Inferring Catchment in Internet Routing

PAVLOS SERMPEZIS, Institute of Computer Science, FORTH, Greece
VASILEIOS KOTRONIS, Institute of Computer Science, FORTH, Greece

BGP is the de-facto Internet routing protocol for exchanging prefix reachability information between Autonomous Systems (AS). It is a dynamic, distributed, path-vector protocol that enables rich expressions of network policies (typically treated as secrets). In this regime, where complexity is interwoven with information hiding, answering questions such as "what is the expected catchment of the anycast sites of a content provider on the AS-level, if new sites are deployed?", or "how will load-balancing behave if an ISP changes its routing policy for a prefix?", is a hard challenge. In this work, we present a formal model and methodology that takes into account policy-based routing and topological properties of the Internet graph, to predict the routing behavior of networks. We design algorithms that provide new capabilities for *informative* route inference (e.g., isolating the effect of randomness that is present in prior simulation-based approaches). We analyze the properties of these inference algorithms, and evaluate them using publicly available routing datasets and real-world experiments. The proposed framework can be useful in a number of applications: measurements, traffic engineering, network planning, Internet routing models, etc. As a use case, we study the problem of selecting a set of measurement vantage points to maximize route inference. Our methodology is general and can capture standard valley-free routing, as well as more complex topological and routing setups appearing in practice.

CCS Concepts: • **Networks** → **Network performance analysis**; *Network performance modeling*; Network structure; Network management.

Additional Key Words and Phrases: Border Gateway Protocol (BGP); Internet Routing; IP Anycast; Catchment Inference; Internet Measurements.

1 INTRODUCTION

Routing between networks (or Autonomous Systems–AS) in the Internet takes place via the Border Gateway Protocol (BGP) [41]. BGP is a policy-based, destination-oriented path-vector protocol, where an AS receives paths to a destination network from its neighbors, selects which path to prefer based on its local routing policies, and advertises it to other neighbors based on its export policies. This typically results in asymmetric paths between networks [20]. Each destination network has control only over *its own* routing decisions, and typically *cannot control or even know how other networks route their traffic* to it.

Knowing how networks route traffic to a destination is important for (i) network planning or monitoring (*e.g.,* allocation of network resources, detection of routing anomalies) [9, 13, 22], and

30

RIGHTSLINK()

(ii) indirect control –if possible– of routing decisions of other networks (*e.g.,* through manipulation of BGP announcements, selection of local routing policies, establishment of new links) [29, 30]. Specifically, for a destination network, it is of particular interest to know from which of its *ingress points* (*e.g.,* border routers) it should expect to receive traffic from other networks under a given routing configuration [2]. We consider the following indicative examples.

Example A: A regional ISP R, whose network spans a region of two major cities, $city_A$ and $city_B$, has a single upstream tier-1 ISP T_A and connects to it at $city_A$. To avoid overloading its infrastructure in $city_A$, R decides to connect to another tier-1 ISP T_B at $city_B$. However, after connecting with T_B, R observes that 90% of the incoming Internet traffic still enters its network at $city_A$, therefore the new setup fails to balance R's load among its infrastructure in the two cities. In fact, how to select a transit provider, is a question that lacks a clear answer, and engages operators in active discussions [35].

Example B: A content provider C applies IP anycast (*i.e.,* announces the same IP prefix) [9, 13, 29, 33, 49] from three sites. Due to traffic increase, C decides to add one more anycast site. It needs to select where to deploy and how to connect the new site, in order to best split the traffic among its sites. The ongoing research in IP anycasting, *e.g.,* [13, 28, 49], indicates that this is a problem that is not well-understood yet.

While a network can *partially* determine how other networks route traffic to it through passive (*e.g.,* BGP data [37]) or active (*e.g.,* traceroute, ping) measurements [2, 9, 13, 27, 32, 49], *measurements can provide information only for an existing deployment*. However, in many applications (traffic engineering, peering decisions, network resilience, etc.) [30], it is important to know, *i.e.,* predict, how the routing behavior of other networks will change *in advance*, before a network actually alters its local policies or connections. Moreover, even when it is possible to afford several trials to test different traffic engineering (TE) decisions, the large number of possible options limits the efficiency and/or applicability of this "trial-and-error" approach, *unless an informed methodology is used*.

To this end, the primary goal of this paper is to *provide an informative inference or prediction for the catchment of the ingress points of a network, under a given (existing or not) topological and routing configuration*. With the term "catchment" (see, *e.g.,* [13, 29, 49]) of an ingress point m of a destination network n_{dst}, we denote the set of networks that route their traffic to n_{dst} through m.

Route inference is identified as a challenging task [29, 30], due to the inherent complexity of the behavior of BGP mechanisms, and lack of public data for networks' routing policies (in fact, only coarse estimates are available, *e.g.,* the AS-relationships [5, 31]). Moreover, the related problems (*e.g.,* TE optimization) that may arise in practice are typically of combinatorial nature [34].

The common approach to predict routes is to use models, such as the valley-free model [17] or other variants [15, 34, 40], that simulate the Internet routing process (BGP) based on available data. Policies are typically inferred from public data [5, 34], and when there is lack of, or coarse-grained, knowledge of policies, they are arbitrarily selected (*e.g.,* random tie-breaking), in order to proceed to a simulation and obtain a prediction. However, a simulator computes only one of all the possible outcomes per simulation run. Thus, this approach can lead to an *output that is heavily affected by the introduced randomness* (*e.g.,* breaking randomly a tie for a central AS in each simulation, may lead to high catchment for an ingress point m in one simulation run, and to low catchment in another run). Most importantly, the output *does not reveal what is the effect of the randomness*, *e.g.,* how many routes are affected by an arbitrarily chosen policy.

In this paper, we revisit the problem of route/catchment inference, and propose a framework and methodology for an informative inference that quantifies the certainty/uncertainty in the prediction for every network (isolating the effect of randomness), and reveals the factors that affect the inference (*e.g.,* certain policies or networks). This in turn enables the development of

advanced methods for optimizing traffic engineering, selecting peering strategies, or conducting measurement campaigns. Specifically:

- We formally model (Section 2) and study the problem of inferring the catchment of the ingress points of a network. To this end, we propose a graph structure, the R-graph, that can efficiently encode rich information about the routing behavior, and isolate the effect of randomness (Section 3.1).
- We design and analytically study methodologies that infer catchment in existing or hypothetical scenarios (Section 3). We identify the networks for which a certain inference is possible, even under coarse estimates of routing policies and topology (Section 3.2), calculate the *probabilistic inference* for the remaining networks (Section 3.3), and enhance the inference when some oracles (*e.g.,* from *measurements*) are given (Section 3.4). The code for an implementation of the proposed methods is available in [46].
- As a use case of our framework, we consider and study the problem of maximizing the inference of catchment under a limited budget of measurements. We propose an efficient greedy algorithm, which leverages the structure of the R-graph, for selecting the measurement targets (Section 4). Our analysis sheds light on the complexity of problems related to route inference, and can be of more general interest.

While the main focus of the paper is on establishing a theoretical framework for catchment inference, we provide an initial evaluation of our approach in realistic Internet routing scenarios through extensive simulations and real experiments, and provide insightful results (Section 5). We present related work in Section 6, and conclude by discussing potential applications and future research directions in Section 7.

As a final remark, we would like to stress that our goal is not to propose a new inter-domain routing model [17], or infer more accurately the routing policies in the Internet [34], but to provide inference methods and insights on top of *any* given model and set of policies. Finally, we believe that our methods can be useful for more general applications of BGP (or, similar policy-oriented path-vector protocols), apart from inter-domain routing, such as in iBGP or data centers [26].

2 MODEL

We present our model in Section 2.1, and provide the necessary definitions related to route inference in Section 2.2. In Section 2.3 we discuss how the commonly used valley-free model [17] can be captured as a special case of our generic model. Important notation is summarized in Table 1.

2.1 Network and Routing

We assume a network with a set of nodes $\mathcal{N}$ and edges $\mathcal{E}$. A node may correspond to a single AS, or a part of an AS (*e.g.,* in case of large/distributed ASes; similarly to the concept of "quasi-routers" in [34]), or even a group of ASes with the same routing policies (*e.g.,* siblings). For brevity, and without loss of generality, in the remainder we consider that a node represents a single AS[1], and an edge corresponds to a peering link between two ASes. We refer to the nodes connected with an edge to a node i, as the neighbors of i.

Routing protocol and policies. Nodes use BGP [41] to establish routes towards different Internet destinations. The main operation of BGP is described as follows. A destination node n_{dst} announces a prefix. Every other node i learns from its neighbors paths to n_{dst} (*i.e.,* its prefix), stores them in a local routing table (Routing Information Base, RIB), and selects one of them as its best path to

---

[1]The study of [34] showed that more than 98% of the ASes can be accurately (w.r.t. inter-domain routing behavior) represented as a single node / "quasi-router".

n_{dst} (according to, *e.g.*, its *local preferences*). Then, i may advertise this best path to its neighbors (according to its *export policies*).

A path contains a sequence of nodes; we denote a path from i to j as $p_{i \to j}$, and use the following notation:

$$p_{i \to j} = [i, x, y, ..., z, j], \ i, x, y, ..., z, j \in \mathcal{N}$$

We further denote the best path from i to j (*i.e.*, the path that i prefers –among all paths in its RIB– to reach j) as $bp_{i \to j}$.

Best path selection. Each node i assigns a *local preference* to each of its neighbors. We denote the set of local preferences in the network as $Q = \{q_{ij} \in \mathbb{R} : i, j \in \mathcal{N}, e_{ij} \in \mathcal{E}\}$. Note that local preferences are in general asymmetric, *i.e.*, $q_{ij} \neq q_{ji}$. If paths are learned from more than one neighbors, then i prefers the path learned from the neighbor with the highest local preference [41]. If a node i has the same local preference for two neighbors j and k ($q_{ij} = q_{ik}$), then the selection is based on other criteria ("tie-breakers"), such as path length (see Section 3.5), the MED attribute, IGP metrics, time of advertisement, etc. [10].

Path export. When a node i selects a best path for n_{dst} via a neighbor j, it may advertise (export) this path to all, some or none of its neighbors. We denote the set of export policies as $\mathcal{H} = \{h_{ijk} \in \{0, 1\} : i, j, k \in \mathcal{N}, e_{ij}, e_{ik} \in \mathcal{E}\}$, where $h_{ijk} = 1$ denotes that i exports to k a path learned from j (and $h_{ijk} = 0$ otherwise). Typically, both export policies and local preferences are based on the economic relationships between the nodes, and are consistent with each other. Therefore, it is safe to assume for practical setups that $q_{ij} = q_{i\ell} \Rightarrow h_{ijk} = h_{i\ell k}, \forall k$, *i.e.*, routes learned from neighbors with the same local preference are similarly treated[2].

Remark on the generality of the model: (i) The proposed model allows to capture generic routing policies by carefully selecting the quantities Q and $\mathcal{H}$; even sophisticated per-prefix policies can be captured by considering different policies Q^p and $\mathcal{H}^p$ per prefix p. (ii) The model can be applied in generic settings: when the detailed policies of a node are known by explicitly setting the Q and $\mathcal{H}$ values; or when we have only coarse-grained information about them (see Section 2.3); or even when we entirely lack policy information for some nodes, where its values for Q and $\mathcal{H}$ can be set equal to a default value, thus without excluding any possible outcome.

Eligible paths. We define the *eligible paths* of a node i to a node n_{dst}, as the paths that can be in the RIB of i; thus, one of them can be selected by i as its best path to n_{dst}. The eligible paths are later used in the route inference methodology (Section 3.2).

DEFINITION 1 (ELIGIBLE PATH). *An eligible path $p_{i \to n_{dst}}$ is a path from i to n_{dst} that (i) conforms to the routing policies Q and $\mathcal{H}$, and (ii) can be selected by i as its best path to n_{dst}.*

The first condition in Def. 1 dictates that only paths that can be received by i (*i.e.*, be in its RIB) can be eligible. For example, if $h_{ijk} = 0$, then i will not export to k a path learned from j, and thus the path $[k, i, j, ..., n_{dst}]$ does not conform to the routing policies $\mathcal{H}$ and cannot be eligible. The second condition excludes paths which are not preferred due to i's local preferences. For instance, let i have in its RIB two paths $p_{i \to n_{dst}}^{(1)} = [i, j, ..., n_{dst}]$ and $p_{i \to n_{dst}}^{(2)} = [i, k, ..., n_{dst}]$. If $q_{ij} > q_{ik}$, then $p_{i \to n_{dst}}^{(2)}$ is *not* eligible, since $p_{i \to n_{dst}}^{(1)}$ will always be preferred by i. However, if $q_{ij} = q_{ik}$, both paths are eligible.

---

[2]In case a node has different export policies for neighbors of same local preference, we can split the node into more than one sub-nodes (with the same neighbors and local preferences), each of them corresponding to one export policy.

Table 1. Important Notation.

$\mathcal{G}$	Network graph; $\mathcal{G}(\mathcal{N}, \mathcal{E}, Q, \mathcal{H})$
$\mathcal{N}$	Set of nodes in $\mathcal{G}$
$\mathcal{E}$	Set of edges e_{ij} in $\mathcal{G}$
Q	Local preferences $Q = \{q_{ij} \in \mathbb{R} : i, j \in \mathcal{N}, e_{ij} \in \mathcal{E}\}$
$\mathcal{H}$	Export policies $\mathcal{H} = \{h_{ijk} \in \{0, 1\} : i, j, k \in \mathcal{N}, e_{ij}, e_{ik} \in \mathcal{E}\}$
$p_{i \to j}$	Path from node i to node j
$bp_{i \to j}$	Best path from node i to node j
$\mathcal{M}$	Ingress points of the destination node n_{dst}
$i \triangleright m$	Node route; i reaches n_{dst} through the ingress point m
$\pi_i(m)$	Route probability for node i and ingress point m, Eq. (1)
f	Routing function, Eq. (2)
$\mathcal{G}_R$	R-graph; $\mathcal{G}_R(\mathcal{N}_R, \mathcal{E}_R)$

2.2 Ingress Points and Catchment

Ingress points. Let a network n_{dst} that originates a prefix, and is connected to its neighbors (and receives traffic) through a set of ingress points $\mathcal{M}$. An ingress point can be a router interface of n_{dst} that is used exclusively in a private peering link (*e.g.,* with its upstream provider) or a router at an IXP that connects n_{dst} to multiple other networks (*i.e.,* the members of the IXP).

Remark: This notion can be generalized for the case where multiple nodes announce the same prefix (Multi-Origin AS, or MOAS). A virtual node n_{dst} can be connected to these MOAS nodes, which then serve as the ingress points of n_{dst}.

