Scalable Monitoring Heuristics for Improving Network Latency

Maxime Mouchet, Martin Randall, Marine Ségneré, Isabel Amigo, Pablo Belzarena, Olivier Brun, Balakrishna Prabhu, Sandrine Vaton *

December 16, 2019

Abstract

We consider a routing overlay in which the delay of a path can be obtained at some fixed cost by sending probe packets, and investigate the joint minimization of the probing cost and the routing delay. Assuming that link delays are modelled by Markov chains, this problem can be cast as a Markov Decision Process (MDP). Unfortunately, computing the exact solution of this MDP is prohibitively expensive due to the well-known "curse of dimensionality". In this work we propose two scalable approaches that are fast enough to provide efficient solutions on practical time scales. We analyze the complexity of both approaches, and evaluate their accuracy in small synthetic scenarios for which the optimal monitoring policy can be computed. Finally, the robustness and the scalability of the proposed solutions are analyzed using real delay data collected over the Internet.

1 Introduction

Accurate and fine-grained network monitoring makes it possible to establish a real-time map of the state of the network, and thus to quickly react to adverse events or to make optimal resource-allocation decisions. Nevertheless, traditional network monitoring techniques can generate a significant amount of traffic thereby degrading the overall network performance. This is particularly true when active network probing techniques are used, and when the number of resources and metrics to handle is large. In many situations, there is a trade-off between the cost of monitoring resources, which brings more accurate state information, and the quality of resource-allocation decisions.

*M.Mouchet, I. Amigo and S. Vaton are with IMT-Atlantique, Brest, France. M. Randall and P. Belzarena are with Instituto de Ingeniería Eléctrica, Universidad de la República, Uruguay. O. Brun, B. Prabhu and M. Ségneré are with LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France.

For instance, such a situation occurs with routing overlays deployed over the Internet [1, 2]. In a routing overlay, the nodes actively probe the quality of the Internet routes between them and each node can act as a relay for the messages exchanged between other nodes. Hence, even if the underlying IP mechanisms for computing routes are unchanged, the overlay can influence the overall path taken by packets by adding intermediate routing hops. If it happens that the direct Internet path between a source node and a destination node of the overlay is unavailable or has an unacceptable performance, it becomes possible to transfer data via an alternate path in the complete graph formed by all overlay nodes. However, active monitoring of the quality of Internet paths by sending probe packets is costly. As the number of overlay links grows as $O(n^2)$ with the number n of overlay nodes, probing all overlay links quickly become overwhelming [3]. It was reported in [4] that a reasonable overlay can support only about 50 routers (see also [5, 6, 7]).

Another situation in which the minimization of the active probing traffic might be desirable is for the dynamic load-balancing of application flows over multiple WAN connections in software-defined wide area networks (SD-WAN). Consider for instance two or more satellite access links operating over distinct frequencies and assume that we need to dynamically select the link with the greater available data rate. The performance of each access link may vary drastically over time (e.g., due to satellite jamming) and therefore it has to be monitored over time. However, due to the scarcity of radiofrequency resources, excessive probe packets generation is susceptible to produce a significant load which can disturb the operation of the network. Instead of probing all access links at all time, it makes sense to use a parsimonious strategy trading-off accuracy against monitoring effort.

In this paper, we focus on routing problems and study the trade-off between the monitoring cost and the optimality of routing decisions. The quality of routing decisions can of course be improved by monitoring paths more often, but at an increased monitoring cost. Our goal is to devise parsimonious monitoring methods allowing to drastically reduce the monitoring overhead but yet sufficiently accurate to enable near-optimal routing decisions. The key idea is that, provided the quality of paths is stable over sufficiently long periods of time, it is not necessary to observe everything at all times in order to make good routing decisions. The proposed methods decide at each measurement epoch which of the paths should be monitored, if any, instead of systematically probing all paths. To start with, and because large scale measurement campaigns are publicly available for validations, we consider as paths quality of service (QoS) metrics the round trip time (RTT) delay.

1.1 Related work

The design of parsimonious monitoring strategies was discussed a few years ago in the context of overlay networks in [8, 9]. The authors propose learning-based solutions and experimentally show that near-optimal performances can be obtained with a significantly reduced monitoring effort. These works assume, however, that a given number of overlay paths has to be measured at each measurement epoch, whereas we seek for an optimal trade-off between monitoring cost and network latency.

More recently, in [10] we introduce a theoretical framework for taking monitoring and routing decisions that offer a good trade-off between monitoring costs and the gain in delay from appropriate routing decisions. This framework is based on Markov Decision Processes (MDPs) and a markovian modeling of path delays. In addition, we demonstrate the pertinence of performing such parsimonious monitoring strategies in the routing overlay use-case by simulating a 30-node overlay based on real RTT data obtained form the RIPE Atlas Internet measurement network [11]. Simulations results show that even reducing drastically the monitoring load one can obtain a routing performance almost as good as that one obtained when all paths are monitored permanently. However, as pointed out, the exact computation of the optimal monitoring policy is only feasible for small-scale examples. In the present paper, we adopt the theoretical framework introduced in [10] and propose scalable monitoring methods which can be used for larger scenarios.

It is also relevant to mention the work on network tomography techniques [12, 13, 14] for inferring the state of network elements from end-to-end measurements made by a small number of probing nodes (beacons). Some authors focused on the minimization of the number of beacons [15, 16, 17], whereas others focused on the selection of the end-to-end paths to be measured in order to minimize the monitoring overhead [18, 19, 20]. Despite some similarities, the above works are different from ours in that they focus on identifying the performance of each individual link with as few end-to-end probes as possible, whereas in our case the monitoring decisions are driven by the perspective of lower routing costs.

Similarly, an approach based on Markovian modeling of network path performance and Markov decision processes has been used to determine optimal flow assignment policies for devices with multiple network access [21]. But this work does not consider the problem of parsimonious measurement.

1.2 Contributions

In [10], the trade-off between accurate latency information and monitoring costs was investigated in the framework of an MDP. However, computing the exact solution of this MDP is prohibitively expensive and not practical when the number of paths for each OD pair is not very small.

We propose two different heuristics which are fast enough to be used for online decision-making, and yet often provide close-to-optimal monitoring policies. The first heuristic is inspired from the Receding Horizon Control [22], for MDPs with large state space. The second heuristic determines whether each path individually should be monitored or not by solving an MDP with a reduced state space in which all other paths are represented by a single virtual deterministic path.

