be described in an extended version of this text.

To evaluate the performances of the proposed solutions, we experiment on two topologies taken from [16]. The first one has five nodes and eight edges. The second graph has nine nodes and twenty-two edges. With each topology we associate a dataset of multiple samples of delay measurements performed on the different links. The traffic was simulated with the `Omnet++4` tool. For each topology, we consider a predefined list of paths and we compute the end-to-end delay on each path by the addition of the delays on the paths' links. The algorithms take as input the end-to-end delays and

evaluate the estimated link metrics. Then, the error is evaluated using (10).

$$\text{Error (in \%)} = 100\frac{|V^{\text{estimated}} - V^{\text{real}}|}{B}. \quad (10)$$



Paths:			
4-3-7-0-5-1	4-0-2-6-1	4-3-7-2-1	4-0-5-2-1
4-0-2-9-1	4-0-2-7-1	4-0-3-2-1	4-3-9-2-1
4-3-7-2-1	4-0-9-1	4-0-2-5-1	4-0-6-2-1
4-0-7-2-1	4-3-5-1	4-3-0-1	4-0-2-1
4-3-7-1	4-0-6-1	4-3-9-1	4-0-7-1
4-3-2-1	4-0-5-1		

Fig. 6: Topology-B

The error and the computing time are measured according to the granularity J and the number of paths P. For a fixed number of paths, the first ones from the global list are selected. For each experience, we run 50 different tests.

A. Topology A

Firstly, we study the impact of the number of paths on the two algorithms. The *k-paths* solution is tested with granularity J equal to 10 (K-10) and the ESA with two granularities 10 and 50 (ESA-10 and ESA-50).

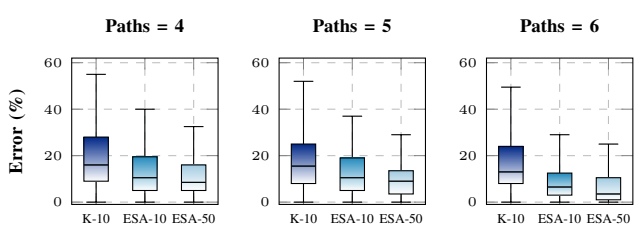


Fig. 7: Topology-A: Error-vs-Paths

Fig.7 illustrates the dispersion of the percentage error respectively of K-10, ESA-10 and ESA-50, while Fig.8 illustrates the variation of the corresponding mean computing time.

For the same granularity 10, ESA-10 shows better performances than K-10. For example, with four paths, the median errors for the two algorithms are 16% and 11% respectively. The variance of the error is also enhanced with ESA-10 since the error takes values between 0 and 40% with ESA-10 and between 0 and 55% with K-10. This enhancement with ESA can be explained by the fact that it is more robust to local optima compared to *k-paths*. In fact, ESA starts by exploring the solution space with random sampling and enhances the accuracy over the successive iterations, while *k-paths* starts from one point and tries to maximize the likelihood of the observed data.

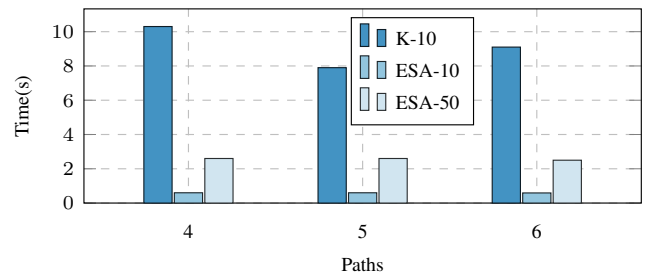


Fig. 8: Topology-A: Time-vs-Paths

ESA-10 shows also better performances in term of computing time. In the experience with four paths, the mean computing time with K-10 is more than 10 seconds while it is less than 1 second with ESA-10. Increasing the granularity J to 50 with ESA-50 enhances the accuracy of the estimation and increases the computing time compared to ESA-10. K-10 still have the largest computing time compared to ESA-10 and ESA-50. Increasing the number of paths enhances the