Catchment: mapping nodes to ingress points. Let assume w.l.o.g. that each neighbor j of n_{dst} is directly connected to n_{dst} through exactly one ingress point m, $m \in \mathcal{M}$. We denote this as $j \triangleright m$. Every other node i, $i \in \mathcal{N}$, selects a best path $bp_{i \to n_{dst}}$ towards n_{dst}, *e.g.,*

$$bp_{i \to n_{dst}} = [i, x, ..., y, n_{dst}]$$

where x is a neighbor of i, and y a neighbor of n_{dst}. In this example, if $y \triangleright m$, $m \in \mathcal{M}$, then $bp_{i \to n_{dst}} \triangleright m$ and $i \triangleright m$.

DEFINITION 2 (NODE ROUTE / CATCHMENT).
The <u>route</u> of a node i is m, and is denoted as $i \triangleright m$, when i routes its traffic to n_{dst} through the ingress point m of n_{dst}.
The <u>catchment</u> of an ingress point m is the set of nodes $i \in \mathcal{N}$, for which it holds that $i \triangleright m$.

We would like to stress that the "route" of a node i, as defined in Def. 2 and used throughout the paper, indicates only how the traffic of i enters the network of n_{dst} (*i.e.,* the last hop closest to n_{dst} in $bp_{i \to n_{dst}}$), and not the entire AS-path.

Route probability and Routing function. In many cases we cannot determine which is the best path of a node i, *e.g.,* when the paths $p_{i \to n_{dst}}$ (i) are not known, or (ii) are known but the local preferences are unknown. We capture this uncertainty in a probabilistic way, by defining the *route probability* as:

$$\pi_i(m) = Prob\{bp_{i \to n_{dst}} \triangleright m\}, \ i \in \mathcal{N}, m \in \mathcal{M} \tag{1}$$

Furthermore, we define the *routing function* $f : \mathcal{N} \to \mathcal{M} \cup \{0\}$ that maps nodes ($i \in \mathcal{N}$) to ingress points ($m \in \mathcal{M}$) as:

$$f(i) = \begin{cases} m & \text{, if } \pi_i(m) = 1 \\ 0 & \text{, otherwise} \end{cases} \tag{2}$$

In other words, $f(i) = m \neq 0$ denotes a certainty for the route of node i (and $f(i) = 0$ denotes uncertainty).

2.3 A Sub-Case: the Valley-Free (VF) Model

The network and routing model of Section 2.1 are generic and can describe the BGP setups encountered in practice. Here, we present how the valley-free (VF) routing model [17] can be captured as a special case of our model. The VF model is widely considered in related work as a useful approximation for Internet routing, thus we believe that this section will facilitate other researchers to apply our framework.

In the VF model, each pair of adjacent nodes has either a *customer-to-provider* or a *peer-to-peer* relationship. We denote a relationship between two nodes i, j ($i, j \in \mathcal{N}, e_{ij} \in \mathcal{E}$) as $\ell_{ij} \in \{c2p, p2p, p2c\}$, *e.g.*, $\ell_{ij} = c2p$ when i is a customer of j. Note that when $\ell_{ij} = c2p$ then $\ell_{ji} = p2c$, but $p2p$ relationships are typically symmetric (e.g., settlement-free peering).

Under the VF model, a node i prefers paths received from customers to paths from peers or providers, and paths from peers to paths from providers. We denote this path preference as $p2c > p2p > c2p$ and we can capture this in our model by assigning local preferences as follows:

$$q_{ij} > q_{ik} \Leftrightarrow \ell_{ij} > \ell_{ik} \tag{3}$$

Moreover, when a node has a best path for n_{dst} through a customer, it advertises this path to all its neighbors (customers, peers, providers); and when the best path is through a peer or provider, it advertises this path only to its customers:

$$h_{ijk} = \begin{cases} 1 & , \text{ if } \ell_{ik} = p2c \text{ or } \ell_{ij} = p2c \\ 0 & , \text{ otherwise} \end{cases} \tag{4}$$

It is worth noting that in practice, only coarse estimates of the AS-relationships ℓ_{ij} are known (*e.g.*, CAIDA AS-relationship dataset [5]), while the detailed local preferences q_{ij} are typically not made public by networks. Hence, it is commonly assumed that $q_{ij} = q_{ik} \Leftrightarrow \ell_{ij} = \ell_{ik}$, *i.e.*, a network assigns equal local preferences to all neighbors of the same type [4].

3 ROUTE INFERENCE

The problem. Our goal is to infer through which ingress point each node i reaches the destination node n_{dst} (or, equivalently, the *route* of each node / the catchment of each ingress point). In this section, we tackle this problem, and provide methods for the route inference. Our methodology is summarized as follows.

Methodology overview. We first calculate for every node $i \in \mathcal{N}$ all its eligible paths to n_{dst} (see Def. 1), and encode them in a directed acyclic graph (DAG) rooted in n_{dst}; we call this graph the *Routing Graph* or *R-graph* (Section 3.1). The R-graph is the basic structure, on which our inference methodology is built.

Proceeding to inference, we first focus on the nodes for which a *certain* inference can be made (Section 3.2); *our goal is to calculate $f(i)$, $\forall i \in \mathcal{N}$.* We infer the values of the routing function f based on the structure of the R-graph; when i has only one eligible path $p_{i \to n_{dst}}$, and this path is through the ingress point m, then $f(i) = m$. However, and most importantly, the R-graph enables to determine non-zero values of $f(i)$ (*i.e.*, certain inference) also for some nodes that have multiple eligible paths; even without knowing which of them is the best path, or enumerating all of them.

We then focus on nodes with uncertain routes, *i.e.*, for $i \in \mathcal{N}$ with $f(i) = 0$, and present a framework and methodology for *probabilistic* inference of routes (Section 3.3). We calculate the route probabilities $\pi_i(m)$ for all nodes $i \in \mathcal{N}$ and ingress points $m \in \mathcal{M}$.

Table 2. Inference Methodology Overview.

Type of Inference				Methodology
Certain	Probabilistic	Oracles	Shortest path preference	Sequence of steps / algorithms. (*BEL: any exact or approximate *belief updating* algorithm [24])
✓			(✓)	Alg.1 $\Rightarrow$ (Alg.5 $\Rightarrow$) Alg.2
	✓		(✓)	Alg.1 $\Rightarrow$ (Alg.5 $\Rightarrow$) Alg.2 $\Rightarrow$ Alg.3
✓		✓	(✓)	Alg.1 $\Rightarrow$ (Alg.5 $\Rightarrow$) Alg.2 $\Rightarrow$ Alg.3 $\Rightarrow$ Alg.4
	✓	✓	(✓)	Alg.1 $\Rightarrow$ (Alg.5 $\Rightarrow$) Alg.2 $\Rightarrow$ Alg.3 $\Rightarrow$ BEL

Next, we study how to enhance (certain or probabilistic) inference, when *oracles* (*e.g., measurements*) are given for a set of nodes with uncertain routes (Section 3.4).

Finally, we consider the case where nodes *prefer shorter paths* that conform to their routing policies (this frequently holds in practice [1, 10]), and incorporate this preference in our framework by modifying the R-graph; this enables route inference for more nodes (Section 3.5).

The aforementioned inference methods (certain, probabilistic, with oracles) can be used independently or complementarily. Table 2 gives an overview of the inference methodology, namely, the sequence of steps (algorithms) needed for applying the different route inference variants proposed in this paper.

Comparison to simulation models. Simulation-based approaches [15, 17, 34, 40] return a single outcome of catchment each time. Running a simulation more than once, may give different outcomes, since simulators typically employ randomization to determine the best path when not sufficient knowledge (*e.g.,* the $Q, \mathcal{H}$ or tie-breaker values) is available. For example, let the outcome for nodes i and j of the first (denoted in the superscript) simulation run be $bp_{i \to n_{dst}}^{(1)} \rhd m1$ and $bp_{j \to n_{dst}}^{(1)} \rhd m1$, and of the second run be $bp_{i \to n_{dst}}^{(2)} \rhd m1$ and $bp_{j \to n_{dst}}^{(2)} \rhd m2$. Based solely on these outcomes, one cannot answer the following questions: *What would be the outcome of a third run? Will i always route to m1, or did it happen twice due to random tie-breaking? Which route (m1 or m2) is more probable for j, if we simulate all possible tie-breaking combinations?*

Our methodology provides answers to these questions (and with low complexity algorithms), where simulation-based models would need several simulation runs (of much higher complexity) to provide only an approximate answer. For instance, the certain inference algorithm (Section 3.2) infers whether i will always route to $m1$, and the probabilistic inference algorithm (Section 3.3) calculates the percentage of all possible outcomes in which j will route to $m1$.

3.1 Building the R-graph

We design Algorithm 1 to build the R-graph $\mathcal{G}_R$ that encodes all eligible paths to n_{dst}. Any eligible path $p_{i \to n_{dst}}, \forall i \in \mathcal{N}$, can be extracted by processing $\mathcal{G}_R$. Figure 1 shows an example of a R-graph rooted in n_{dst}.

Input/Output. Algorithm 1 receives as input a network graph and its routing policies $\mathcal{G}(\mathcal{N}, \mathcal{E}, Q, \mathcal{H})$, and a destination node $n_{dst} \in \mathcal{N}$. It returns as output the R-graph $\mathcal{G}_R(\mathcal{N}_R, \mathcal{E}_R)$, which is a DAG rooted in node n_{dst}.

Workflow. First, Algorithm 1 simulates the operation of BGP; when needed, "ties" are broken randomly, *e.g.,* if multiple paths from neighbors with equal local preferences exist, one of them is selected randomly as the best path. This randomness *does not affect* the construction of the R-graph, since all incoming path advertisements exist in the RIB of a node i ($\mathcal{P}_i$, returned in *line 1*), and are
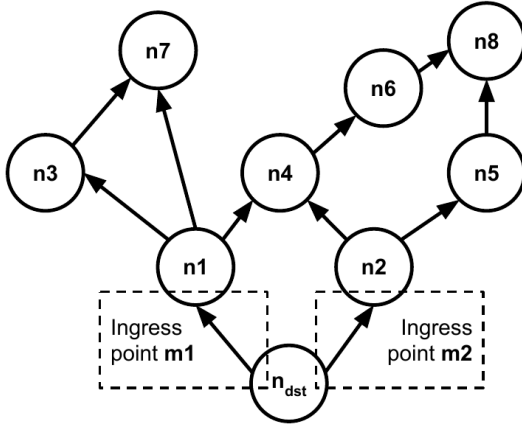
Node	Eligible path(s)	f(n)
$n1$	$[n1, n_{dst}]$	$m1$
$n2$	$[n2, n_{dst}]$	$m2$
$n3$	$[n3, n1, n_{dst}]$	$m1$
$n4$	$[n4, n1, n_{dst}]$	0
	$[n4, n2, n_{dst}]$	
$n5$	$[n5, n2, n_{dst}]$	$m2$
$n6$	$[n6, n4, n1, n_{dst}]$	0
	$[n6, n4, n2, n_{dst}]$	
$n7$	$[n7, n1, n_{dst}]$	$m1$
	$[n7, n3, n1, n_{dst}]$	
$n8$	$[n8, n5, n2, n_{dst}]$	0
	$[n8, n6, n4, n1, n_{dst}]$	
	$[n8, n6, n4, n2, n_{dst}]$	

Fig. 1. Example of a R-graph (left), and the corresponding eligible paths and values of the routing function f returned by Alg. 2 for every node (right). The destination node n_{dst} has two ingress points $m1$ and $m2$ through which it connects to its neighbors $n1$ and $n2$, respectively.

taken into account (loop in *line 6*). Then, it initializes the R-graph by adding only the nodes, without adding any edge (*line 2*). For each node i (*lines 3–18*), it accesses its RIB $\mathcal{P}_i$ and finds all neighbors that advertised a path for n_{dst} (*i.e.,* the next-to-i hops in the RIB paths; *line 7*), and selects the set of the neighbors (*best_neighbors*) with the highest local preference (*max_q*). The paths from these neighbors are the eligible paths of i, since they (a) exist in the RIB and (b) are from neighbors with the highest local preference (as requested by Def. 1). For each neighbor k of i in *best_neighbors*, it adds a directed edge from k to i.

Complexity: $O(|\mathcal{N}| \cdot |\mathcal{E}|)$. The computational complexity of Algorithm 1 is dominated by the complexity of running a BGP simulation (*line 1*) which is equivalent to this of the centralized Bellman-Ford algorithm $O(|\mathcal{N}| \cdot |\mathcal{E}|)$. The loop in *lines 3–18* examines every edge in the graph at most once and runs in $O(|\mathcal{E}|)$.

The following theorem formally states that (i) any path in the R-graph is eligible and (ii) any eligible path is encoded in the R-graph.

THEOREM 1. *A path $p_{i \to n_{dst}}$ is an eligible path if and only if it can be constructed by starting from n_{dst} and following a sequence of directed edges in R-graph $\mathcal{G}_R$ until reaching i.*

PROOF. The proof is given in Appendix A.   □

3.2 Route Inference on the R-Graph

We proceed to infer through which ingress point a node i routes its traffic to n_{dst}, by exploiting the structure of the R-graph. We demonstrate this inference using the example of Fig. 1, where it is given that $n1 \triangleright m1$ and $n2 \triangleright m2$ (*i.e.,* $f(n1) = m1$ and $f(n2) = m2$).

Case A: When the best path is known, the route inference is straightforward (from Eq. (2)). Node $n3$ has only one way/path to reach n_{dst} (*i.e.,* by following links in the R-graph; see Theorem 1). This path is through node $n1$, and since $f(n1) = m1$, it follows that $f(n3) = f(n1) = m1$.

Case B: Route inference is possible, even when the best path cannot be determined. Node $n7$ has two incoming links from nodes $n1$ and $n3$; it selects only one of them to form its best path, based on its local preferences to $n1$ and $n3$. Without knowing these local preferences, we cannot infer the best path. However, since both $n1$ and $n3$ route traffic through the same ingress

---

Algorithm 1 Building the R-graph.