The performance of these heuristics is evaluated and compared against those of the optimal policy and the myopic policy (that is, a greedy policy minimizing the expected immediate cost) on various scenarios constructed either artificially or from real traces collected over the Internet. It is observed that these two heuristics work reasonably across these scenarios whereas the myopic policy can have a widely variable performance. For real traces, it is shown that the heuristics track closely the minimum delay path at a low monitoring overhead and provide a significant improvement over the direct IP path, while having computing times that are compatible with a real-time implementation.

1.3 Organization of the paper

The paper is organized as follows. In Section 2 we describe our assumptions as well as the formulation of the problem as an MDP. We then present our heuristic monitoring schemes in Sections 3 and 4. Numerical validation results are presented in Section 5. Finally, we give some concluding remarks in Section 6.

2 MDP Formulation

We consider a decision-maker that has to send packets from a source node to a destination node. The decision-maker has the choice between P possible paths and its goal is to choose paths so as to minimize the total delay incurred by packets.

Each path can be in multiple states, and the value of the delay over the path is different in each state. The issue is that, although it knows the statistical models governing state transitions of the paths, the decision-maker does not directly observe the delay over a path, even when a packet is sent over it. It is therefore not possible for the decision-maker to infer information about the state of the paths from packet transmissions. The decision-maker can, however, decide to monitor at some cost the quality of certain paths, before sending a packet. An issue is therefore trading off the cost of monitoring paths, which brings more accurate state information, with the higher probability of experiencing high transmission delays.

2.1 Assumptions and notations

We assume that time is slotted and that packet transmissions occur in each time slot. The P possible paths are numbered from 1 to P and $\mathcal{P} = \{1, 2, \ldots, P\}$ denotes the set of paths.

The state $X_i(t)$ of each path i evolves randomly at each time slot over state space $\mathcal{S}_i$ according to a Discrete Time Markov Chain (DTMC) with known transition probability matrix P_i. The transmission delay of path i at time t is a random variable $L_i(t)$ which depends on the state of the path: when the path is in state $x \in \mathcal{S}_i$, $L_i(t) = \ell_{i,x}$. We denote by $\boldsymbol{\ell}_i$ the vector $\boldsymbol{\ell}_i = (\ell_{i,1}, \ell_{i,2}, \ldots)$. We also define the product space $\mathcal{S} = \bigotimes_{i \in \mathcal{P}} \mathcal{S}_i$.

2.2 Decision problem

At each time step, the decision-maker first chooses a vector $\mathbf{u}(t) \in \{0,1\}^P$ and observes the state of the paths i such that $u_i(t) = 1$. The decision-maker has to pay a fee c_i to observe the actual state $X_i(t)$ of path i, so that the total monitoring cost associated to the monitoring action $\mathbf{u}(t)$ is $\sum_{i:u_i(t)=1} c_i = \mathbf{c} \cdot \mathbf{u}(t)$, where $\mathbf{c} = (c_1, \ldots, c_P)$. However this information can be used by the decision-maker not only to make its immediate routing decision, but also to gain some knowledge on the state of paths for its future decisions.

Once the observation is made, the decision-maker takes its routing decision. If the decision-maker chooses path i, then it incurs a routing cost $L_i(t)$. We let $r(t)$ be the path chosen at time t by the decision-maker, possibly taking into account the information brought by all past observations.

In summary, we consider the following setting. At each round $t = 1, 2, \ldots$:

(1) the state $X_i(t)$ of each path i evolves according to a DTMC with transition probability matrix P_i,

(2) the decision-maker chooses a vector $\mathbf{u}(t) \in \{0,1\}^P$, pays the monitoring cost $\mathbf{c} \cdot \mathbf{u}(t)$ and observes the state $X_i(t)$ of each path i such that $u_i(t) = 1$,

(3) the decision-maker sends a packet over path $r(t)$ and incurs a routing cost $L_{r(t)}(t)$.

The objective of the decision-maker is to make a trade-off between the cost of monitoring paths, which brings more up-to-date state information, and the expected delay (the expected routing cost). Obviously, the optimal routing decision is to choose the path $r(t)$ with the minimum expected delay. However, the expected delay of a path depends on the belief of the decision-maker on the state of that path, since the state is exactly known only for

those paths that have been observed. Formally, the goal of the decision-maker is to take its monitoring and routing decisions so as to minimise

$$\mathbb{E}\left\{\sum_{t=0}^{\infty}\rho^t\left[L_{r(t)}(t)+\mathbf{c}\cdot\mathbf{u}(t)\right]\right\}, \tag{1}$$

where $\rho\in(0,1)$ is a given positive discount factor.

2.3 Optimal tradeoff as the solution of an MDP

The problem as described above can be cast as an MDP, as we now explain [10]. Defining $y_i(t)$ as the last observed state of path i at time t (just before measurement) and $\tau_i(t)\geq 1$ as the age of this observation, all the information available to the decision-maker at time t is summarized by the vector $\mathbf{s}(t)=(s_1(t),s_2(t),\ldots,s_P(t))$, where $s_i(t)=(y_i(t),\tau_i(t))$ because of the Markov property satisfied by each path i. We shall refer to the vector $\mathbf{s}(t)$ as the information state or belief state just before measurement at time t.

Given an information state $\mathbf{s}$, the decision-maker can compute for each path i the vector $\mathbf{b}_i(\mathbf{s}_i)=\mathbf{e}_{y_i}P_i^{\tau_i}$, where $\mathbf{e}_i$ is a vector whose components consist of one 1 in position i and otherwise 0s. The vector $\mathbf{b}_i(\mathbf{s}_i)$ corresponds to a probability distribution on $\mathcal{S}_i$ representing the decision-maker's current belief on the state of path i. $b_{i,x}(\mathbf{s}_i)=\mathbb{P}(X_i=x)$ represents the probability, as estimated by the decision-maker from past measurements, that path i is in state $x\in\mathcal{S}_i$ at time t before it takes its monitoring decision.