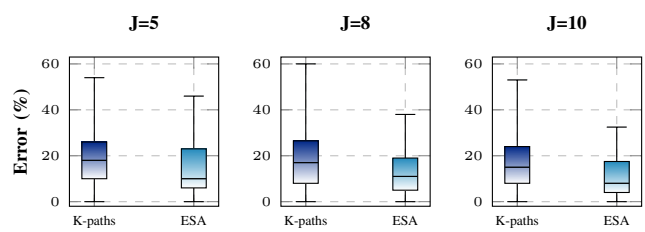


Fig. 9: Topology-A: Error-vs-J

accuracy of the estimations for the three experiences. This is a trivial and expected result since each path provides an additional equation for the linear system that can help in the problem resolution. Increasing the number of paths has a low impact on the computation time. In fact, increasing the number of equations in the linear system has a linear impact on the computation complexity of the worst case as explained in sections IV-B3 and IV-C1. However, providing additional equations can help the algorithms to converge faster which can explain some variation of computing time values in Fig.8.

Secondly, we study the impact of the granularity J. Fig.9 shows the variation of the percentage error of the two algorithms according to J. Increasing the granularity J has

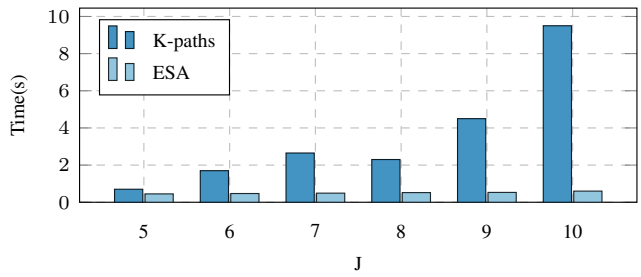


Fig. 10: Topology-A: Time-vs-J

an important effect on the enhancement of the accuracy of the two algorithms, especially ESA. Fig. 9 shows that by increasing J from 5 to 10, the maximum error value decreases from 47% to 34% for ESA. However the maximum error decreases only from 55% to 52% for the *k-paths* solution. Fig.10 shows the computing time variations according to J. The experimental results are in conformity with the complexity studied in sections IV-B3 and IV-C1. In fact, the computing time increases exponentially with the *k-paths* and linearly with ESA.

B. Topology B

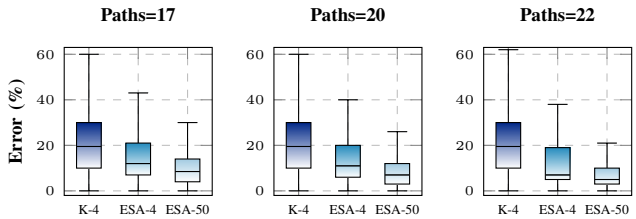


Fig. 11: Topology-B: Error-vs-Paths

We make similar tests with a larger topology described in Fig. 6. Fig. 11 illustrates the variation of the percentage error according to the number of paths. The *k-paths* is tested with a granularity J equal to 4 (K-4) and ESA with two different values of J, 4 (ESA-4) and 50 (ESA-50). Fig. 12 shows the mean computing time of these tests.

For $J = 4$, ESA-4 has always the best results in terms of accuracy and computing time compared to K-4. At one point, adding additional paths does not enhance the estimations. In fact, there is a sampling error due to the discretization of the solution space that can be avoided only by choosing a finer granularity. With the EM-based algorithm, a higher granularity means that the computing time increases exponentially. ESA is then tested with granularity 50 (ESA-50). The results show that the percentage error of ESA-50 is significantly enhanced by increasing the number of probing paths. In fact, the maximum error decreases from $35\,\%$ to $21\,\%$ and the median decreases from $10\,\%$ to $5\,\%$ with ESA-10. However, we do not have the same enhancement with ESA-4 and K-4.