Input: Network graph $\mathcal{G}(\mathcal{N}, \mathcal{E}, Q, \mathcal{H})$; destination node n_{dst}.
1: $\mathcal{P} \leftarrow$ RunBGP_RandomTieBreak$(\mathcal{G}(\mathcal{N}, \mathcal{E}, Q, \mathcal{H}), n_{dst})$
                                                  /* $\mathcal{P} = \{\mathcal{P}_i : i \in \mathcal{N}\}$, where $\mathcal{P}_i$ is the BGP RIB of i */
2: $\mathcal{G}_R(\mathcal{N}_R, \mathcal{E}_R) : \mathcal{N}_R \leftarrow \mathcal{N}; \mathcal{E}_R \leftarrow \varnothing$
3: **for** $i \in \mathcal{N}$ **do**
4:     $best\_neighbors \leftarrow \varnothing$
5:     $max\_q \leftarrow -\infty$
6:     **for** $p_{i \rightarrow n_{dst}} \in \mathcal{P}_i$ **do**
7:         $k \leftarrow$ GetNeighbor$(i, p_{i \rightarrow n_{dst}})$
8:         **if** $q_{ik} < max\_q$ **then**
9:             **continue**
10:        **else if** $q_{ik} > max\_q$ **then**
11:            $best\_neighbors \leftarrow \{k\}$
12:            $max\_q \leftarrow q_{ik}$
13:        **else**   /* $q_{ik} = max\_q$ */
14:            $best\_neighbors \leftarrow best\_neighbors \cup \{k\}$
15:        **end if**
16:     **end for**
17:     $\mathcal{E}_R \leftarrow \mathcal{E}_R \cup \{e_{ki} : k \in best\_neighbors\}$
18: **end for**
19: **return** $\mathcal{G}_R(\mathcal{N}_R, \mathcal{E}_R)$

---

point ($f(n1) = f(n3) = m1$), selecting either path leads to the same value of the routing function: $f(n7) = m1$.

Case C: Route inference might not be possible for some nodes. On the contrary to $n7$, while node $n4$ has also two incoming links, they are from nodes $n1$ and $n2$ for which it holds that $f(n1) \neq f(n2)$. Thus, in this case we cannot infer which path will be selected, and we write $f(n4) = 0$.

The above rules can be applied sequentially for all nodes in the R-graph. Algorithm 2 formalizes this inference process.

Input/Output. Algorithm 2 receives as input a R-graph, a destination node n_{dst} (root of the graph), and a mapping of the neighbors of n_{dst} to its ingress points $\mathcal{M}$. It returns the values of the routing function f for all nodes in the R-graph.

Workflow. Algorithm 2 starts from the neighbors of n_{dst} and sets the values of f according to their mapping to ingress points (*lines 2–7*). Then, it calculates a topological ordering[3] of the R-graph nodes (*line 8*) and sequentially visits nodes starting from those that are closer to the n_{dst} (*lines 9–20*). For each node i, it calculates the set of *routes* (Def. 2) of its parent nodes CR_i (*lines 10–14*), which are the candidate routes for node i. If some of the parents do not have a certain route ($0 \in CR_i$) or there are more than one candidate routes ($|CR_i| \neq 1$), then it cannot make a certain route inference for node i, and sets $f(i) = 0$ (*lines 15–16*). Otherwise (*i.e.*, there is only one candidate route for i), an inference is made and the route of i is set equal to this of its parent(s) (*line 18*).

Remark: Visiting nodes in their topological order ensures correctness of the algorithm, *i.e.*, that the routing function of a node i will not be mis-inferred (*e.g.*, $f(i) = 0$ instead of $f(i) = m$, $m \in \mathcal{M}$).

---

[3] A topological sort/ordering $\mathcal{T}$ of a directed graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$ is a linear ordering of its nodes $\mathcal{N}$ such that for every directed edge $e_{ij} \in \mathcal{E}$ from node $i \in \mathcal{N}$ to node $j \in \mathcal{N}$, i comes before j in the sort/ordering $\mathcal{T}$. For example, in Fig. 1, node numbering ($n1$, ..., $n8$) corresponds to a topological ordering.

Algorithm 2 Inference on the R-graph.

Input: R-graph $\mathcal{G}_R(\mathcal{N}_R, \mathcal{E}_R)$; destination node n_{dst}; mapping ($\rhd$) of the neighbors of n_{dst} to ingress points $\mathcal{M}$.

1: $f(i) \leftarrow 0, \ \forall i \in \mathcal{N}_R$ /* Initialization */
2: $dst\_neighbors \leftarrow \{i \in \mathcal{N}_R : e_{n_{dst}, i} \in \mathcal{E}_R\}$
3: **for** $i \in dst\_neighbors$, $m \in \mathcal{M}$ **do**
4:      **if** $i \rhd m$ **then**
5:          $f(i) \leftarrow m$
6:      **end if**
7: **end for**
8: $\mathcal{T} \leftarrow \textsc{TopologicalSort}(\mathcal{G}_R)$
9: **for** $i \in \mathcal{T} \setminus \{n_{dst} \cup dst\_neighbors\}$ **do**
10:     $CR_i \leftarrow \varnothing$ /* Candidate routes for node i */
11:     $P_i \leftarrow \{j \in \mathcal{N}_R : e_{ji} \in \mathcal{E}_R\}$ /* Parents of i */
12:     **for** $j \in P_i$ **do**
13:         $CR_i \leftarrow CR_i \cup \{f(j)\}$
14:     **end for**
15:     **if** $(0 \in CR_i)$ **or** $(|CR_i| \neq 1)$ **then**
16:         $f(i) \leftarrow 0$
17:     **else**
18:         $f(i) \leftarrow CR_i$
19:     **end if**
20: **end for**
21: **return** $f(i), \ \forall i \in \mathcal{N}_R$

This is because all parent nodes of i, which are the only nodes that affect the route of this node, will have been visited before node i.

Complexity: $O(|\mathcal{N}_R| + |\mathcal{E}_R|)$. The topological sort in *line 8* is of complexity $O(|\mathcal{N}_R| + |\mathcal{E}_R|)$ and the loop in *lines 9–20* is of complexity $O(|\mathcal{E}_R|)$ since it visits each edge in $\mathcal{E}_R$ exactly once.

3.3 Probabilistic Route Inference

The goal of probabilistic inference is to calculate the route probabilities $\pi_i(m)$ (defined in Eq. (1)). Hence, even for nodes for which a certain inference is not possible, the *probabilities $\pi_i(m)$ can provide extra information* that can be useful, *e.g.,* to predict the total load per ingress point by taking the expectation over the route probabilities:

$$Traffic\_Load(m) = \sum_{i \in \mathcal{N}} T_i \cdot \pi_i(m) \tag{5}$$

where T_i is the known traffic load from i to n_{dst} (T_i can be estimated independently of the deployment/routing setup, *e.g.,* from Netflow statistics or similarly to the system proposed in [13]).

The R-graph as a Bayesian Network (BN). To proceed to probabilistic route inference, we handle the R-graph as a Bayesian network (BN)[4], where a node i can take a value $m \in \mathcal{M}$, and the respective probability is given by $\pi_i(m)$. Based on BN properties (and the *causality* in the R-graph , *i.e.,* children nodes select routes learned from their parents and not the opposite), the following expression can be used to calculate the probabilities $\pi_i(m)$, from the probabilities of the parents

---

[4]A BN is a directed acyclic graph (DAG), where a directed edge e_{ij} denotes a dependence of node j on node i [24]. We remind that the R-graph is a DAG that encodes routing path dependencies; *e.g.,* a directed edge e_{ij} denotes that node i is the next hop of j in a path $p_{j \rightarrow n_{dst}}$ from j to n_{dst}.

Algorithm 3 Probabilistic route inference on the R-graph.

Input: R-graph $\mathcal{G}_R(\mathcal{N}_R, \mathcal{E}_R)$; ingress points $\mathcal{M}$; routing function $f(i)$, $\forall i \in \mathcal{N}_R$; probabilities p_{ij}, $\forall e_{ji} \in \mathcal{E}_R$.

1: $\pi_i(m) \leftarrow 0$, $\forall i \in \mathcal{N}_R, m \in \mathcal{M}$ /* Initialization */
2: $\mathcal{T} \leftarrow$ TopologicalSort$(\mathcal{G}_R)$
3: **for** $i \in \mathcal{T}$ **do**
4:     **if** $f(i) \neq 0$ **then**
5:         $\pi_i(f(i)) \leftarrow 1$
6:     **else**
7:         $P_i \leftarrow \{j \in \mathcal{N}_R : e_{ji} \in \mathcal{E}_R\}$ /* Parents of i */
8:         **for** $j \in P_i$ **do**
9:             **for** $m \in \mathcal{M}$ **do**
10:                 $\pi_i(m) \leftarrow \pi_i(m) + \pi_j(m) \cdot p_{ij}$
11:             **end for**
12:         **end for**
13:     **end if**
14: **end for**
15: **return** $\pi_i(m)$, $\forall i \in \mathcal{N}_R, m \in \mathcal{M}$

(P_i) of i:

$$\pi_i(m) = \sum_{j \in P_i} \pi_j(m) \cdot p_{ij} \tag{6}$$

where p_{ij} the probability for i to prefer a path from j than any other parent node, and $\sum_{j \in P_i} p_{ij} = 1$.

Algorithm 3 applies the above equation and calculates the probabilistic route inference on a R-graph.

Input/Output. Algorithm 3 receives as input the R-graph, the ingress points, the values of the routing function and the probabilities p_{ij}, and returns the route probabilities $\pi_i(m)$, $\forall i \in \mathcal{N}, m \in \mathcal{M}$.

Workflow. Algorithm 3 initializes all probabilities to zero (*line 1*) and starts visiting all nodes according to a topological sort (*lines 2–14*). If a visited node i has a certain route m, then it sets the probability $\pi_i(m)$ equal to 1 (*lines 4–5*). Otherwise, it applies Eq. (6) to calculate $\pi_i(m)$ from the probabilities of the parent nodes (*lines 7–13*). Visiting nodes in a topological order satisfies that the probability of all parent nodes P_i will have been calculated before visiting i.

Complexity: $O(|\mathcal{N}_R| + |\mathcal{E}_R|)$. Similarly to the certain inference methodology, the complexity of the topological sort in *line 2* is $O(|\mathcal{N}_R| + |\mathcal{E}_R|)$, and this of the loop in *lines 3–14* is $O(|\mathcal{E}_R|)$. However, Algorithm 3 is used with Algorithm 2 (see Table 2), which means that the topological sort is already calculated in Algorithm 2 and can be passed as input to Algorithm 3.

Setting the values of the probabilities p_{ij}. Algorithm 3 and Eq. (6), require the probabilities p_{ij} to be known. We stress that these probabilities are not the local preferences q_{ij} (which are equal for all the parents of a node in the R-graph; cf. Algorithm 1), but other criteria based on which a node will break ties, such as, the router IP address or the time of the received BGP announcements [50]. In some cases, these criteria (and the respective probabilities) can be inferred from past measurements,*e.g.,* [34]. However, given no prior knowledge on the criteria or in the case where the tie-breaker values change over time, the probabilities can be set to equal values (uniformly) for all parents in the R-graph, *i.e.,* $p_{ij} = \frac{1}{|P_i|}$, $\forall j \in P_i$.

3.4 Inference under Oracles

We proceed to study how to enhance the certain or probabilistic route inference, when an "oracle" for the value of the routing function for a set of nodes X, $X \subset \mathcal{N}$, with previously uncertain routes

($f(i) = 0, \forall i \in X$), is given. Obviously, the values of f for nodes in X are trivially inferred (from the oracle). However, here we show that *an oracle for the routing function for a set of nodes X, enables route inference for a –potentially– larger set of nodes Y, $Y \supseteq X$.*

"Oracles" in reality. In practice, an "oracle" can be obtained by a measurement, such as BGP messages/RIBs collected at some node, *e.g.,* through a route collector [48] (passive measurement), or traceroutes/pings (see, *e.g.,* [13]) from a node towards the destination node n_{dst} (active measurement). In the remainder, we consider oracles in the context of a measurement, however, our methodology is valid in the general case, independently of how the oracle is obtained.

Remark: Actual measurements are applicable only in the case of an existing deployment, where a destination node n_{dst} has already established connections and announces prefixes to its neighbors. The measurement-enhanced inference can then be useful for lightweight route inference, *e.g.,* with only a few, instead of exhaustive [13], measurements. However, the oracle-enhanced inference techniques can be useful for planning purposes (hypothetical scenarios) as well, *e.g.,* identifying the optimal locations for installing monitoring equipment to efficiently monitor future deployments and routing configurations (see, *e.g.,* Section 4).

We use again the example of Fig. 1 to demonstrate the measurement-enhanced inference methodology. The basic inference methodology (Sections 3.2 and 3.3) cannot infer with certainty the values f for nodes $n4$, $n6$, and $n8$ (see right column of the table in Fig. 1). By conducting measurements for some of these nodes, the following cases of route inference are possible.

Case A: The routes of the measured nodes are directly inferred. When we measure a node i, we either learn its best path (*e.g.,* from BGP data, traceroutes) or through which ingress point m it routes traffic to n_{dst} (pings [13]). In both cases, we can directly infer $f(i)$.

Case B: The routes of the children of measured nodes might be inferred. If node $n4$ is measured, then the route of $n6$ can be directly determined as well, since the eligible paths for $n6$ are through $n4$, and thus it must hold $f(n6) = f(n4)$. However, if $n6$ is measured, it is not always possible to infer the route of $n8$ as well: if $f(n6) = m2 = f(n5)$, then we can infer $f(n8) = m2$, whereas if $f(n6) = m1 \neq f(n5)$, then we cannot infer with certainty the route of $n8$.

Case C: The routes of the parents of measured nodes might be inferred. If $n6$ is measured, then we can directly infer the route for $n4$ (since, as discussed above, it must hold $f(n6) = f(n4)$). If $n8$ is measured there are two cases: (i) if $f(n8) = m1$, then, since $f(n5) = m2$ (see Fig. 1), we can infer that $n8$ selects its best path through $n6$ and thus $f(n6) = f(n8)$; (ii) if $f(n8) = m2$, then we cannot infer with certainty through which node is the best path of $n8$, and, in contrast to the previous case, we cannot infer $f(n6)$.

Algorithm 4 is based on the aforementioned guidelines to enhance the route inference in a R-graph, given a set of oracles.

Input/Output. Algorithm 4 receives as input a R-graph, the ingress points, the values of the routing function f and the probabilities π (which are calculated by Algorithms 2 and 3, respectively), and a set of oracles that map nodes to ingress points. It returns the updated values of the routing function f.

Workflow. For each node $i \in X$ for which an oracle is provided, Algorithm 4 calls the function SetRoute, which updates the routing function f and probabilities π (*lines 1–5*). Specifically, SetRoute sets the value of the routing function equal to the one of the provided oracle (*line 8*), and updates the probabilities for node i (*lines 9–10*). Then, it finds the subset CP_i of the parent nodes P_i of i, which may route (or actually route) through the same ingress point with i (*lines 11–12*). These are the candidate nodes that can be in the best path $bp_{i \rightarrow n_{dst}}$. If there is only one such candidate parent node ($|CP_i| = 1$), then with certainty this node has the same route with i. Hence, in case the

---

Algorithm 4 Enhancing inference with measurements.