Note that the information state $\mathbf{s}(t)$ has the Markov property: the next information state $\mathbf{s}(t+1)$ depends only on the current information state and the monitoring action $\mathbf{u}(t)$ taken, but not on past information states. It follows that the information state $\mathbf{s}(t)$ evolves according to a controlled Markov chain with transition probabilities

$$\pi(\mathbf{s},\mathbf{s}';\mathbf{u})=\prod_{i=1}^{P}\pi_i(s_i,s_i';u_i), \tag{2}$$

where, for any two states $s_i=(y_i,\tau_i)$ and $s_i'=(y_i',\tau_i')$ in $\mathcal{S}_i\times\{1,2,\ldots\}$

$$\pi_i(s_i,s_i';u_i)=\begin{cases}1 & \text{if } u_i=0,\ s_i'=(y_i,\tau_i+1)\\ b_{i,x}(s_i) & \text{if } u_i=1 \text{ and } s_i'=(x,1)\\ 0 & \text{otherwise.}\end{cases} \tag{3}$$

The decision-maker's optimal routing decision is to transmit the message over the path with the minimum expected transmission delay. Whereas the information state $\mathbf{s}(t)$ is the belief of the decision-maker on the states of the different paths just before measurement, the routing decision $r(t)$ is taken

just after measurement. Conditioning on the information available to the decision-maker just after measurement, the minimum expected delay is

$$D(\mathbf{s}, \mathbf{x}; \mathbf{u}) = \min_i \left(u_i \ell_{i,x_i} + (1 - u_i) \sum_{y \in \mathcal{S}_i} b_{i,y}(s_i) \ell_{i,y} \right) \tag{4}$$

where $\mathbf{x} \in \mathcal{S}$.

In other words, once it has observed the states of the paths i for which $u_i = 1$, the decision-maker chooses to send its message over the path with the minimum expected delay. This can be the monitored path with the minimum observed delay, or a path which has not been monitored but which has a lower expected delay given the belief on its state. It follows that the expected cost of monitoring decision $\mathbf{u}$ in information state $\mathbf{s}$ is

$$\bar{D}(\mathbf{s}; \mathbf{u}) = \mathbf{c} \cdot \mathbf{u} + \sum_{\mathbf{x} \in \mathcal{S}} \left(\prod_{i=1}^{P} b_{i,x_i}(s_i) \right) D(\mathbf{s}, \mathbf{x}; \mathbf{u}). \tag{5}$$

2.4 Value function

Given a stationary monitoring policy μ associating to every possible information state $\mathbf{s}$ a vector $\mu(\mathbf{s}) \in \{0, 1\}^P$, which represents the monitoring decision $(u_1, \ldots, u_P)$ in state $\mathbf{s}$, we define the value of state $\mathbf{s}$ under this policy as

$$V_\mu(\mathbf{s}) = \mathbb{E}_\mu \left\{ \sum_{t=0}^{\infty} \rho^t \bar{D}(\mathbf{s}(t); \mu(\mathbf{s}(t))) \mid \mathbf{s}(0) = \mathbf{s} \right\}. \tag{6}$$

Let $V^*(\mathbf{s}) = \min_\mu V_\mu(\mathbf{s})$ be the value of information state $\mathbf{s}$ under the optimal policy. The dynamic programming equation, or Bellman equation, is then

$$V^*(\mathbf{s}) = \min_{\mathbf{u}} \left[\bar{D}(\mathbf{s}; \mathbf{u}) + \rho \sum_{\mathbf{s}'} \pi\left(\mathbf{s}, \mathbf{s}'; \mathbf{u}\right) V^*(\mathbf{s}') \right], \tag{7}$$

and the optimal action in state $\mathbf{s}$ corresponds to the vector $\mathbf{u}$ achieving the minimum in (7).

A classical approach to solve the Bellman equation is to use the value iteration algorithm

$$V_{n+1}(\mathbf{s}) = \min_{\mathbf{u}} \left[\bar{D}(\mathbf{s}; \mathbf{u}) + \rho \sum_{\mathbf{s}'} \pi\left(\mathbf{s}, \mathbf{s}'; \mathbf{u}\right) V_n(\mathbf{s}') \right], \tag{8}$$

which converges at a geometric rate to the optimal value function V^*. Other approaches, such as the policy iteration algorithm, also exist [23].

But the numerical complexity of the value iteration algorithm (8) becomes very large when the number of considered paths P is not extremely small. Indeed the discrete information state is $\mathbf{s} = (s_1, s_2, \ldots, s_P)$, where $s_i = (y_i, \tau_i)$. In order to have a finite state space one can assume that $\tau_i \leq \tau_{\max}$ where $\tau_{\max}$ is a large enough upper bound on the time between two measurements. If we assume that each path has K possible states, that is, $|\mathcal{S}_i| = K$ for all $i \in \mathcal{P}$, then the total number of information states $\mathbf{s}$ is $S = (\tau_{max}\, K)^P$, whereas the total number of actions is $A = 2^P$. As the complexity of the value iteration algorithm (8) is proportional to S and A, computing the optimal policy quickly becomes prohibitively expensive.

3 Heuristic 1: A Receding Horizon approach

In this section, we consider an approximation scheme for solving the MDP formulation. This approach creates a stationary policy using an approximated solution with a fixed finite-horizon of the given infinite-horizon MDP. Let us recall that a finite-horizon MDP is a problem in which the sum which appears in Equation (1) is limited to a finite number of terms.

3.1 Initial formulation

First, it is important to remark some properties of this type of approximation. In order to approximate an infinite horizon MDP, one possible way is to use a finite-horizon optimization (which is different from the receding horizon approximation introduced in this section). The finite-horizon approximation leads to a control sequence $\mu(0), \mu(1), \ldots, \mu(H-1)$ which begins at the current time t and ends at some future time $t + H - 1$, where H is the horizon over which we are optimizing. This fixed horizon approximation suffers from an important drawback: the policy obtained is not stationary and so we have different optimal actions for the same state at different decision times.

The above issue is addressed by the receding horizon approach [22], which can be explained as follows:

1. At time t and for the current state $\mathbf{s}$, solve the MDP problem over a fixed future horizon $\{t, t + H - 1\}$.
2. Apply only the first action of the resulting optimal policy.
3. Determine the state reached at time $t + 1$.
4. Repeat step 1 at time $t + 1$ over the interval $\{t + 1, t + H\}$.

The policy obtained has a receding horizon interpretation: the decision optimizes over the current cost plus the expected cost of looking H steps ahead. This decision is applied recursively; the future is taken into account,

but at a limited level of complexity since the number of states considered is considerably reduced.

The receding horizon approximation is implemented as follows. We use the initial steps of the value iteration algorithm (see Equation (8)) to approximate the optimal policy. Starting from $V^0(\mathbf{s}) \equiv 0$, for the current state $\mathbf{s}$ we have

$$V^1(\mathbf{s}) = \min_{\mathbf{u}} \left[\bar{D}(\mathbf{s}; \mathbf{u}) + \rho \sum_{\mathbf{s}'} \pi \left(\mathbf{s}, \mathbf{s}'; \mathbf{u}\right) 0 \right]. \tag{9}$$

This first step gives the *myopic policy* minimizing the expected immediate cost and disregarding future costs. In some particular cases, the myopic policy may already provide satisfactory results.