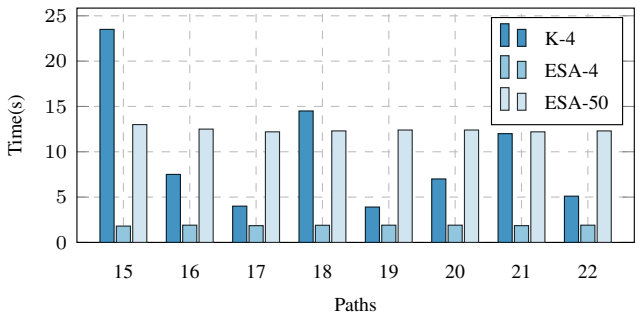


Fig. 12: Topology-B: Time-vs-Paths

VI. Conclusions

In this paper, we focus on the additive metrics inference in general network topologies with unicast probing. One of the main specifications in our work is considering the probing scheme as a set of unicast probing paths. This feature gives an abstract view to multiple topologies and gives more flexibility in the monitoring strategy deployment. The use of simple unicast probing paths highlights other interesting issues to study related to the collection points placement and probing schemes configuration in the context of statistical approaches. The *k-paths* solution extends the applicability of an existing solution for tree topologies. The ESA algorithm enhances the performances of the approach since it gives the best results on the studied topologies. Testing it with larger topologies requires studying other parameters and strategies related to the collection points and the probing schemes deployment, topics belonging to the scope of our future work.

We may observe that the procedures described here, and specially the ESA technique, have a number of parameters whose impact on the algorithm performance is clearly an interesting issue. The analysis of these aspects will be the object of next immediate work.

REFERENCES

[1] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *Journal of Network and Computer Applications*, vol. 103, 12 2017.

[2] J. D. Case, M. Fedor, M. L. Schoffstall, and J. Davin, "Simple network management protocol (SNMP)," RFC 3176, 1990.

[3] S. Panchen, N. McKee, and P. Phaal, "InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks," RFC 3176, Sep. 2001.

[4] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: Recent developments," *Statistical Science*, vol. 19, no. 3, pp. 499–517, 2004.

[5] E. Lawrence, G. Michailidis, V. N. Nair, and B. Xi, "Network tomography: A review and recent developments," in *In Fan and Koul, editors, Frontiers in Statistics*. College Press, 2006, pp. 345–364.

[6] R. Cáceres, N. G. Duffield, J. Horowitz, and D. F. Towsley, "Multicast-based inference of network-internal loss characteristics," *IEEE Trans. Information Theory*, vol. 45, no. 7, pp. 2462–2480, 1999.

[7] D. Z. Tootaghaj, T. He, and T. La Porta, "Parsimonious tomography: Optimizing cost-identifiability trade-off for probing-based network monitoring," *SIGMETRICS Perform. Eval. Rev.*, vol. 45, no. 3, pp. 43–55, Mar. 2018.

[8] T. He, A. Gkelias, L. Ma, K. K. Leung, A. Swami, and D. Towsley, "Robust and efficient monitor placement for network tomography in dynamic networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 3, pp. 1732–1745, June 2017.

[9] E. Lawrence, G. Michailidis, and V. N. Nair, "Network delay tomography using flexicast experiments," *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, vol. 68, no. 5, pp. 785–813, 2006.

[10] W. Zhu, "Loss tomography in general topology," *arXiv e-prints*, vol. 1603.07825, 2016.

[11] K. Deng, Y. Li, W. Zhu, and J. Liu, "Fast parameter estimation in loss tomography for networks of general topology," *The Annals of Applied Statistics*, vol. 10, 10 2015.

[12] V. N. Padmanabhan, L. Qiu, and H. J. Wang, "Server-based inference of internet link lossiness," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 1, 2003, pp. 145–155 vol.1.

[13] K. Deng, Y. Li, W. Zhu, Z. Geng, and J. S. Liu, "On delay tomography: Fast algorithms and spatially dependent models," *IEEE Trans. Signal Processing*, vol. 60, no. 11, pp. 5685–5697, 2012.

[14] N. Duffield, "Simple network performance tomography," in *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '03, 2003, pp. 210–215.

[15] C. Liu, T. He, A. Swami, D. Towsley, T. Salonidis, A. I. Bejan, and P. Yu, "Multicast vs. unicast for loss tomography on tree topologies," in *MILCOM 2015 - 2015 IEEE Military Communications Conference*, 2015, pp. 312–317.

[16] A. Mestres, A. Rodríguez-Natal, J. Carner, P. Barlet-Ros, E. Alarcón, M. Solé, V. Muntés-Mulero, D. Meyer, S. Barkai, M. J. Hibbett, G. Estrada, K. Maruf, F. Coras, V. Ermagan, H. Latapie, C. Cassar, J. D. Evans, F. Maino, J. C. Walrand, and A. Cabellos-Aparicio, "Knowledge-defined networking," *Computer Communication Review*, vol. 47, pp. 2–10, 2017.