Input: R-graph $\mathcal{G}_R(\mathcal{N}_R, \mathcal{E}_R)$; ingress points $\mathcal{M}$; routing function f; probabilities π; set of measured nodes $\mathcal{X}$ and their mapping ($\triangleright$) to ingress points.

1: **for** $i \in \mathcal{X}, \; m \in \mathcal{M}$ **do**
2:      **if** $i \triangleright m$ **then**
3:          $(f, \pi) \leftarrow$ SETROUTE$(i, m, f, \pi, \mathcal{G}_R)$
4:      **end if**
5: **end for**
6: **return** f

7: **function** SETROUTE$(i, m, f, \pi, \mathcal{G}_R)$
8:      $f(i) \leftarrow m$
9:      $\pi_i(m) \leftarrow 1$
10:     $\pi_i(d) \leftarrow 0, \; \forall d \neq m$
11:     $P_i \leftarrow \{j \in \mathcal{N}_R : e_{ji} \in \mathcal{E}_R\}$ /* *Parents of i* */
12:     $CP_i \leftarrow \{j \in P_i : \pi_j(m) > 0\}$ /* *Candidate parents* */
13:     **if** $|CP_i| = 1$ **and** $f(CP_i) = 0$ **then**
14:        $(f, \pi) \leftarrow$ SETROUTE$(CP_i, m, f, \pi, \mathcal{G}_R)$
15:     **end if**
16:     $C_i \leftarrow \{j \in \mathcal{N}_R : e_{ij} \in \mathcal{E}_R, f(j) = 0\}$ /* *Children of i without inferred route* */
17:     **for** $j \in C_i$ **do**
18:        $P_j \leftarrow \{k \in \mathcal{N}_R : e_{kj} \in \mathcal{E}_R\}$
19:        $CR_j \leftarrow \varnothing$ /* *Candidate routes for j* */
20:        **for** $k \in P_j$ **do**
21:           $CR_j \leftarrow CR_j \cup \{f(k)\}$
22:        **end for**
23:        **if** $|CR_j| = 1$ **and** $CR_j \neq 0$ **then**
24:           $(f, \pi) \leftarrow$ SETROUTE$(j, m, f, \pi, \mathcal{G}_R)$
25:        **end if**
26:     **end for**
27:     **return** (f, π)
28: **end function**

---

route for this node is not already inferred ($f(CP_i) = 0$), there is a new inference for this node and SETROUTE is called. After making the inferences for the parents of i (*lines 13–15*), the algorithm proceeds to inference for the children nodes of i (*lines 16–26*). For each child j without an inferred route (*line 16*), it collects the distinct values of the routing function of its parents P_j (*lines 18–22*). If there is only one such value CR_j, and $CR_j \neq 0$, then it means that all the parent nodes of j route traffic to CR_j (in fact, in this case it holds that $CR_j \equiv m$). Thus an inference for the route of j is possible, and SETROUTE is called. Finally, SETROUTE returns the updated f and π.

Complexity: $O(|\mathcal{N}_\mathcal{R}|)$. The method SETROUTE is called at most once per node (even if called recursively), *i.e.*, up to $|\mathcal{N}_\mathcal{R}|$ times; for more details see Theorem 2 and its proof in Appendix C.

Problem properties and complexity. As discussed in Section 3.3, the R-graph is a BN. When an oracle is given, the probabilities in this BN can be updated to infer extra routes. However, updating exactly the probabilities π is NP-hard (Lemma 1), since the R-graph is a multiply-connected BN (and not a *polytree*) [11]. However, efficient algorithms to *approximate* the updated probabilities π exist [24].

LEMMA 1. *Updating the probabilities π in the R-graph to their new values π' when an oracle is given, is NP-hard.*

PROOF. The proof is given in Appendix B . □

Algorithm 4 is based on BN belief propagation methods [24]. The main difference is that it does not aim to update exactly all the probabilities π, but only the probabilities whose new value π' is either 1 or 0. This is sufficient for a certain route inference (for the nodes for which this is possible), and can take place in polynomial time, as Theorem 2 states.

THEOREM 2. *Algorithm 4 updates the probabilities π for all nodes i for which $\max_m \pi'_i(m) = 1$ holds, in polynomial time $O(\mathcal{N}_R)$.*

PROOF. The proof is given in Appendix C. □

3.5 Preference of Shorter Paths

The R-graph encodes all eligible paths, given the set of local preferences Q. In practice, a node commonly *prefers the shortest (in terms of AS-hops) among the paths learned from neighbors of equal local preference* (*i.e.,* its parents in the R-graph) [10]. This common behavior is widely considered in related work as well, *e.g.,* [1, 19, 40]. Hence, route inference under the assumption of shortest path preference is relevant to real network operations.

Here, we show how to incorporate the shortest path preference in our methodology. We do this in Algorithm 5, by modifying the R-graph to eliminate the eligible paths that are always longer and thus never preferred by a node. Specifically, assuming preference of shorter paths, means that not all the paths in the R-graph are eligible anymore. For example, in the R-graph of Fig. 1, node $n7$ has two paths; however, the path through $n1$ is shorter and preferred. The path through $n3$ is not eligible anymore, and thus the edge between $n3$ and $n7$ must be removed.

Input/Output. Algorithm 5 receives as input the R-graph, modifies it, and returns the modified R-graph.

Workflow. A minimum length (of eligible paths) L_i is set for each node i, and is initialized to 0 for n_{dst}, and to ∞ for every other node (*line 1*). L_i denotes the minimum length of the eligible paths $p_{i \to n_{dst}}$. A node will prefer the shorter paths, and thus the objective is to remove the longer paths of a node from the R-graph. To this end, starting from nodes closer to n_{dst} and following a topological sort, the set of parents P_i of the node i is calculated, and the value of L_i is set equal to the minimum value L_j, $j \in P_i$, plus one (*lines 3–7*). The parents that have longer paths to n_{dst} will never be preferred by a node i. Hence, the incoming edges to i from such parents are removed from the R-graph (*line 8*).

Complexity: $O(|\mathcal{N}_R| + |\mathcal{E}_R|)$. The complexity of the topological sort in *line 2* is $O(|\mathcal{N}_R| + |\mathcal{E}_R|)$, and this of the loop in *lines 3–9* is $O(|\mathcal{E}_R|)$. Similarly, Algorithm 5 is used with Algorithm 2 (see Table 2), which means that the topological sort is calculated only once.

THEOREM 3. *Applying Algorithm 5 on a R-graph, can only increase (not decrease) the set of nodes with certain routes.*

PROOF. We provide a sketch of the proof in Appendix D. □

4 USE CASE: EFFICIENT MEASUREMENTS

In this section, we investigate how to efficiently select measurements in order to increase the (certain) inference under a routing configuration. Specifically, we consider the following problem.

The problem. *Given a budget of B measurements, what is the optimal set of nodes to be measured that maximizes the (certain) route inference in the R-graph?*

Algorithm 5 R-graph transformation for shortest path preference.

Input: R-graph $\mathcal{G}_R(\mathcal{N}_R, \mathcal{E}_R)$; destination node n_{dst}.
Ouput: (updated) R-graph $\mathcal{G}_R(\mathcal{N}_R, \mathcal{E}_R)$.
1: $L_{n_{dst}} \leftarrow 0; L_i \leftarrow \infty, \ \forall i \in \mathcal{N}_R \backslash \{n_{dst}\}$ /* Initialization */
2: $\mathcal{T} \leftarrow$ TOPOLOGICAL_SORT$(\mathcal{G}_R)$
3: **for** $i \in \mathcal{T} \backslash \{n_{dst}\}$ **do**
4: $\quad P_i \leftarrow \{j \in \mathcal{N}_R : e_{ji} \in \mathcal{E}_R\}$
5: $\quad$ **for** $j \in P_i$ **do**
6: $\quad\quad L_i \leftarrow \min\{L_i, L_j + 1\}$
7: $\quad$ **end for**
8: $\quad \mathcal{E}_R \leftarrow \mathcal{E}_R - \{e_{ji} \in \mathcal{E}_R : L_j + 1 > L_i\}$
9: **end for**
10: **return** $\mathcal{G}_R(\mathcal{N}_R, \mathcal{E}_R)$

The above problem may emerge in the context of a number of measurement-related applications in the Internet, such as how to efficiently select a set of vantage points from which to trigger data-plane measurements (*e.g.,* select the best set of RIPE Atlas probes [42], given a limit on measurement credits), or how to optimally deploy monitoring infrastructure for passive (*e.g.,* route collectors) or active (*e.g.,* probes) measurements.

In the remainder, we study this problem: in Section 4.1 we show that it is hard to be solved exactly or even approximated (since it requires exponential –to the number of nodes– complexity), and in Section 4.2 we propose a greedy algorithm for efficient measurement selection, leveraging the R-graph's structure and properties.

4.1 Problem Formulation and Properties

Problem formulation. Let $\mathcal{X}, \mathcal{X} \subseteq \mathcal{N}_R$, be a set of nodes for which we have an oracle (i.e., route measurement), and let $x, x \in \mathcal{M}^{|\mathcal{X}|}$, the routes of nodes in $\mathcal{X}$ (*i.e.,* x is a vector of size $|\mathcal{X}|$, taking values in state space $\mathcal{M}^{|\mathcal{X}|}$). We will denote $\mathcal{X} \triangleright x$. For example, if $\mathcal{X}$ consists of three nodes $\{n1, n2, n3\}$, which route to ingress points $\{m1, m2\}$ as follows: $n1 \triangleright m1, n2 \triangleright m1, n3 \triangleright m2$, then we denote $x = \{m1, m1, m2\}$.

Given a set $\mathcal{X}$ and its routes x, we denote as $\mathcal{NC}_R(\mathcal{X} \triangleright x)$ the *number of nodes with a certain route* given these oracles:

$$\mathcal{NC}_R(\mathcal{X} \triangleright x) = | \{i \in \mathcal{N}_R : f(i) \neq 0 | \mathcal{X} \triangleright x\} | \tag{7}$$

Note that we cannot know through which ingress point each measured node routes its traffic before conducting a measurement. Hence, to evaluate the effectiveness of selecting a set of nodes, we consider all the possible measurement outcomes $x, x \in \mathcal{M}^{|\mathcal{X}|}$. To this end, we denote the *expected number of nodes with a certain route*, under a set of measured nodes $\mathcal{X}$ as:

$$E_P\left[\mathcal{NC}_R(\mathcal{X})\right] = \sum_{x \in \mathcal{M}^{|\mathcal{X}|}} \mathcal{NC}_R(\mathcal{X} \triangleright x) \cdot P(\mathcal{X} \triangleright x) \tag{8}$$

where $P(\mathcal{X} \triangleright x)$ denotes the probability of realization of the measurements outcome x.

Then, given a budget of at most B measurements, and a set $\mathcal{Y}, \mathcal{Y} \subseteq \mathcal{N}_R$ of nodes which can be measured (e.g., for measurements with RIPE Atlas, $\mathcal{Y}$ can be the set of ASes that host at least one probe), the optimization problem can be expressed as[5]:

---

[5]Generalizations of the problem can be expressed as well, *e.g.,* by weighting with w_i (*e.g.,* based on the incoming traffic load from i) the importance of knowing the route of each node i and modifying the definition of the objective function in Eq. (7)

PROBLEM 1.　　$\max_{\mathcal{X} \subseteq \mathcal{Y}} E_P \left[\mathcal{N}C_R(\mathcal{X}) \right], \quad s.t. \, |\mathcal{X}| \leq B$

Modularity of the objective and the greedy algorithm. Problem 1 belongs to the class of combinatorial problems of maximizing a set function under a cardinality constraint. Lemma 2 summarizes the properties of the objective function of Problem 1, which allow us to characterize its complexity and approximability.

LEMMA 2. *The objective function of Problem 1 is (i) non-negative and monotone, (ii) non-submodular, (iii) non-supermodular.*

PROOF. The proof is given in Appendix E. □

On the one hand, if the objective function of Problem 1 was *submodular*, then applying a greedy algorithm, of polynomial to $\mathcal{N}_R$ number of evaluations of the objective function $E_P \left[\mathcal{N}C_R(\mathcal{X}) \right]$, would come with an approximation guarantee of $1 - 1/e$ of the optimal solution [25]. On the other hand, if it was *supermodular*, then the problem would be NP-hard to approximate [25][6]. However, in the generic case of the R-graph we consider, with a monotone neither submodular, nor supermodular, objective, it has been recently shown that applying a greedy algorithm still comes with approximation guarantees (however, worse than in the case of a submodular function $1 - 1/e$) and usually the performance in practice is not far from the optimal [3]. Therefore, in the following, we design a greedy algorithm for Problem 1, which starts with an empty set $\mathcal{X}^0 = \varnothing$, and at each step k adds to set $\mathcal{X}^{k-1}$ the node that increases the most the expected number of nodes with certain inference, i.e.,

$$\mathcal{X}^k = \mathcal{X}^{k-1} \cup \arg\max_{i \in \mathcal{Y} \setminus \mathcal{X}^{k-1}} E_P \left[\mathcal{N}C_R(\mathcal{X}^{k-1} \cup \{i\}) \right]$$

Remark: The approximation of the greedy algorithm depends on the *submodularity ratio* and *curvature* of the objective function, which in our case is determined by the structure of the R-graph [3]. While deriving approximation guarantees as a function of structure and properties of the R-graph is an interesting research direction, it is out of the scope of this paper, and we defer it to future work.

Complexity in evaluating the objective. A second challenge in solving Problem 1, even with a greedy algorithm, is that the evaluation of the objective function (Eq. (8)) in each step, involves the calculation of the probabilities P, which may require also exponential to $|\mathcal{N}|$ time (see Section 3.4). We demonstrate this with the following example. Let $\mathcal{X}^k$ be the set of the first k nodes selected by the greedy (or, any) algorithm, and a node $j \notin \mathcal{X}^k$. To evaluate the value of the objective function when adding node j to the set of measurements, we need to proceed as follows:

$$E_P \left[\mathcal{N}C_R(\mathcal{X}^k \cup \{j\}) \right] = \sum_x \sum_m \mathcal{N}C_R(\mathcal{X}^k \cup \{j\} \triangleright x \cup m) \cdot P(\mathcal{X}^k \cup \{j\} \triangleright x \cup m)$$

$$= \sum_x \sum_m \mathcal{N}C_R(\mathcal{X}^k \cup \{j\} \triangleright x \cup m) \cdot P(j \triangleright m | \mathcal{X}^k \triangleright x) \cdot P(\mathcal{X}^k \triangleright x)$$

where we applied the Bayes theorem to express the joint probability as a product of the conditional probability.