To improve on it, we apply h steps of the value iteration algorithm by iterating over

$$V^h(\mathbf{s}) = \min_{\mathbf{u}} \left[\bar{D}(\mathbf{s}; \mathbf{u}) + \rho \sum_{\mathbf{s}'} \pi \left(\mathbf{s}, \mathbf{s}'; \mathbf{u}\right) V^{h-1}(\mathbf{s}') \right], \tag{10}$$

for $h = 1, 2, \ldots, H$, and the optimal policy $\mu^h(\mathbf{s})$ in information state $\mathbf{s}$ corresponds to the vector $\mathbf{u}$ achieving the minimum in (10).

Therefore, in order to find $V^H(\mathbf{s})$ we build a tree of all possibles states visited within H steps starting at $\mathbf{s}$. That is, for each action $\mathbf{u}$ we first find the set of all possible states $\mathbf{s}'$ from $\mathbf{s}$. Then, for each $\mathbf{s}'$ and for each action $\mathbf{u}$ we find the set of possible next states $\mathbf{s}''$, and so on, H times. This is represented in Figure 1, where for illustration purposes we have taken the case of two paths ($P = 2$), K states per path and $H = 2$.

Once the tree is built, we are able to go H steps forward from $\mathbf{s}$. As described in Algorithm 1, we start applying the Bellman equation, first for obtaining V^1, then for V^2, until we find V^H, V^0 being initialized to 0. This recursive algorithm returns at each step the value function V^h and the optimal policy $\mathbf{u}^h$, but the latter is only used at step H to apply action $\mathbf{u}^h(\mathbf{s})$ in the current state $\mathbf{s}$.

3.2 Complexity analysis

The problem with this first receding horizon algorithm is that each information state (white nodes in Figure 1) has $(1 + K)^P$ successors in the tree. Then, for going up to horizon H we would have $((1 + K)^P)^H$ information states to take into account.

Fortunately, looking at the structure of the information state space of our problem we can see that at each depth of the tree there are many duplicated information states.

More precisely, to see which information states are present at depth H without duplicates, it is sufficient to find, for each path i, the possible values

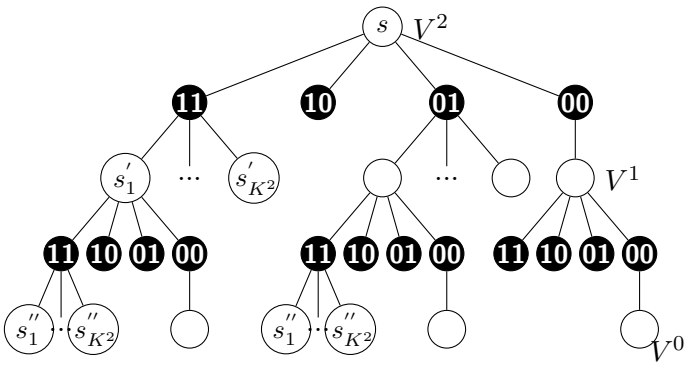

Figure 1: An illustration of the information states (white nodes) visited when executing the receding horizon approximation with $P = 2$ paths with K states each and an horizon $H = 2$. Black nodes represent the possible actions in each state. We start building the tree at depth 0 (the root) and finish at depth H. We then calculate V^0 at depth H, and go up to calculate V^H at the current information state (root of the tree).

the associated coordinate of the information state s_i can take, and combine this information for all paths.

Let's consider one path, say path i at depth H, and let's see the possible values that s_i^H can take. We note that for quantifying the different values s_i^H can take it does not matter how many times path i has been measured, but it does matter when was the last depth at which it was measured, for a new measure resets the lag τ, and gets the path on one of the K possible states of the path. After H steps, this path will be in state $(y_i^0, \tau_i + H)$ if it has never been measured. If it has been measured for the last time at depth $h = 1, \ldots, H$, then it will be in state $(y_i^h, H-h+1)$, where y_i^h can take any of the K possible state values in $\mathcal{S}_i$. Hence, we have $(1+KH)$ distinct information states at depth H for each of the P paths. This results in $(1+KH)^P$ possible information states for the whole network, at depth H. We therefore conclude that the complexity of the receding horizon approximation is linear in H and K, while it is exponential in the number of paths P. We address this issue in the following subsection.

3.3 Complexity reduction

As discussed above, the number of information states increases exponentially with the number of paths P. However, if we restrict the number of paths that can be measured at each step to at most one path (thereby considering only a subset of the action space), then the number of possible information states is considerably reduced.

Indeed, consider an information state $s = (s_i)_{i=1\ldots P}$. The number of

Algorithm 1 Receding Horizon

function RECEDINGV($\mathbf{s}, h$)
  inputs: state $\mathbf{s}$, depth h
  if $h = 0$ **then**
    $V^0(\mathbf{s}) \leftarrow 0$
    return $V^0(\mathbf{s}), any\ \mathbf{u}$
  else
    for each state $\mathbf{s}'$ reachable from $\mathbf{s}$ **do**
      $V^{h-1}(\mathbf{s}'), \mathbf{u}^{h-1^*} \leftarrow$ RECEDINGV($\mathbf{s}'$,$h-1$)
    end for
    $\bar{D}(\mathbf{s}, .) \leftarrow$ solve Eqn. (5) for each action $\mathbf{u}$
    $V^h(\mathbf{s}) \leftarrow \min_{\mathbf{u}} \left[\bar{D}(\mathbf{s};\mathbf{u}) + \rho \sum_{\mathbf{s}'} \pi(\mathbf{s},\mathbf{s}';\mathbf{u}) V^{h-1}(\mathbf{s}')\right]$
    $\mathbf{u}^{h^*} \leftarrow \arg\min_{\mathbf{u}} \left[\bar{D}(\mathbf{s};\mathbf{u}) + \rho \sum_{\mathbf{s}'} \pi(\mathbf{s},\mathbf{s}';\mathbf{u}) V^{h-1}(\mathbf{s}')\right]$
    return $V^h(\mathbf{s}), \mathbf{u}^{h^*}$
  end if
end function
function MAIN($\mathbf{s}$, H)
  inputs: current state $\mathbf{s}$, receding horizon depth $H > 0$
  $V^H(\mathbf{s}), \mathbf{u}^{H^*} \leftarrow$ RECEDINGV($\mathbf{s}$,H)
  return $\mathbf{u}^{H^*}$
end function

different information states at level H is given by the different values each component of such vector (i.e. s_i) can take. Consider a departure information state of the form $(y_i, \tau_i)_{i=1...P}$. As in the previous case, the values s_i can take are $(y_i, \tau_i + H - h)$, y_i being any of the K states of path i, and h being the level with the highest depth where path i is measured. Note thus that what determines the different information states at level H is given by the different paths that are measured within H levels, and the possible different values h can take. This is quantified in (11),

$$\sum_{n=0}^{P} P(H,n) \binom{P}{n} K^n = \sum_{n=0}^{H} \frac{H!}{(H-n)!} \frac{P!}{(P-n)!n!} K^n, \tag{11}$$

where $P(H,n)$ represents the number of $n-$permutations of H.