In the last equation, we can calculate the terms $\mathcal{N}C_R(\mathcal{X}^k \cup \{j\} \triangleright x \cup m)$ using Algorithm 4 (in $O(N)$ steps), and the terms $P(\mathcal{X}^k \triangleright x)$ are already calculated in the $k - 1$ step of the greedy algorithm. The remaining terms $P(j \triangleright m | \mathcal{X}^k \triangleright x)$ correspond to the updated probabilities π_j for

---

as $\mathcal{N}C_R(\mathcal{X} \triangleright x) = \sum_{i \in \{\mathcal{N}_R : f(i) \neq 0 | \mathcal{X} \triangleright x\}} w_i$, and/or assigning different measurement costs c_i per node i by modifying the constraint as $\sum_{i \in \mathcal{X}} c_i \leq B$.

[6]Maximizing a super-modular function is equivalent to minimizing a sub-modular function, which is NP-hard when the size of the set is constrained.

node j, given the set of oracles $\mathcal{X}^k \triangleright x$. As discussed in Section 3.4, the exact calculation of the updated probabilities π is NP-hard.

In the greedy algorithm we propose, we trade accuracy for efficiency in the calculations for π at each step, and update the probabilities π with an approximate ("belief propagation") method.

4.2 A Greedy Algorithm

We present the greedy algorithm we propose for Problem 1, which is built upon the aforementioned guidelines.

Input/Output. Algorithm 6 receives as input a R-graph, the values of the routing function f and the probabilities π, a set of nodes $\mathcal{Y}$ that are eligible to be measured, and a measurement budget B. It returns a set $\mathcal{X}$ of size B, containing the nodes to be measured.

Workflow. After the initialization (*line 1*), Algorithm 6 enters the greedy node selection loop (*lines 2–6*), where at each iteration a node i is added to the set of measured nodes $\mathcal{X}$ (*line 4*). The node that is added is the one that –if measured– increases the most the expected number of nodes with a certain route (*line 3*). The expectation is calculated by Eq. (8) using the probabilities P, i.e.,

$$P(\mathcal{X} \cup \{j\} \triangleright x \cup m) = P(j \triangleright m | \mathcal{X} \triangleright x) \cdot P(\mathcal{X} \triangleright x) = \pi_j^{(\mathcal{X} \triangleright x)}(m) \cdot P(\mathcal{X} \triangleright x) \qquad (9)$$

where we denote $\pi_j^{(\mathcal{X} \triangleright x)}(m) = P(j \triangleright m | \mathcal{X} \triangleright x)$. Note that $\pi_j^{(\varnothing \triangleright x)}(m) = \pi_j(m)$. After adding node i to set $\mathcal{X}$, the probabilities $\pi^{(\mathcal{X} \triangleright x)}$ and $P(\mathcal{X} \triangleright x)$, which will be needed in the next iteration, are calculated using the approximate method UPDATEPROBABILITIES (*line 5*).

The method UPDATEPROBABILITIES calculates the probabilities $P(\mathcal{X} \cup \{j\} \triangleright x \cup m)$ and $\pi_j^{(\mathcal{X} \triangleright x)}(m)$, $\forall$ possible measurement outcomes (*lines 9–15*). The former probabilities are calculated by using Eq. (9) and previous values (*line 10*). The latter probabilities are calculated approximately in *lines 12–14*. First, for the given outcome $\mathcal{X} \cup \{j\} \triangleright x \cup m$, Algorithm 4 is used to calculate the set of nodes with certain inference $\mathcal{Z}$ (*lines 12–13*). We remind that Algorithm 4 (called in *line 11*) is a belief propagation method to update the probabilities of all the nodes with a certain route inference after a measurement/oracle is given. For the remaining nodes (with uncertain inference), we approximately update their probabilities by taking into account the inference for nodes in $\mathcal{Z}$ and applying only forward belief propagation in the R-graph (i.e., only in the direction of its edges). In other words, when a certain inference is made for a node $i \in \mathcal{Z}$, we consider only its effect on the probabilities of the (direct and indirect) children of i, and neglect the effect on the probabilities of its parents. This can be done by removing from the R-graph all the incoming edges to nodes in $\mathcal{Z}$ (*line 13*), and then applying Algorithm 3 (*line 14*), which starts at the roots of the R-graph and through forward belief propagation calculates the probabilities π for all nodes.

Finally, we would like to remark that considering only forward belief propagation is the most reasonable choice in many use cases of our framework; namely, when the detailed preferences of a node i to its parents p_{ij} are not known and their values are arbitrarily set, *e.g.*, to equal values among all parents (see discussion in Section 3.3).

5 PERFORMANCE EVALUATION

In this section, we apply the proposed methods to the Internet AS-graph. Using realistic simulations, we evaluate the capability of our methodology to infer Internet routes, and discuss related insights.

5.1 Setup

We build the AS-level topology using the experimentally collected CAIDA AS-relationship dataset [5]. This contains a list of $\sim 452k$ peering links between $\sim 62k$ ASes, and their relationships $\ell_{ij} \in$

Algorithm 6 Selection of the set of nodes to be measured.

Input: R-graph $\mathcal{G}_R(\mathcal{N}_R, \mathcal{E}_R)$; ingress points $\mathcal{M}$; routing function f; probabilities π; set of measurement-eligible nodes $\mathcal{Y}$, with $\mathcal{Y} \subseteq \mathcal{N}_R$; measurement budget B, with $B < |\mathcal{Y}|$.

1: $X \leftarrow \varnothing$; $P(\varnothing \triangleright m) \leftarrow 1$; $P(i \triangleright m) \leftarrow \pi_i(m)$, $\forall i \in \mathcal{Y}, m \in \mathcal{M}$ /* Initialization */
2: **while** $|X| < B$ **do**   /* The greedy node selection loop */
3:       $i \leftarrow \arg\max_{j \in \mathcal{Y} \setminus X} E_{(P,\pi)}[\mathcal{N}C_R(X \cup \{j\})]$
4:       $X \leftarrow X \cup \{i\}$
5:       $\pi, P \leftarrow$ UPDATEPROBABILITIES (X, i, π, P)
6: **end while**
7: **return** X

8: **function** UPDATEPROBABILITIES(X, i, π, P)
9:       **for** $x \in \mathcal{M}^{|X|}, m \in \mathcal{M}$ **do**
10:          $P(X \cup \{j\} \triangleright x \cup m) \leftarrow P(X \triangleright x) \cdot \pi_i^{(X \triangleright x)}(m)$
11:          $f \leftarrow$ ALGORITHM4$(\mathcal{G}_R, \mathcal{M}, f, \pi^{(X \triangleright x)}, X \cup \{i\} \triangleright x \cup m)$
12:          $\mathcal{Z} \leftarrow \{j \in \mathcal{N}_R : f(j) \neq 0\}$
13:          $\mathcal{G}'_R \leftarrow \mathcal{G}_R(\mathcal{N}_R, \mathcal{E}_R - \{e_{j\ell} \in \mathcal{E}_R : \ell \in \mathcal{Z}\})$
14:          $\pi^{(X \triangleright x, i \triangleright m)} \leftarrow$ ALGORITHM3$(\mathcal{G}'_R, \mathcal{M}, f)$
15:       **end for**
16:       **return** π, P
17: **end function**

$\{p2c, p2p, c2p\}$. We consider a single node per AS, valley-free routing, and set the policies according to Section 2.3 (Eq. (3) and Eq. (4)). In the simulations, we break ties for routes received from neighbors of the same type (*e.g.,* from two customers) arbitrarily. However, in the inference, we assume that we do not know how exactly the nodes break ties (otherwise inference would be trivial). To account for this (assumed) lack of knowledge, we consider in the inference the more generic values $\hat{q}_{ij} = \hat{q}_{ik} \Leftrightarrow \ell_{ij} = \ell_{ik}$, *i.e.,* equal preferences for all neighbors of the same type. This takes into account all possible tie-breaking outcomes, and corresponds to a practical scenario, where we would like to infer catchment with coarse knowledge of the policies (ℓ_{ij}).

At each simulation, we create a new node n_{dst}, add it to the topology, and add $c2p$ links (with n_{dst} the customer) to $|\mathcal{M}|$ randomly selected nodes; these $|\mathcal{M}|$ nodes are assumed to be connected in different ingress points of n_{dst}. We announce a prefix from n_{dst}, and run (simulate) BGP. For each different scenario setup, we conduct 1000 simulation runs.

5.2 Gains from the R-graph-based Inference

A main contribution of the proposed methodology (basic/certain inference, Sections 3.1 and 3.2) is that it achieves to (i) encode all eligible paths in a simple graph, and (ii) exploit the structure of the R-graph to infer routes even for nodes with multiple eligible paths. The simulation results in Fig. 2 quantify these gains.

Figure 2(a) compares the average number of nodes for which our methodology inferred a certain route (white bars), and the average number of nodes with only one eligible path (black bars). For scenarios in which the network has two ingress points (leftmost bars), *our methodology infers the routes of almost an order of magnitude more nodes than a naive inference* (that infers routes only for nodes with a single eligible path). As the number of ingress points increases, the number of eligible paths –and thus the uncertainty– increases as well; however, even for a large number of ingress points (rightmost bars), our methodology infers around two times more nodes than a naive approach.

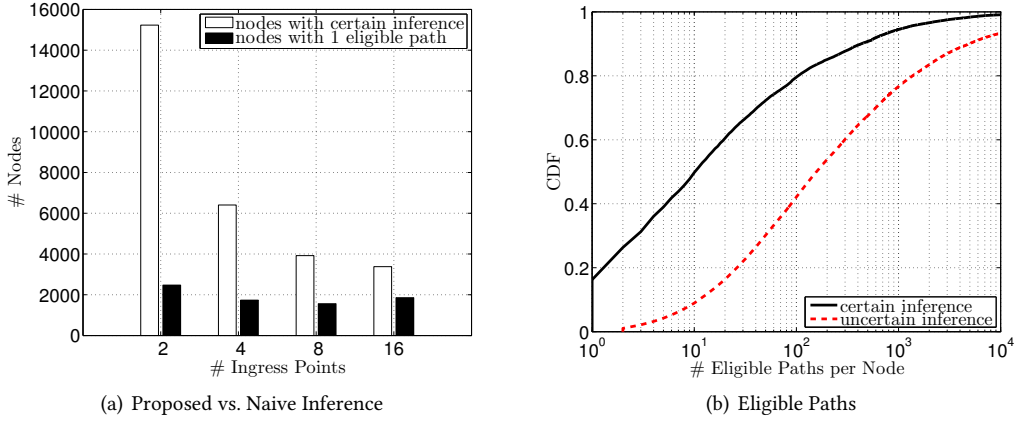(a) Proposed vs. Naive Inference

(b) Eligible Paths

Fig. 2. (a) Number of nodes with certain inference, and number of nodes with one eligible path. (b) Distribution of the number of eligible paths per node, for nodes with certain and uncertain inference; setup with 2 ingress points.

Moreover, Fig. 2(b) shows that 50% (0.5 in y-axis) of the nodes for which an inference can be made (continuous line), have more than 10 eligible paths (x-axis); respectively, in 20% of the inferences (0.8 in y-axis) the nodes have more than 100 eligible paths. This further highlights the gains from exploiting the structure of the R-graph towards making certain inferences.

5.3 R-graph vs. Simulation-based Inference

As discussed earlier, one could use simulation-based approaches to estimate the catchment [17, 34, 40]. Since each simulation run returns a single outcome (which is affected by the randomness in tie-breaking), several runs are needed to calculate estimates of the catchment. On the contrary, our methodology *exactly* calculates the statistics for catchment, in a *lightweight* way (computational complexity is approximately equal to one simulation run). The results in Fig. 3 demonstrate these advantages of our methodology.

In Fig. 3(a), we present results for 5 indicative scenarios (x-axis); in each of them n_{dst} is connected to a randomly selected pair of ingress points, *i.e.,* $\mathcal{M} = \{m1, m2\}$. For each scenario we do the following. (i) We run 1000 simulations, assuming shortest path preference, and for each run we measure the catchment of $m1$; we present the distribution of the results in boxplots (SIMS). (ii) We then apply our methodology to calculate the following quantities:

Lower (LOW) and Upper (UPP) bounds: The certain catchment $|CC(m1)|$ of $m1$, where $CC(m1) = \{i \in \mathcal{N} : f(i) = m1\}$, (calculated by Algorithm 2) is a *lower bound* for the catchment of $m1$, since more nodes (whose inference is not certain) may route to $m1$ as well. Respectively, an *upper bound* for the catchment of $m1$ is given by $|\mathcal{N}| - |CC(m2)|$, since the nodes in $CC(m2)$ cannot route to $m1$. We present the lower/upper bounds with (RG-LOW-SP / RG-UPP-SP) and without (RG-LOW-NO-SP / RG-UPP-NO-SP) shortest path preference.
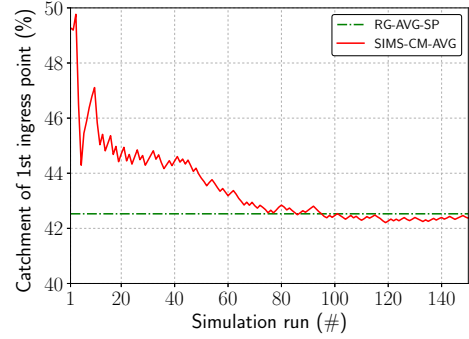
Mean value (AVG): We calculate the mean value of the catchment for $m1$ as $\sum_i \pi_i(m1)$, where the route probabilities π_i are calculated by the probabilistic inference Algorithm 3, with (RG-AVG-SP) and without (RG-AVG-NO-SP) assuming shortest path preference.

Some main observations and insights from the comparison of the simulation results (SIMS) with our predictions are:
(i) The predicted mean values RG-AVG-SP with shortest path preference (*i.e.,* as in the simulation setup) *coincide always* with the average values calculated from the simulation results (SIMS).
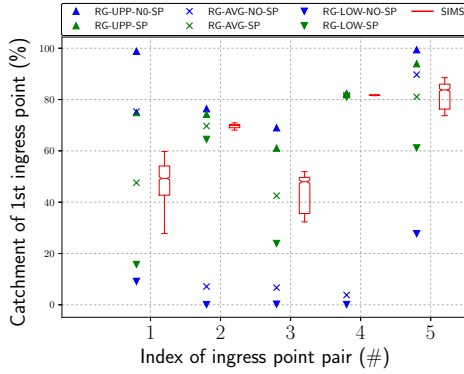
(a) R-graph vs simulations in space   (b) R-graph vs simulations in time

Fig. 3. (a) Simulation results and inferences for the catchment of 1st ingress point ($m1$) for different scenarios. (b) Catchment of 1st ingress point ($m1$) for a scenario, calculated as a cumulative moving average of different simulation runs (`CM-AVG`) and as predicted by our methodology (`RG-AVG-SP`).