The intuition behind (11) is as follows. From root to level H, consider n different paths that are measured, this leads to K^n different information states at level H. In addition, this n paths can be any of the P available paths, we have thus n-combinations of P possibilities to choose n paths to be measured. In addition, the order on which the paths are measured provides different information states. We have n-permutations of H orderings of measurements. Finally, n takes all values from 0 to P. Note that if $P > H$, (11) is still valid since terms from $P+1$ onwards are equal to 0.

We shall evaluate the receding horizon approach along with this constraint on the number of actions in Section 5.

4 Heuristic 2: Reducing the MDP's state space

We now propose an heuristic that reduces the dimension of the state space over which the value iteration is performed in (8). Indeed, the main drawback of (8) is that the state space grows exponentially in the number of paths to be explored.

Instead, we propose to compute the optimal policy for each path locally using a value iteration algorithm and assuming that the other paths can be summarized by one virtual path. That is, to take the monitoring decision for each path i, we solve an MDP with two paths. One of them is the path in question and the other one is a virtual deterministic path that reflects the behaviour of the other paths. We further assume that the virtual path is not controlled, which simplifies even more the computation of the optimal policy for the path in question.

The proposed approach decomposes the original MDP into P smaller MDPs, one for each path. For path i, we first compute the expected delay, $\bar{D}_{-i}$ of the other paths[1] given the current beliefs and the actions assuming that these actions would be carried out in the next step. That is,

$$D_{-i}(\cdot) = \min_{j \neq i} \left(u_j \ell_{j,x_j} + (1 - u_j) \sum_{y \in \mathcal{S}_j} b_{j,y}(\mathbf{s}_j) \ell_{j,y} \right). \tag{12}$$

and

$$\bar{D}_{-i} = \sum_{\mathbf{x} \in \mathcal{S}_{-i}} \left(\prod_{j=1, j \neq i}^{P} b_{j,x_j}(\mathbf{s}_j) \right) D_{-i}(\cdot). \tag{13}$$

Remark that (13) is similar to (5) but for two main differences. First, path i is excluded in the computation of the mean delay which depends upon the current state and the current actions of the other paths only. Second, the cost of monitoring of the other paths is not taken into account. Since the monitoring decisions on the other paths are fixed, we know what delay to expect from these paths. The decision problem for path i, is to determine whether by monitoring it we can get a better delay than that of the other paths. Thus, we do not need to take into account the monitoring cost of the other paths.

To make the notation easier to read, we have not made explicit the dependence of D_{-i} on $(\mathbf{s}_{-i}, \mathbf{x}_{-i}; \mathbf{u}_{-i})$, i.e. we just write D_{-i} instead of $D_{-i}(\mathbf{s}_{-i}, \mathbf{x}_{-i}; \mathbf{u}_{-i})$. We shall do the same in the rest of the paper whenever this does not lead to ambiguity.

Now that the delays of the other paths are represented by a deterministic quantity $\bar{D}_{-i}$, we solve an MDP for path i assuming that it is in competition

[1] We shall hereon use the notation x_{-i} to mean a quantity for all other paths except path i are considered.

with a path of delay $\bar{D}_{-i}$. Towards this end, we first compute the expected routing cost for action u_i as was done in (5) but with one deterministic path:

$$D_i(\cdot) = \min \left(u_i \ell_{i,x_i} + (1-u_i) \sum_{y \in \mathcal{S}_i} b_{i,y}(\mathbf{s}_i) \ell_{i,y}, \bar{D}_{-i} \right),$$

and

$$\bar{D}_i(\cdot) = c_i \, u_i + \sum_{x_i \in \mathcal{S}_i} b_{i,x_i}(\mathbf{s}_i) \, D_i(\cdot). \tag{14}$$

The Bellman equation for the MDP for path i is given by:

$$V(\mathbf{s}_i) = \min_{u_i} \left[\bar{D}_i(\cdot) + \rho \sum_{\mathbf{s}_i'} \pi_i \left(\mathbf{s}_i, \mathbf{s}_i'; u_i \right) V(\mathbf{s}_i') \right]. \tag{15}$$

This heuristic is meant to be applied online in an iterative fashion. We are given some initial states and monitoring actions for each path. At each decision epoch, we solve an MDP for each path to determine its local optimal monitoring action. These monitoring actions are then implemented. Based on the observations, the optimal path is chosen and the states are updated for the next decision epoch. The details are given in Algorithm 2. Note that the step requiring the most computation resources is Step 7 in which the Bellman equation is solved for path i. This step has complexity $O((K\tau_{max})^2)$ for one path, which translates into a complexity of $O(P(K\tau_{max})^2)$ at each decision epoch.

Algorithm 2 Heuristic 2

1: **input:** $\mathbf{s}^{(0)}$ initial state
2: $n \leftarrow 1$ // decision epoch
3: **while** true **do**
4: $\quad \mathbf{u}^{(n)} \leftarrow \mathbf{0}$
5: $\quad$ **for** $i \in \mathcal{P}$ **do** // determine which paths are monitored
6: $\quad\quad D_{-i} \leftarrow$ solve (13) with $\mathbf{s}_{-i}^{(n-1)}$ and $\mathbf{u}_{-i}^{(n)}$ as inputs
7: $\quad\quad$ Solve Bellman equation (15)
8: $\quad\quad \mathbf{u}_i^{(n)} \leftarrow$ optimal action in state $\mathbf{s}_i^{(n-1)}$
9: $\quad$ **end for**
10: $\quad$ Implement action $\mathbf{u}^{(n)}$ and update $\mathbf{s}^{(n)}$
11: $\quad$ From the observed delays, compute optimal path
12: $\quad n \leftarrow n+1$
13: **end while**

An important remark is that the locally optimal monitoring action computed in Step 8 is used to update the current monitoring action vector.

Once the monitoring action for path i is known at epoch n, it is logical to use this information to compute the deterministic expected delays for paths for which the monitoring action is computed after that of path i. This asynchronous update of the monitoring actions means that the quality of the heuristic will depend upon the order in which the paths are iterated upon in Step 5 of the heuristic. In the numerical section, we shall assume that this order corresponds to the order of decreasing expected delays.

5 Numerical Results

Two types of validation results are presented in this section. We first describe in Section 5.1 the results obtained with small synthetic scenarios for which the optimal monitoring policy can be computed. The goal is to evaluate and compare the accuracy of the proposed monitoring solutions. In Section 5.2, we evaluate the robustness and the scalability of the proposed solutions using latency data collected over the Internet.

5.1 Synthetic scenarios

The first example shows that the myopic policy can have relatively bad performance compared to the heuristics. There are two paths with two states each. The delays are given by $\boldsymbol{\ell}_1 = \boldsymbol{\ell}_2 = (0.01, 100)$, and the transition matrices are as follows

$$P_1 = P_2 = \begin{pmatrix} \beta & 1-\beta \\ 1-\beta & \beta \end{pmatrix}.$$

Note that the paths have symmetric parameters. For $c = 25$, $\rho = 0.99$, and different values of β, we have computed the optimal monitoring policy for this example, as well as the monitoring policies obtained with the proposed heuristics. The latter policies have been evaluated using the *Policy Evaluation* algorithm [23]. The mean relative error (MRE), defined as $\frac{1}{S}\sum_{\mathbf{s}}(V_\mu(\mathbf{s}) - V^*(\mathbf{s}))/V^*(\mathbf{s})$, of our heuristic policies is given in Figure 2. We observe that the myopic policy performs badly when the transitions of Markov chains occurs infrequently and there is a large difference between the delays of the two states.

In the second example, we show that the myopic policy can be better or equal than both heuristics in some cases. Consider the following example with three stochastic paths with two states each. The delays are given by $\boldsymbol{\ell}_1 = (1, 3)$, $\boldsymbol{\ell}_2 = (\frac{1}{2}, 4)$, and $\boldsymbol{\ell}_3 = (\frac{1}{4}, \frac{15}{4})$, and the transition matrices are as follows:

$$P_1 = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix},\ P_2 = \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix},\ P_3 = \begin{pmatrix} 0.65 & 0.35 \\ 0.35 & 0.65 \end{pmatrix}.$$

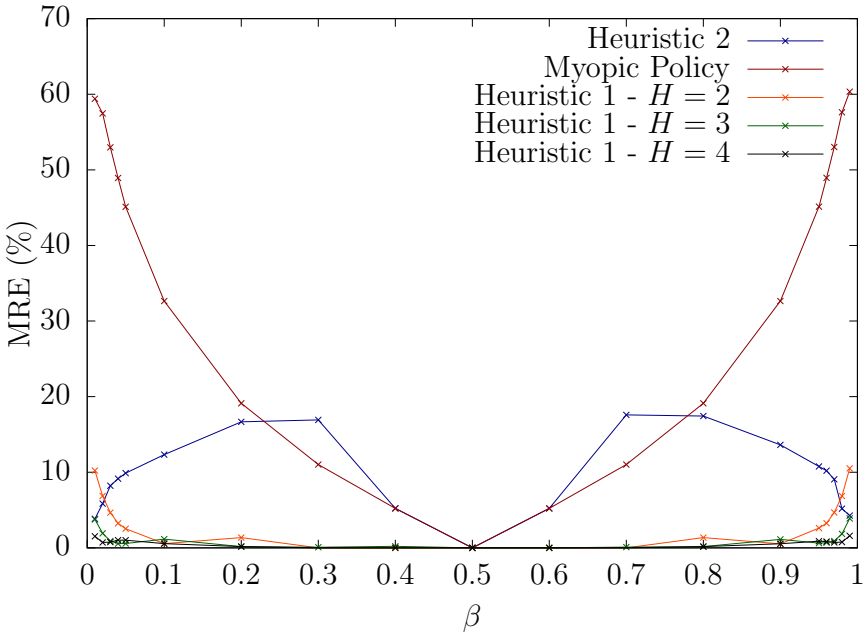

Figure 2: Example 1 - MRE in % of the myopic policy and of both heuristics.

Note that the steady state expected delays of the three paths are very close (1.99, 1.9 and 2). The maximum age of an observation is taken to be $\tau^1_{max} = 20$ for path 1, whereas we use $\tau^2_{max} = \tau^3_{max} = 10$ for the two other paths. It yields $S = 16,000$ belief states. Table 1 gives the MRE for our heuristic policies and for different values of ρ and c.

We observe from the above two examples that the two heuristics perform relatively well in both settings whereas the myopic policy has a much more variable performance.

5.2 Validation against real latency data

In order to assess the applicability of our results against a real-world scenario, in which our Markovian assumptions are not perfectly met, we simulate an overlay using NLNog RTT measurements. These RTT measurements were collected in 2014 during an Internet-scale experiment where 20 nodes of the *NLNog ring*[2] were used. The RTT was measured between every pair of nodes every two minutes for a period of 5 days using the ICMP-based ping utility, resulting in 3836 samples per origin-destination pairs (see [8] for further details). This dataset is particularly convenient as measurements are performed between every nodes in a full-mesh topology. Hence we can

[2]The NLNog ring is a network of 517 nodes scattered over 55 countries (see https://ring.nlnog.net).

ρ	c	MRE of Heuristic (%)	MRE of Myopic Policy (%)	MRE of Receding (%)
0.99	0.5	9.19	0.60	0.08
0.99	0.25	6.68	0.88	0.20
0.99	0.125	7.21	0.10	0.10
0.99	0.0625	7.86	0.15	0.15
0.999	0.5	11.97	0.724	0.19
0.999	0.25	7.45	1.00	0.30
0.999	0.125	5.50	0.23	0.23
0.999	0.0625	6.97	0.27	0.27

Table 1: Example 2: Mean relative errors over all belief states of the Heuristics proposed. For the receding horizon we used an horizon of H=3, without the restriction on measuring only one route at each step.

simulate the delay between two nodes A and B going through a proxy node C by summing-up the measurements between A and C, and C and B. For each origin-destination (OD) pair in the dataset, we select alternative paths going through one proxy node that have a shorter delay at-least 10% of the time.