Note though that our prediction requires only a single simulation run, whereas simulation-based approaches require several runs to converge to the mean value. We demonstrate this in Fig. 3(b), which shows that the average value of catchment calculated from simulation runs (continuous line), *needs almost 100 runs to converge* to the predicted mean value (dashed line). The presented results are for the 3rd scenario of Fig. 3(a); however, similar patterns were observed across all the pairs we examined.

(ii) As expected, none of the simulation results is outside the bounds `RG-LOW-SP` and `RG-UPP-SP`. When the upper and lower bounds are closer, simulation results are more concentrated around the mean. *The distance between the lower/upper bounds shed light on the effect of the randomness in a simulation.* For example, in the 4th scenario, the bounds coincide, thus showing that the catchment is not affected by the tie-breaking process; or, equivalently, knowing only coarse estimates of the policies is enough for an accurate prediction. On the other hand, in the 1st scenario, the distance between the bounds is larger, which implies that measurements would be needed for an accurate calculation of the catchment.

(iii) The bounds that are calculated without assuming shortest path preference (`NO-SP`) are looser, since they account for a larger set of possible scenarios. The difference between the predictions with (`SP`) and without (`NO-SP`) shortest path preference reveals the effect of the path lengths in a routing configuration. This knowledge can be useful in traffic engineering. For example, in the 2nd scenario, the two *upper* bounds are very close. This means that the maximum catchment of $m1$ is affected mainly by the local preferences and not by the path lengths. Hence, even if we increase the length of the paths to $m2$ through path prepending, this would not increase significantly the catchment of $m1$. On the contrary, the large distance between the two *lower* bounds in the same 2nd scenario, indicates that applying path prepending to announcements through $m1$, can significantly decrease its catchment.

5.4 Completeness of Inference

In Fig. 2(a) we see that, *e.g.,* for $|\mathcal{M}| = 2$ a certain inference is possible for $\sim 15k$ of the total $\sim 62k$ nodes in the graph. Here, we investigate for how many nodes (completeness) our methodology returns a *certain* inference, with or without measurements. *Remark: probabilistic* inference is made for all nodes (see Section 3.3).
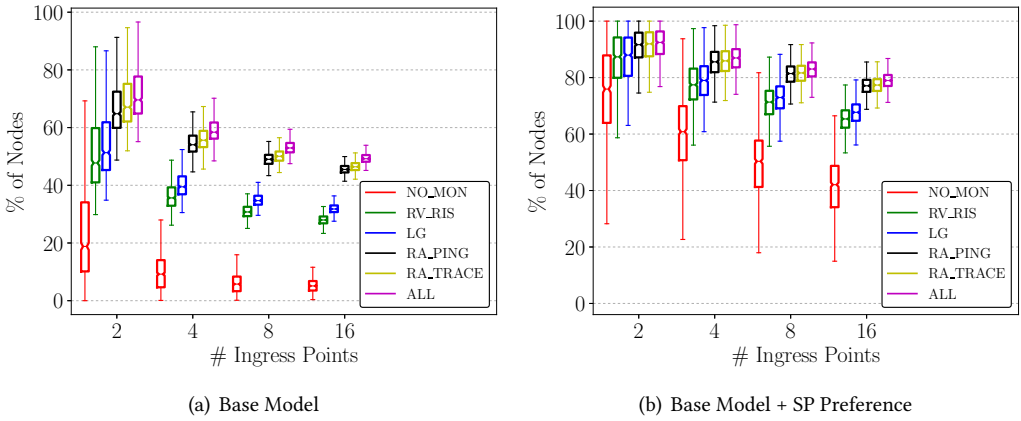
Fig. 4. Distribution (over 1000 simulation runs) of the percentage of nodes with a certain inference (y-axis), in scenarios with different number of ingress points (x-axis). Boxplots correspond to different setups without and with measurements, for scenarios (a) *without* and (b) *with* preference of shorter paths (SP).

To consider realistic scenarios, we simulate measurements from the vantage points of several real Internet measurement platforms:

- **RouteViews** [48] and **RIPE RIS** [43] (RV_RIS) provide BGP RIBs and updates collected from more than 400 ASes worldwide.
- **RIPE Atlas** [42] comprises more than $25k$ probes (in $\sim 3.5k$ ASes), *i.e.,* devices able to run pings (RA_PING) or traceroutes (RA_TRACE) towards certain Internet destinations.
- **Looking Glasses** (LG) are servers that provide the BGP RIBs of the networks (ASes) they are hosted in. We use the Periscope platform [6], to obtain a list of LGs in 883 ASes.

Remark: The BGP data of a network i or traceroutes from i to n_{dst} can provide a route oracle for all the nodes in the best path $bp_{i \rightarrow n_{dst}}$. Pings from n_{dst} to i can provide a route oracle only for i [13].

Figure 4 shows for how many nodes a certain inference is possible in different setups. Some key observations are:

(i) The number of inferences decreases with the number of ingress points. Although this is expected, our methodology quantifies how this behavior is affected by different parameters (number of ingress points, measurement setup, etc.).

(ii) Assuming preference of shorter paths leads to significantly more inferences. For instance, even without measurements (red boxplots - NO_MON) the median percentages increase from $5\% - 19\%$ (Fig. 4(a)) to $42\% - 76\%$ (Fig. 4(b)).

(iii) Public measurement platforms can significantly enhance inference. Their contribution is crucial when many ingress points are in use; *e.g.,* in Fig. 4(a) for $|\mathcal{M}| = 16$ using all platforms increases inference from 5% to 49%, and in Fig. 4(b) from 42% to 79%.

(iv) Interestingly enough, even a lightweight measurement campaign with pings (black boxplots - RA_PING; *e.g.,* as suggested in [13]), can achieve almost the same enhancement with employing all platforms together. However, we simulated pings only to RIPE Atlas probes ($3.5k$ measurements), in contrast to [13] that requires orders of magnitude more measurements; combining our methodology with that technique could potentially lead to even more efficient route inference.
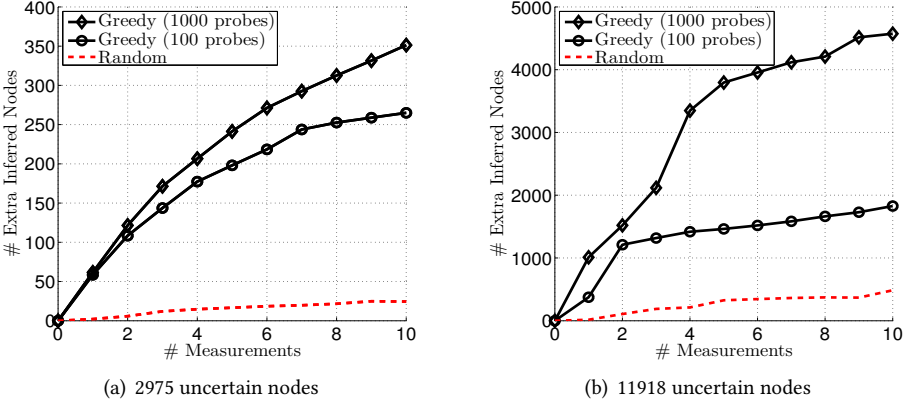
(a) 2975 uncertain nodes   (b) 11918 uncertain nodes

Fig. 5. Number of extra nodes whose routes can be inferred with certainty (y-axis) when a set of measurements $\mathcal{X}$ is provided ($|\mathcal{X}|$ in x-axis), in two scenarios with (a) low and (b) high number of nodes with initially uncertain route.

5.5 Efficient Measurements

Next, we evaluate the ability of Algorithm 6 to select a set of nodes to be measured. We consider two scenarios with $|\mathcal{M}| = 2$, taking into account only the $\sim 20k$ non-stub nodes of the AS-graph, and apply Algorithm 2 to calculate the certain inference. The number of nodes whose routes cannot be inferred with certainty are 2975 ("low") and 11918 ("high") in the scenarios of Fig. 5(a) and Fig. 5(b), respectively. To enhance inference, we conduct measurements to a set of nodes $\mathcal{X}$, and then apply Algorithm 4. In Fig. 5 we present results for the extra number of nodes whose routes are inferred with certainty after the measurements. Sets $\mathcal{X}$ are selected with the greedy Algorithm 6 among 100 and 1000 nodes with RIPE Atlas probes (continuous lines, "Greedy"), or are selected randomly among nodes with RIPE Atlas probes (dashed line, "Random"). The main observation is that selecting the nodes to be measured with the proposed algorithm is significantly more efficient than a random selection. Our algorithm is able to select a good set of nodes, and its efficiency increases when the set of available nodes (with probes) is larger ("Greedy 1000" vs. "Greedy 100"). Comparing the results in Fig. 5(a) and 5(b), reveals that the careful selection of the set $\mathcal{X}$ is more crucial for scenarios with high –initial– uncertainty (Fig. 5(b)); *e.g.,* a single measurement from the node selected with our algorithm can infer with certainty up to 1000 extra routes ("Greedy 1000" in Fig. 5(b)).

5.6 Real-World Evaluation

Besides simulations, here, we provide evaluation results from measurements and experiments in the real Internet.

Measurements for MOAS prefixes. The proposed inference framework can be applied on top of any given topology and routing model (*i.e.,* it takes this information as input). As a result, its accuracy depends on how *complete* and *accurate* the knowledge of (i) the routing policies Q and $\mathcal{H}$ and (ii) the AS-level graph is (perfect knowledge leads to 100% inference accuracy).

We verified this by comparing our inference results against real BGP routing entries collected from more than 200 route collectors of RIPE RIS [43] and RouteViews [48] for around 300 prefixes that are anycasted by more than one AS (*i.e.,* Multi-Origin AS, MOAS) in the Internet [7]. When using the VF model (see Section 2.3) and the available AS-relationships [5], the achieved accuracy

(for networks whose routing entries are available) is 60-70%[7], which complies with the observed accuracy of the VF-model for Internet routing [1]. As a comparison, the accuracy of simulation-based catchment prediction in these scenarios is 10% lower than the accuracy of the certain inference with shortest path preference. Introducing fine-grained refinements in the routing policies for some nodes, increases accuracy; we tested this by considering per-prefix policies, similarly to [1]. Specifically, we re-ran our inference by "correcting" (*i.e.,* replacing, adding, removing) in the R-graph the links close to the anycasters (starting from the first hops), with the actually observed links in the measured paths. For example, if a link was not included in the initial topology dataset (AS-relationships [5]), but was observed in the real measurements, we added it in the R-graph to increase the completeness of the topology; or, if an observed link existed in the topology, but did not appear in the R-graph (*i.e.,* due to an inaccurate routing policy), we similarly added it in the "corrected" R-graph. With a 30% of the links observed by the monitors being corrected, the average accuracy increases to 80%. This observation validates that the inference accuracy depends on the underlying knowledge of the topology and routing policies.

Moreover, the structure of the R-graph provides further insights about what are the important links and policies for a routing configuration, and how missing information (*e.g.,* topology incompleteness) would affect inference. For example, a link that is in the topology dataset but does *not* appear in the R-graph, does not affect the inference, *i.e.,* missing this link (*e.g.,* in an incomplete dataset) would not be important. Similarly, a link that appears in the R-graph but removing it does not affect the *certain* inference of any node, would not be important for the examined routing configuration. This information could be used –similarly to our experiments– to design methods for targeted corrections (*e.g.,* through targeted measurements) of a topology/routing model.

Anycast experiments with the PEERING testbed. We conducted controlled IP anycast experiments in the real Internet using the PEERING testbed [39, 44], which owns ASNs, IP prefixes and has BGP connections with operational networks in several locations around the world. We announce the same prefix from different PEERING locations, *i.e.,* ingress points. Figure 6 shows the fraction of the catchment of each ingress point in four experiment scenarios (SC-0, SC-1, SC-2, and SC-2*), as measured from the RIPE RIS [43] and RouteViews [48] route collectors (black bars), and inferred using our framework (Eq. (5) with $T_i = T, \forall i$) on top of the VF model (white bars).

We consider an initial scenario ("SC-0"), where a network has two ingress points, *AMS* and *UFMG* (for more details about the PEERING locations see [39]). As seen in Fig. 6, the load distribution is highly skewed towards *AMS*; our inference captures this imbalance, with a 10% deviation from measured values. The network would like to evaluate whether it can balance the load by adding more ingress points, before proceeding to an actual deployment (*i.e.,* hypothetical scenario).

One option ("SC-1") is to add an extra ingress point (at the PEERING location *GRNET*). However, our inference (white bars for SC-1) predicts that this would have only a small effect on the load distribution, and thus it would be an inefficient deployment. Our experimental results (black bars for SC-1) verify this behavior, i.e., the added ingress point in SC-1 ("other" bars) attracts a small percentage of traffic. Hence, the network considers a second option ("SC-2") to add two other ingress points (at the PEERING locations *ISI* and *UW*). Our inference predicts that SC-2 would (i) move a significant fraction of load from *AMS* to the added ingress points, and (ii) not affect the load of *UFMG, i.e.,* SC-2 achieves a better load balancing than SC-1. While deviations between inferred and measured catchment exist also here (due to the employed naive VF model), the actual behavior is approximated well by our predictions.

---

[7]An interesting observation is that assuming shortest path preference (Section 3.5) increases the inference completeness from 30% to 65%, without significantly affecting the accuracy; this supports the real-world relevance of this assumption [1, 19, 40].
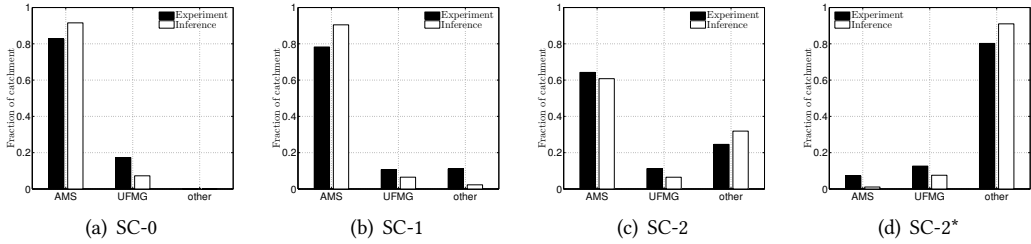
Fig. 6. Experiments in the real Internet with the PEERING testbed: Catchment of the ingress points {*AMS, UFMG, other*} in four deployments/scenarios (SC-0, SC-1, SC-2, SC-2*) as observed from monitors in the Internet (black bars) and predicted using our framework (white bars).

Moreover, we calculated the certain catchment *without* shortest path preference for *AMS* in SC-2, which corresponds to a "lower bound" (cf. Section 5.3) for the *AMS* catchment, i.e., under any path length combination. We found the *AMS* certain catchment to be almost zero (not shown in Fig. 6). This means that the short path lengths towards *AMS* are the main causes for traffic to be routed to this ingress point. Therefore, prepending the announcements from *AMS* (to artificially increase path lengths) could further decrease the attracted load. In fact, since the *AMS* certain catchment is very small, an intensive prepending could even diminish the load in *AMS*. We verified this through experiments ("SC-2*") with the ingress points of SC-2, where we prepended 5 hops (i.e., more than the median AS-path length in the Internet [45]) in the announcements from *AMS*.

6 RELATED WORK

The majority of related literature focuses on methodologies for *measuring* the catchment in existing deployments [2, 9, 13, 27, 32, 49]. A methodology for measuring the routes towards the different ingress points of a destination network, based on past measurements, is proposed in [2]. Similarly, [32] infers AS-level paths $bp_{i \to n_{dst}}$ without measurements from the source network i, but based on BGP tables collected from multiple vantage points, AS-relationship data, and valley-free assumptions, while [27] infers routes by stitching path segments from existing measurements. Latency-based [9] and data-plane [13] measurement methodologies have been recently proposed for mapping anycast catchment. In *Verfploeter* [13], a system in the network of n_{dst} performs exhaustive ping measurements (to all routed IP prefixes) and monitors from which ingress point the reply packets arrive to n_{dst}. In contrast to these works, our methodology can infer catchment also on *hypothetical* deployments (see, *e.g.*, Section 5.6), a task more challenging than the already demanding task of calculating existing catchment [29, 30]. Furthermore, our methodology can complement existing measurement methods, and can be used as a base for devising more light-weight/efficient techniques, *e.g.*, by exploiting the structure and knowledge offered by the R-graph (similarly and by extending the concepts presented in Section 4).

Prior work on Internet route prediction comprises mainly simulation-based approaches [15, 17, 34, 40], which simulate the operation of BGP based on known or estimated routing policies. Our work builds on top of these approaches, and provides more informative results and insights. The works [15, 40] develop models [15] and tools [15, 40] mainly for the intra-domain routing (iBGP) and traffic engineering (TE) of egress traffic, whereas our goal is to predict ingress routes and perform TE with eBGP policies. Nevertheless, these approaches could be combined with ours for a joint intra/inter-domain TE. The importance of the intra-domain structure to the inter-domain

routing is highlighted in [34], whose approach is orthogonal and could be used complementarily to our approach as well, *e.g.,* to provide more fine-grained routing policies Q and $\mathcal{H}$.

Route inference or prediction has been employed in different contexts as well, *e.g.,* for designing targeted active measurements [12], optimal monitor placement [21], or investigation of potential path redundancy in the Internet [23]. Our framework can be used complementary to these works. Finally, probabilistic network programming languages [18], which capture probabilistic network behavior and analyze it through standard probabilistic inference methods, could be combined with our work to design novel efficient inference tools for Internet routing applications.

7 CONCLUSIONS

We proposed and studied a methodology to infer routing behavior in the Internet for existing or hypothetical topological and routing configurations. Our methodology deviates from and enhances existing approaches, by predicting ingress point catchment with certainty or probabilistically, with or without measurements, and under generic routing assumptions. Our methods can be useful for a number of network management application, as well as open new research directions; some indicative examples are:

Applications. (i) *Traffic Engineering:* An operator can efficiently predict and obtain rich information (lower/upper bounds through certain catchment, effect of path lengths, etc.) about the impact of adding/removing ingress points or doing path prepending; due to the large number of possible actions and their combinations, evaluation through experiments or simulation-based approaches would become inefficient. (ii) *Peering strategy:* Establishing a new peering connection with a single network or many networks at an IXP, may significantly change the catchment of ingress points or may have negligible impact (see experiments in Section 5.6). Today, networks have higher flexibility in establishing peerings even with distant networks, *e.g.,* through resellers and remote peering [8, 36]; catchment prediction can enable them to make informed decisions, before proceeding to actual deployments. (iii) *Resilience:* Our framework facilitates to study the resilience of a network against failures of ingress points or peering links. The structure and properties of the R-graph (*e.g.,* centrality) can further reveal the links whose failure would affect the network the most. We believe that a graph-theoretic approach, based on the R-graph, could complement and enhance existing measurement-based approaches, *e.g.,* [16]. (iv) *Network security:* IP anycast is used by DDoS protection organizations to attract and scrub DDoS traffic destined to a victim network [14], or to mitigate hijacking attacks [47]. These organizations can select where to deploy ingress points in order to maximize their catchment (*e.g.,* by mapping potential attackers to "illegitimate" ingress points), and thus best protect their customers.

Future research directions. We identify two future research directions that could be facilitated by our framework. (i) *Internet routing models:* Existing models for Internet topology and routing [15, 17, 34, 40], such as the AS-graph and the VF, are widely used in research and network operations. Despite (or, due to) their generality, they suffer from limited accuracy. However, this accuracy can be increased when modeling the topology and the policies from the perspective of a single network (*e.g.,* per-prefix policies; see [1] and Section 5.6), rather than having a common topology/routing model for all networks. To this end, one could use the R-graph, which encodes the topology and routing policies from the perspective of a single network, n_{dst}. For example, a R-graph can be created on top of a general model, and then refined from real measurement data, *e.g.,* similarly to [34]. While building a *general* (*i.e.,* for all networks) data-driven model, such as [34], may require a very large number of measurements [21] to capture accurately all routing information, when it comes to the perspective of a single network n_{dst}, one can focus her efforts only on the "important" links (see Section 5.6). (ii) *Reinforcement learning for network management:* Optimization problems

that arise in network management processes are frequently combinatorial (see, *e.g.,* Section 4), and thus difficult to solve with analytic methods. Recently, Reinforcement Learning (RL) methods have been proposed for efficient network management and routing operations [51, 52]. In absence of real data, RL agents can be trained on simulated environments. Our framework offers richer information than simulations and requires less computations (*e.g.,* see Section 5.3). Hence, it could significantly reduce the time needed for the training of a RL agent that would involve testing over a large number of different scenarios.

To facilitate further research and reproducibility, we make the code for an implementation of the proposed methods available in [46].

REFERENCES

[1] Ruwaifa Anwar, Haseeb Niaz, David Choffnes, et al. 2015. Investigating interdomain routing policies in the wild. In *Proc. ACM IMC.*

[2] Guillermo Baltra, Robert Beverly, and Geoffrey G Xie. 2014. Ingress point spreading: A new primitive for adaptive active network mapping. In *Proc. PAM.*

[3] Andrew An Bian, Joachim M Buhmann, Andreas Krause, and Sebastian Tschiatschek. 2017. Guarantees for Greedy Maximization of Non-submodular Functions with Applications. In *International Conference on Machine Learning (ICML).*

[4] Matthew Caesar and Jennifer Rexford. 2005. BGP routing policies in ISP networks. *IEEE network* 19, 6 (2005), 5–11.

[5] CAIDA. 2018. AS-Relationships Dataset. http://data.caida.org/datasets/as-relationships/. Dataset collected on 1st July 2018.

[6] CAIDA. 2018. Periscope Looking Glass API. http://www.caida.org/tools/utilities/looking-glass-api/.

[7] CAIDA. 2018. Routeviews Prefix-to-AS mappings (pfx2as) for IPv4 and IPv6. http://data.caida.org/datasets/routing/routeviews-prefix2as/.

[8] Ignacio Castro, Juan Camilo Cardona, Sergey Gorinsky, and Pierre Francois. 2014. Remote peering: More peering without internet flattening. In *Proc. ACM CoNEXT.* 185–198.

[9] Danilo Cicalese, Diana Joumblatt, Dario Rossi, Marc-Olivier Buob, Jordan Augé, and Timur Friedman. 2015. A fistful of pings: Accurate and lightweight anycast enumeration and geolocation. In *Proc. IEEE INFOCOM.*

[10] Cisco. 2019. BGP Best Path Selection Algorithm. https://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html.

[11] Gregory F Cooper. 1990. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial intelligence* 42, 2-3 (1990), 393–405.

[12] Ítalo Cunha, Pietro Marchetta, Matt Calder, et al. 2016. Sibyl: a practical Internet route oracle. In *Proc. USENIX NSDI.*

[13] Wouter B De Vries, Ricardo de O Schmidt, Wes Hardaker, et al. 2017. Broad and load-aware anycast mapping with verfploeter. In *Proc. ACM IMC.*

[14] Wouter B de Vries, Ricardo de O Schmidt, and Aiko Pras. 2016. Anycast and its potential for DDoS mitigation. In *IFIP International Conference on Autonomous Infrastructure, Management and Security.* Springer, 147–151.

[15] Nick Feamster, Jared Winick, and Jennifer Rexford. 2004. A model of BGP routing for network engineering. In *ACM SIGMETRICS PER*, Vol. 32. 331–342.

[16] Romain Fontugne, Anant Shah, and Emile Aben. 2018. The (thin) bridges of as connectivity: Measuring dependency using AS hegemony. In *International Conference on Passive and Active Network Measurement.*

[17] Lixin Gao and Jennifer Rexford. 2001. Stable Internet routing without global coordination. *IEEE/ACM TON* 9, 6 (2001), 681–692.

[18] Timon Gehr, Sasa Misailovic, Petar Tsankov, Laurent Vanbever, Pascal Wiesmann, and Martin Vechev. 2018. Bayonet: probabilistic inference for networks. In *Proc. ACM SIGPLAN Conference on Programming Language Design and Implementation.* 586–602.

[19] Phillipa Gill, Michael Schapira, and Sharon Goldberg. 2011. Let the market drive deployment: A strategy for transitioning to BGP security. In *ACM SIGCOMM CCR*, Vol. 41. 14–25.

[20] Vasileios Giotsas, Matthew Luckie, Bradley Huffaker, et al. 2014. Inferring complex AS relationships. In *Proc. ACM IMC.*

[21] Enrico Gregori, Alessandro Improta, Luciano Lenzini, Lorenzo Rossi, and Luca Sani. 2012. On the incompleteness of the AS-level graph: a novel methodology for BGP route collector placement. In *Proc. ACM IMC.*

[22] Gonca Gürsun and Mark Crovella. 2012. On traffic matrix completion in the internet. In *Proc. ACM IMC.*

[23] Rowan Kloti, Vasileios Kotronis, Bernhard Ager, et al. 2015. Policy-compliant path diversity and bisection bandwidth. In *Proc. IEEE INFOCOM.*

[24] Kevin B. Korb and Ann E. Nicholson. 2010. *Bayesian Artificial Intelligence* (2nd ed.). CRC Press, Inc.
[25] Andreas Krause and Daniel Golovin. 2012. Submodular function maximization. *Tractability: Practical Approaches to Hard Problems* 3, 19 (2012), 8.
[26] P Lapukhov, A Premji, and J Mitchell. 2016. Use of BGP for routing in large-scale data centers. RFC 7938.
[27] DK Lee, Keon Jang, Changhyun Lee, Gianluca Iannaccone, and Sue Moon. 2011. Scalable and systematic Internet-wide path and delay estimation from existing measurements. *Computer Networks* 55, 3 (2011), 838–855.
[28] Zhihao Li, Dave Levin, Neil Spring, and Bobby Bhattacharjee. 2018. Internet Anycast: Performance, Problems, and Potential. In *Proc. ACM SIGCOMM*.
[29] Kurt Lindqvist and Joe Abley. 2006. Operation of Anycast Services. RFC 4786. https://doi.org/10.17487/RFC4786
[30] Aemen Lodhi, Nikolaos Laoutaris, Amogh Dhamdhere, and Constantine Dovrolis. 2015. Complexities in Internet peering: Understanding the "black" in the "black art". In *Proc. IEEE INFOCOM*.
[31] Matthew Luckie, Bradley Huffaker, Amogh Dhamdhere, et al. 2013. AS relationships, customer cones, and validation. In *Proc. ACM IMC*.
[32] Z Morley Mao, Lili Qiu, Jia Wang, and Yin Zhang. 2005. On AS-level path inference. In *ACM SIGMETRICS PER*, Vol. 33. 339–349.
[33] Giovane C.M. Moura, Ricardo de O. Schmidt, John Heidemann, Wouter B. de Vries, Moritz Muller, Lan Wei, and Cristian Hesselman. 2016. Anycast vs. DDoS: Evaluating the November 2015 Root DNS Event. In *Proc. ACM IMC*.
[34] Wolfgang Mühlbauer, Anja Feldmann, Olaf Maennel, Matthew Roughan, and Steve Uhlig. 2006. Building an AS-topology model that captures route diversity. *ACM SIGCOMM CCR* 36, 4 (2006), 195–206.
[35] NANOG mailing list archives. 2018. How to choose a transit provider? http://seclists.org/nanog/2018/Dec/281.
[36] George Nomikos, Vasileios Kotronis, Pavlos Sermpezis, Petros Gigis, Lefteris Manassakis, Christoph Dietzel, Stavros Konstantaras, Xenofontas Dimitropoulos, and Vasileios Giotsas. 2018. O Peer, Where Art Thou?: Uncovering Remote Peering Interconnections at IXPs. In *Proc. ACM IMC*. 265–278.
[37] Chiara Orsini, Alistair King, Danilo Giordano, Vasileios Giotsas, and Alberto Dainotti. 2016. BGPStream: a software framework for live and historical BGP data analysis. In *Proc. ACM IMC*. https://bgpstream.caida.org/.
[38] Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference* (1 ed.). Morgan Kaufmann.
[39] PEERING. 2019. The PEERING testbed. https://peering.usc.edu/.
[40] Bruno Quoitin and Steve Uhlig. 2005. Modeling the routing of an autonomous system with C-BGP. *IEEE network* 19, 6 (2005), 12–19. http://c-bgp.sourceforge.net//index.php.
[41] Yakov Rekhter, Tony Li, and Susan Hares. 2006. A Border Gateway Protocol 4 (BGP-4). RFC 4271.
[42] RIPE NCC. 2018. RIPE Atlas. https://atlas.ripe.net/.
[43] RIPE NCC. 2018. Routing Information Service (RIS). https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris.
[44] Brandon Schlinker, Kyriakos Zarifis, Italo Cunha, Nick Feamster, and Ethan Katz-Bassett. 2014. PEERING: An AS for Us. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks (HotNets)*.
[45] Pavlos Sermpezis and Xenofontas Dimitropoulos. 2017. Can SDN accelerate BGP convergence? A performance analysis of inter-domain routing centralization. In *2017 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, 1–9.
[46] Pavlos Sermpezis and Vasileios Kotronis. 2019. GitHub repository with the Catchment Inference in Internet Routing code and algorithm implementation. https://github.com/FORTH-ICS-INSPIRE/anycast_catchment_prediction.
[47] Pavlos Sermpezis, Vasileios Kotronis, Petros Gigis, Xenofontas Dimitropoulos, Danilo Cicalese, Alistair King, and Alberto Dainotti. 2018. ARTEMIS: Neutralizing BGP hijacking within a minute. *IEEE/ACM Transactions on Networking (TON)* 26, 6 (2018), 2471–2486.
[48] University of Oregon. 2018. Route Views Project. www.routeviews.org.
[49] Verizon. 2017. Seeing the World with RIPE Atlas. https://labs.ripe.net/Members/verizon_digital/seeing-the-world-with-ripe-atlas.
[50] Lan Wei and John Heidemann. 2018. Does anycast hang up on you? *IEEE Transactions on Network and Service Management* (2018).
[51] Haipeng Yao, Tianle Mai, Xiaobin Xu, Peiying Zhang, Maozhen Li, and Yunjie Liu. 2018. NetworkAI: An intelligent network architecture for self-learning control strategies in software defined networks. *IEEE Internet of Things Journal* 5, 6 (2018), 4319–4327.
[52] Changhe Yu, Julong Lan, Zehua Guo, and Yuxiang Hu. 2018. DROM: Optimizing the Routing in Software-Defined Networks With Deep Reinforcement Learning. *IEEE Access* 6 (2018), 64533–64539.