Out of the 380 OD pairs, we selected 4 OD pairs for which 1) a significant decrease in the round-trip delay can be achieved by selecting another path than the direct IP route, and 2) for which the minimum-latency route is not always the same. These OD pairs are described in Table 2. We also give the processing time per time step of both heuristics for each of the these OD pairs. In all cases, the processing time is much lower than the 2 minutes between decision points. The Markovian model for each OD pair has been learnt using the HDP-HMM (Hierarchical Dirichlet Process Hidden Markov Model) approach described in [24]. In contrast to the classical HMM where the number of states must be known at learning time, the HDP-HMM treats the number of states as unknown and sample it from the data, alongside the other model parameters. Inference is performed in linear time (with respect to the number of data points) using Gibbs sampling. This allows us to obtain Markovian models for real Internet delay measurements where the ground truth on the number of network states is not known. Note that in most cases the computation of the optimal monitoring policy is prohibitively expensive.

Assuming that $\tau_{max} = 100$, $\rho = 0.99$ and $c = 0.5$, we then ran the heuristic presented in algorithm 2 and receding horizon described in algorithm 1 for each OD pair to compute the monitoring decision $\mathbf{u}(t)$ and the routing path $r(t)$ at each time step $t = 1, \ldots, 3836$. The approach used is similar to the one described in [10]. In particular, we have simplified the HMM models

Origin-Destination pair	Number of paths	Number of belief states	Processing time (s)	
			H	RH
Haifa-Santiago	4	$\approx 3.1 \times 10^{15}$	14.5	35.6
Paris-Santiago	4	$\approx 7.9 \times 10^{16}$	35.8	21.9
Paris-Tokyo	2	$13,950$	6.1	0.04
Singapore-HongKong	3	$\approx 1.1 \times 10^{10}$	2.66	8.3

Table 2: Number of paths and number of belief states for the selected origin-destination pairs.

of the paths to MC models, by neglecting the Gaussian noise of the HMM model and considering only mean values. It is also worth mentioning that since we do not directly observe the state of a monitored path, this state was estimated as the most likely state of the HMM given past observations.

The results obtained for these 4 scenarios are summarized in Table 3, where H-S stands for Haifa-Santiago, P-S for Paris-Santiago, etc. Figures 3 (resp. 4) shows the latencies obtained with Heuristic 2 (resp. 1) for the Singapore-HongKong (resp. Haifa-Santiago) scenario. In this case, the heuristic monitors only 1.06 (resp. 0.29) paths on the average out of the 4 paths (that is, each path is monitored only 25% (resp. 7%) of the time), and yet the routing performance in terms of delay is very close to the performance that one could get if all paths were constantly monitored. We also note a delay reduction with respect to the direct IP path. We should however point out that significantly improving over the IP path is not always possible (this is the case for the Paris-Tokyo OD pair). It is also worth emphasizing that achieving the minimum delay most of the times is difficult when the RTTs have a great variability (as is the case for the Haifa-Santiago OD pair), mainly because the routing decisions are based solely on the expected values of the paths' delays.

O-D pair	Avg. number of measures			% of time min delay is reached			gap to min delay (%)		
	H_2	H_1	MP	H_2	H_1	MP	H_2	H_1	MP
H-S	0.70	0.29	0.12	74	87	84	1.6	0.2	0.3
P-S	0.28	0.11	0.05	88	97	93	0.7	0.1	0.2
P-T	0.0	0.02	0.002	79	88	84	0.2	0.1	0.2
S-H	1.06	0.56	0.4	99	98	98	0.3	0.8	0.8

Table 3: Summary of the results achieved for the 4 considered OD pairs, where H_2 stands for heuristic 2, H_1 for heuristic 1 (receding horizon approximation) and MP for the myopic policy. We used an horizon of H=3 for the receding policy.

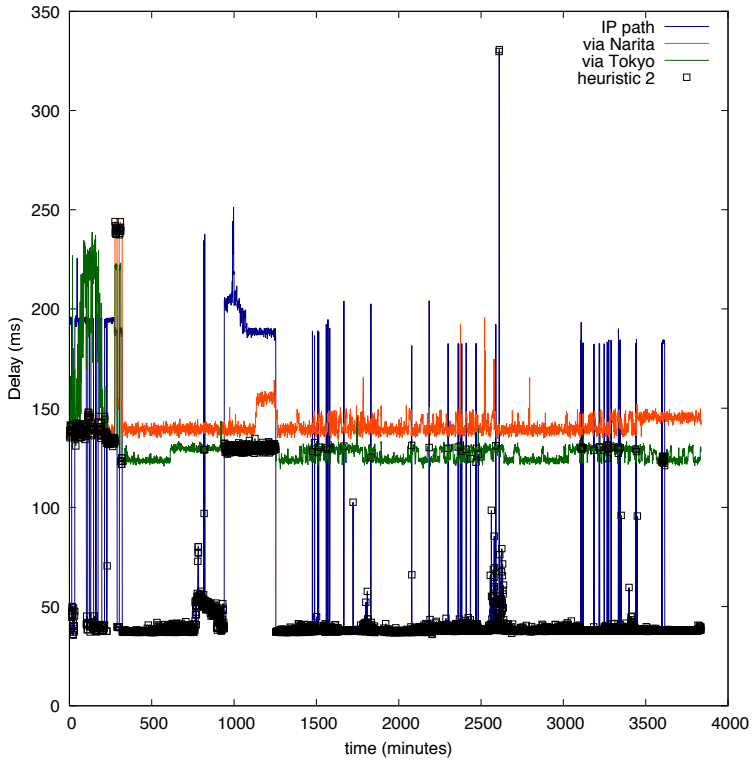

Figure 3: End-to-end delay between Singapore and Hong Kong. On average, Heuristic 2 monitors only 1.06 paths per time step, and yet it provides the minimum delay 99.2 % of the time.

6 Conclusion and future work

We proposed two scalable monitoring heuristics for the problem of jointly optimizing end-to-end delay and monitoring cost over an origin-destination pair in an overlay network. Numerical evaluation on real as well as synthetic data show both the heuristics have good performance in terms of delays and costs and can generate a decision in a few seconds which makes them implementable on practical time-scales where decisions are made every few minutes.

As for the next steps, it would be interesting to have theoretical guarantees on the inefficiency of the heuristics. Further, we are also investigating scalable heuristics for other performance metrics such as bandwidth and energy.

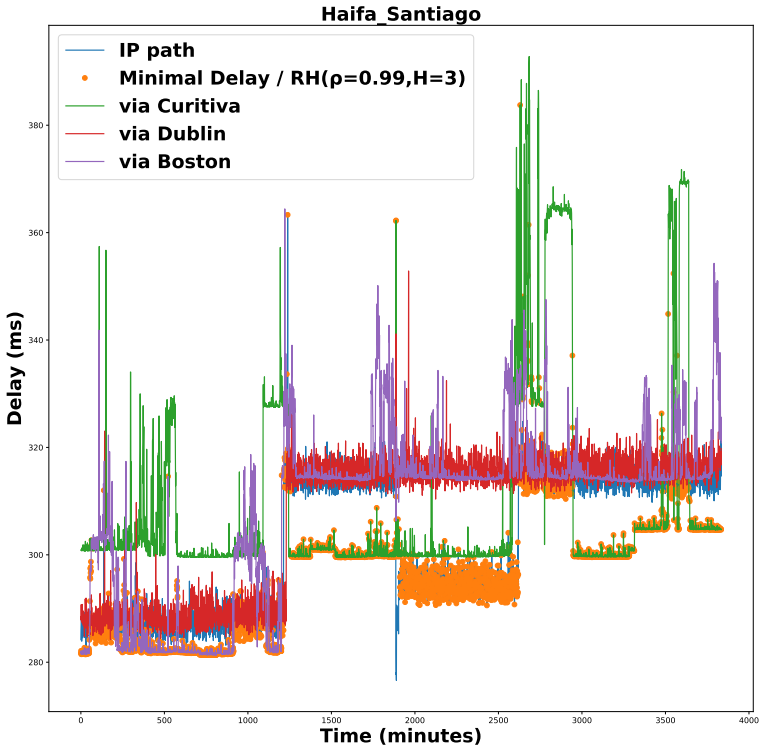

Figure 4: End-to-end delay between Haifa and Santiago de Chile. On average, Heuristic 1 monitors only 0.29 paths per time step, and yet it provides the minimum delay 87.4 % of the time.

Acknowledgment

The work of Martin Randall and Pablo Belzarena was partially supported by Agencia Nacional de Investigación e Innovación, Uruguay. The work of Marine Ségneré was partially funded by a contract with Direction Générale de l'Armement (DGA), France.

References

[1] M. Beck, T. Moore, and J. Plank, "An end-to-end approach to globally scalable programmable networking," in *in Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, A. Press, Ed., 2003.

[2] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The case for separating routing from routers," in *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, A. Press, Ed., 2004.

[3] C. Thraves Caro, J. Doncel, and O. Brun, "Optimal path discovery problem with homogeneous knowledge," *Theory of Computing Systems*, vol. https://doi.org/10.1007/s00224-019-09928-w, pp. 1–24, 2019.

[4] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, ser. SOSP '01. New York, NY, USA: ACM, 2001, pp. 131–145. [Online]. Available: http://doi.acm.org/10.1145/502034.502048

[5] K. P. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall, "Improving the reliability of Internet paths with one-hop source routing." in *In Proceedings of the 6th Symposium on Operating Systems Design and Implementation*, 2004.

[6] S.-Y. Hu and G.-M. Liao, "Scalable peer-to-peer networked virtual environment," in *In NetGames'04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games.* New York, NY, USA: ACM Press, 2004, pp. 129–133.

[7] A. Nakao, L. Peterson, and A. Bavier, "Scalable routing overlay networks," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 49–61, 2006.

[8] O. Brun, L. Wang, and E. Gelenbe, "Big data for autonomic intercontinental overlays," *IEEE Jour. Selected Areas in Communications (special Issue on Emerging Technologies in Communications - Big data)*, vol. 34, pp. 575–584, 2016.

[9] O.Brun, H. Hassan, and J. Vallet, "Scalable, self-healing, and self-optimizing routing overlays," in *IFIP Networking 2016*, Vienna, Austria, May 17-19 2016.

[10] S. Vaton, O. Brun, M. Mouchet, P. Belzarena, I. Amigo, B. Prabhu, and T. Chonavel, "Joint Minimization of Monitoring Cost and Delay in Overlay Networks: Optimal Policies with a Markovian Approach," *Journal of Network and Systems Management*, vol. 27, no. 1, pp. 188–232, Jan. 2019.

[11] "RIPE Atlas," https://atlas.ripe.net/, accessed: 2019-11-26.

[12] D. Rubenstein, J. Kurose, and D. Towsley, "Detecting shared congestion of flows via end-to-end measurement." *IEEE/ACM Transactions on Networking*, vol. 10, no. 3, p. 381–395, 2002.

[13] A. Coates, A. O. H. III, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal Processing Magazine*, vol. 19, no. 3, pp. 47–65, May 2002.

[14] Y. Vardi, "Network Tomography: estimating source-destination traffic intensities from link data." *Journal of the American Statistical Association. American Statistical Association*, vol. 91, no. 433, p. 365–377, 1996.

[15] Y. Bejerano and R. Rastogi, "Robust monitoring of link delays and faults in IP networks." *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. 5, pp. 1092 – 1103, Oct 2006.

[16] J. Horton and A. Lopez-Ortiz, "On the number of distributed measurement points for network tomography." in *Proceedings of the 2003 ACM SIGCOMM conference on Internet measurement*, 2003, p. 204–209.

[17] Y. A. Pignolet, S. Schmid, and G. Trédan, "Tomographic Node Placement Strategies and the Impact of the Routing Model." *Proc. ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 42:1–42:23, 2017.

[18] A. Gopalan and S. Ramasubramanian, "On identifying additive link metrics using linearly independent cycles and paths." *IEEE/ACM Transactions on Networking (TON)*, vol. 20, no. 3, pp. 906 – 916, Jun 2012.

[19] L. Ma, T. He, K. K. Leung, D. Towsley, and A. Swami, "Efficient identification of additive link metrics via network tomography." in *IEEE ICDCS*, 2013.

[20] D. Z. Tootaghaj, T. He, and T. L. Porta, "Parsimonious tomography: Optimizing cost-identifiability trade-off for probing-based network monitoring," in *IFIP Performance 2017*, 2017.

[21] J. P. Singh, T. Alpcan, P. Agrawal, and V. Sharma, "A markov decision process based flow assignment framework for heterogeneous network access," *Wireless Networks*, vol. 16, pp. 481–495, 2010.

[22] G. C. Goodwin, M. M. Seron, and J. A. D. Doná, *Constrained Control and Estimation. An Optimisation Approach.* Springer-Verlag London, 2005.

[23] V. Krishnamurthy, *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing.* Cambridge University Press, 2016. [Online]. Available: https://books.google.fr/books?id=FJSzCwAAQBAJ

[24] M. Mouchet, T. Chonavel, and S. Vaton, "Statistical Characterization of Round-Trip Times with Nonparametric Hidden Markov Models," in *IFIP/IEEE IM 2019 Workshop: 4th International Workshop on Analytics for Network and Service Management (AnNet 2019)*, Washington, DC, United States, Apr. 2019.