RIGHTSLINK()

APPENDIX

A PROOF OF THEOREM 1

We first define as R-path from i to n_{dst}, a path created by starting at n_{dst} and following directed edges until reaching i.

We prove the Theorem, by proving the following two items: (i) any path in the R-graph (*i.e.,* R-path), is an eligible path; (ii) any eligible path is encoded in the R-graph as a R-path.

Any path in the R-graph (i.e., R-path), is an eligible path. Let a R-path $rp = [n_1, n_2, ..., n_K, n_{dst}]$. The rp is constructed by following edges in the R-graph, which means that $e_{n_{k+1}n_k} \in \mathcal{E}_\mathcal{R}$, where $k = 1, ..., K - 1$. The existence of the edge denotes that (see Algorithm 1): (a) n_{k+1} is in the set *best_neighbors* of n_k, or equivalently $q_{n_k n_{k+1}} \geq q_{n_k j}, \forall j \in \{i \in \mathcal{N}_\mathcal{R} : e_{in_k} \in \mathcal{E}_\mathcal{R}\}$, and thus can be selected by n_k. (b) n_{k+1} exports its best path $bp_{n_{k+1} \to n_{dst}}$ to n_k, and routes all paths of equal local preference similarly (see Section 2.1); thus any path $[n_{k+1}, x, ..., n_K, n_{dst}]$ can be a path that reaches the RIB of n_k, for any x that $q_{n_{k+1}x} = q_{n_{k+1}n_{k+2}}$. These two conditions satisfy the definition of eligible paths (Def. 1, Section 2).

Any eligible path is encoded in the R-graph as a R-path. Let an eligible path $ep = [n_1, n_2, ... n_K, n_{dst}]$ that is not a R-path, *i.e.,* at least one edge in ep does not exist in the R-graph; let this edge be between n_{k+1} and n_k. Let also a node x that is a parent of n_k in the R-graph (*i.e.,* $e_{xn_k} \in \mathcal{E}_\mathcal{R}$). Then, it must hold that $q_{n_k x} > q_{n_k n_{k+1}}$ or $h_{n_{k+1}n_{k+2}n_k} = 0$. In the former case, the path $[n_k, n_{k+1}, ..., n_{dst}]$ cannot be the best path of n_k and thus n_1 will never have in its RIB the path ep (contradiction). In the latter case, the path $[n_{k+1}, n_{k+2}, ..., n_{dst}]$ is never exported to n_k, which means that ep does not conform to routing policies (contradiction).

B PROOF OF LEMMA 1

In general, a node i in the R-graph has more than one paths to n_{dst}, which means that the R-graph is a multiply-connected BN (and not a *polytree*). The problem of updating the probabilities (or, "belief updating") in non-polytree BNs is known to be NP-hard (by reduction to a SAT problem) [11].

C PROOF OF THEOREM 2

Correctness: A node i has a certain route only if (a) all its parents P_i have a certain route, or (b) (at least) one of its children $j \in C_i$ has a certain route and j routes through i. The former case is captured by the condition in *line 23* (for node j and its parents), and the latter in *line 13* where the condition requires that i routes traffic through the node $j \equiv CP_i$.

Completeness: The updating process of Algorithm 4 is based on the fact that a BN node is conditionally independent of all other nodes given its Markov blanket, *i.e.,* its parents, children, and "spouses" (parents of common children) [38]. Hence, for each oracle, let for a node i, Algorithm 4 visits *all nodes that are dependent* on i, *i.e.,* its parents (*line 12*), children (*line 16*), and parents of children (*line 18*). Any other node is not dependent, unless a change in the value/route of a visited node takes place (in that case Algorithm 4 calls again SETROUTE for this node; in *line 14* or *24*).

Complexity: The function SETROUTE is called only for nodes with an oracle (*line 3*), or only for nodes without a certain route (see in *line 13* condition $f(CP_i = 0)$, and in *line 24*, node $j \in C_i$ satisfies the condition $f(j) = 0$ from *line 16*). As soon as a node sees a certain route, it is not considered for further inference. Let $i, j \in \mathcal{X}$ and k a neighbor of both i and j; if SETROUTE is called for k when i is visited, then it will not be re-called for k when j is visited. Hence, SETROUTE is called at most $|\mathcal{N}|$ times. *Remark:* This does not mean that the recursion depth of *lines 14* and *24* is at most one, but that *the sum of recursive calls of SETROUTE is bounded by* $|\mathcal{N}|$.
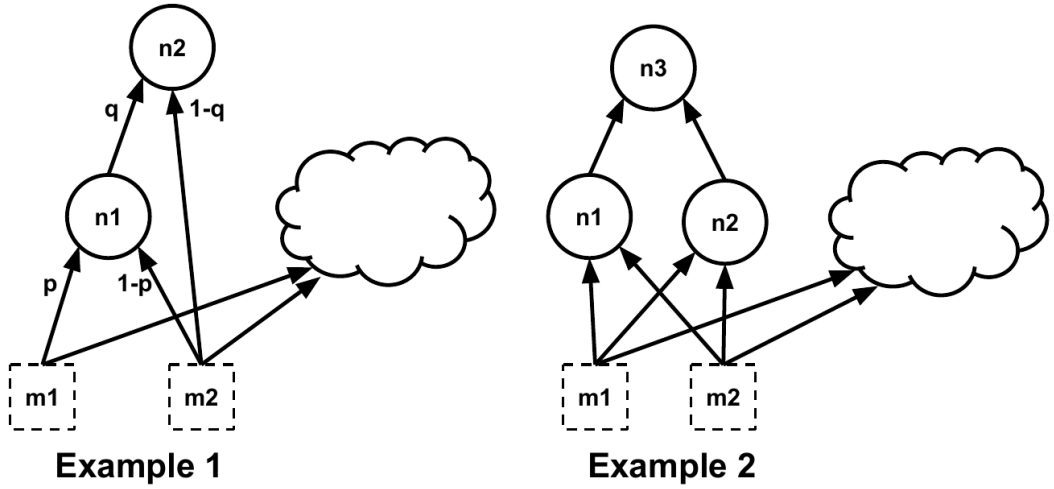
Fig. 7. Examples of a R-graph. The ingress points $m1$ and $m2$ are denoted with boxes, the nodes of interest are denoted with circles, and the clouds correspond to a part of the R-graph that has no (outgoing) edges towards the nodes of interest and whose structure is not of interest. The weight next to each edge denotes the probability for a node to select the route from this incoming edge.

D PROOF OF THEOREM 3

We provide a sketch of the proof. By design, Algorithm 5 only removes edges from the R-graph. A node i has a certain route only if all its parents P_i have the same certain route as well; removing a parent changes neither the route of the other parents, nor the route of i. On the other hand, let all parents of i, except for one parent $j \in P_i$, have the same route m; then i does not have a certain route. If the edge e_{ji} is removed, the remaining parents of i will have the same route m, and thus a new route inference for node i can be safely made.

E PROOF OF LEMMA 2

The first item follows straightforwardly from the definition of the function $|\mathcal{NC}_R(X)|$ (the size of a set is non-negative), and the fact that a measurement is only an observation that cannot change the (certainly inferred) route of a node and thus decrease the number of nodes which already have certain routes. In particular, a certain route for node i is independent of the route probabilities of other nodes without a certain inference (otherwise the route of i would not have been inferred with certainty). Hence, (a) in case the measured node already has a certain route, a (valid) measurement cannot change this route, and (b) in case the measured node does not have a certain route, then it does not affect the route of i; in either case the route of i is not affected.

We prove the second and third items through two counter-examples depicted in Fig. 7.

We remind that a set function f is *submodular* when $\forall A \subseteq B$ and $\forall \epsilon \notin A$ it holds that

$$f(A \cup \{\epsilon\}) - f(A) \ge f(B \cup \{\epsilon\}) - f(B) \tag{10}$$

and is *supermodular* when $\forall A \subseteq B$ and $\forall \epsilon \notin A$ it holds that

$$f(A \cup \{\epsilon\}) - f(A) \le f(B \cup \{\epsilon\}) - f(B) \tag{11}$$

In other words, the marginal gain in a submodular function by adding an element ϵ to a set S diminishes with the size of the set S.

Example 1. Consider the first example in Fig. 7, let A be a set of nodes in the "cloud" of Fig. 7, and let

$$A \cap \{n1, n2\} = \varnothing$$
$$B \equiv A \cup \{n2\}$$
$$\epsilon \equiv \{n1\}$$

In this case, the objective function of Eq. (8) takes the value

$$E_P \left[\mathcal{NC}_R(A \cup \{\epsilon\}) \right] = E_P \left[\mathcal{NC}_R(A) \right] + 1 + (1 - p) \tag{12}$$

because we will have an oracle for $n1$ (i.e., we increment by 1), and if this oracle is $n1 \rhd m2$ (which happens with probability $1 - p$) then we can certainly infer that $n2$ routes to $m2$ as well (i.e., we increment one more by 1, but now with probability $1 - p$); otherwise we cannot infer the route of $n2$ with certainty (and we do not increment).

Also for B we get for the objective function

$$E_P \left[\mathcal{NC}_R(B) \right] = E_P \left[\mathcal{NC}_R(A) \right] + 1 + p \cdot q \tag{13}$$

because we will have an oracle for $n2$ (i.e., we increment by 1), and if this oracle is $n2 \rhd m1$, this means that we can also infer that $n1 \rhd m1$, because this is the only way that $n2$ can route to $m1$, and the respective probability is $p \cdot q$ (i.e., w.p. p the node $n1$ routes to $m1$ and $n2$ selects the route from $n1$ w.p. q); otherwise (i.e., $n1 \rhd m2$) we cannot infer the route of $n1$ with certainty.

Finally, trivially, we get

$$E_P \left[\mathcal{NC}_R(B \cup \{\epsilon\}) \right] = E_P \left[\mathcal{NC}_R(A) \right] + 2 \tag{14}$$

because we have oracles for both nodes $n1$ and $n2$.

The above equations give

$$\Delta_A = E_P \left[\mathcal{NC}_R(A \cup \{\epsilon\}) \right] - E_P \left[\mathcal{NC}_R(A) \right] = 1 + (1 - p) \tag{15}$$
$$\Delta_B = E_P \left[\mathcal{NC}_R(B \cup \{\epsilon\}) \right] - E_P \left[\mathcal{NC}_R(B) \right] = 1 - p \cdot q \tag{16}$$

It is easy to see that $\Delta_A \geq 1 \geq \Delta_B$, which means that *the objective function cannot be supermodular*, because there exists an ϵ for which the inequality Eq. (11) does not hold.

Example 2. Consider the second example in Fig. 7, let A be a set of nodes in the "cloud" of Fig. 7, and let

$$A \cap \{n1, n2, n3\} = \varnothing$$
$$B \equiv A \cup \{n2\}$$
$$\epsilon \equiv \{n1\}$$

The objective function of Eq. (8) takes the value

$$E_P \left[\mathcal{NC}_R(A \cup \{\epsilon\}) \right] = E_P \left[\mathcal{NC}_R(A) \right] + 1 \tag{17}$$

because we will have an oracle for $n1$ (i.e., we increment by 1), and no matter what this oracle is, the route probabilities π for $n2$ and $n3$ will be always non-zero for both $m1$ and $m2$ (i.e., we cannot make any other certain inference).

Also for B we get for the objective function

$$E_P \left[\mathcal{NC}_R(B) \right] = E_P \left[\mathcal{NC}_R(A) \right] + 1 \tag{18}$$

because we will have an oracle for $n2$ (i.e., we increment by 1), and no matter what this oracle is, the route probabilities π for $n1$ and $n3$ will be always non-zero for both $m1$ and $m2$ (*i.e.,* we cannot make any other certain inference).

Finally, let w denote the probability that $n1$ and $n2$ route to the same ingress point. Then, we get

$$E_P\left[\mathcal{NC}_R(B \cup \{\epsilon\})\right] = E_P\left[\mathcal{NC}_R(A)\right] + 2 + w \tag{19}$$

because we will have an oracle for nodes $n1$ and $n2$ (i.e., we increment by 2), and if both oracles for $n1$ and $n2$ are for the same ingress point (which happens w.p. w), we can make one more certain inference for $n3$; otherwise we cannot infer the route of $n3$ with certainty.

The above equations give

$$\Delta_A = E_P\left[\mathcal{NC}_R(A \cup \{\epsilon\})\right] - E_P\left[\mathcal{NC}_R(A)\right] = 1 \tag{20}$$

$$\Delta_B = E_P\left[\mathcal{NC}_R(B \cup \{\epsilon\})\right] - E_P\left[\mathcal{NC}_R(B)\right] = 1 + w \tag{21}$$

It is easy to see that $\Delta_A \leq \Delta_B$, which means that *the objective function cannot be submodular*, because there exists an ϵ for which the inequality Eq. (10) does not hold.