Inferring Visibility: Who's (Not) Talking to Whom?

Gonca Gürsun, Natali Ruchansky, Evimaria Terzi, and Mark Crovella
Department of Computer Science
Boston University

ABSTRACT

Consider this simple question: how can a network operator identify the set of routes that pass through its network? Answering this question is surprisingly hard: BGP only informs an operator about a limited set of routes. By observing traffic, an operator can only conclude that a particular route passes through its network – but not that a route does *not* pass through its network. We approach this problem as one of statistical inference, bringing varying levels of additional information to bear: (1) the existence of traffic, and (2) the limited set of publicly available routing tables. We show that the difficulty depends critically on the position of the network in the overall Internet topology, and that the operators with the greatest incentive to solve this problem are those for which the problem is hardest. Nonetheless, we show that suitable application of nonparametric inference techniques can solve this problem quite accurately. For certain networks, traffic existence information yields good accuracy, while for other networks an accurate approach uses the 'distance' between prefixes, according to a new network distance metric that we define. We then show how solving this problem leads to improved solutions for a particular application: traffic matrix completion.

Categories and Subject Descriptors: C.2.3 [Computer-Communication Networks]: Network Operations.

Keywords: BGP, matrix completion.

1. INTRODUCTION

The global state of the Internet's interdomain routing system at any instant is an almost unknowable mystery. Yet, absent this knowledge, a network operator cannot answer this very simple question: 'Which routes pass through my network?'

Answering this simple question can be useful in many ways. For example, it can inform traffic engineering and capacity planning; when traffic loads change, operators would like to know whether the cause is a change in demand, or a routing change. From a security standpoint, knowing whether a route passes through one's network can be used to identify packets with spoofed source addresses.

Knowing what routes pass through one's network can also provide business intelligence, by shedding light on the routing behavior of customers and competitors. As a concrete example, consider a network A with competitor B. Suppose B has customer C and A would like to make a bid for C's business. The price offered should depend on the proportion of C's traffic that is already passing through A. A simple way to estimate this is to see what fraction of C's routes are passing through A.

From a scientific standpoint, this question has basic appeal. It is equivalent to asking: 'If I capture a traffic matrix from a particular network, and see a zero value in the (i, j) position, what does that mean?'

In this paper we develop methods to answer this visibility question, taking the standpoint of a single network operator. To discover whether the route from i to j passes through the operator's network (i.e., is *visible* in the network), the operator has available two kinds of potentially useful information: observed traffic, and known BGP routes. While BGP provides knowledge of only a limited set of routes to each destination, observed traffic may potentially provide information about any source-destination pair. Hence our methods concentrate on the knowledge obtained from observed traffic. (More discussion of this issue appears in Section 8.)

Unfortunately, traffic only provides *positive* information about visibility: if traffic is observed flowing from i to j then (i, j) is known to be visible; but if no traffic is observed, then it is not possible to conclude anything about the visibility status of (i, j); i.e., traffic does not provide *negative* information about visibility. Thus the key question comes down to asking: if no traffic is observed from i to j, is it because the path from i to j does not pass through the observer, or because i is simply not sending traffic to j? Like Sherlock Holmes, we are interested in 'the dog that didn't bark.'

Our approach starts from simple intuition about how routes spread through the Internet; based on this understanding we develop a family of nonparametric classifiers. We show that different kinds of networks (essentially, networks in different tiers) require different instantiations of this classifier. We then show how to configure these classifiers for different kinds of networks, and demonstrate that we can answer our original question with a surprisingly high level of accuracy. We conclude with an example showing the utility of our classifier when applied to the traffic matrix completion problem.

SIGCOMM'12, August 13–17, 2012, Helsinki, Finland.
Copyright 2012 ACM 978-1-4503-1419-0/12/08 ...$15.00.

2. OVERVIEW

Our central notion is *visibility:* we say that a source-destination pair (i, j) is visible to AS x if, were i to send traffic to j, the traffic would pass through x. We refer to the AS whose owner wishes to infer visibility as the *observer* AS.[1] We use the term *path* to refer to the sequence of ASes taken along the route from a source (traffic generator) to a destination (traffic recipient). We naturally group addresses into prefixes as used in BGP, and associate prefixes with the ASes that appear at the end of their BGP routes. Hence, in what follows 'source' will mean 'source AS' or 'source prefix' (and likewise for 'destination.')

For any given observer AS, we define the corresponding *visibility matrix* T, which has sources on the rows and destinations on the columns, such that $T(i, j) = 1$ iff the source-destination pair (i, j) is visible to the observer, and zero otherwise. In our work, we assume that T is constant during the period of study; in practice we work with snapshots of T.

Of course, the observer does not actually have access to T. Instead the observer can measure traffic flowing through the network. We assume that traffic is measured over some fixed duration, and we organize measured traffic volumes into a matrix V, with the same rows and columns as T. For example, the value of $V(i, j)$ may be the number of bytes sent from i to j during the measurement interval. We refer to V as the *traffic matrix* or *volume matrix.* The (positive) visibility information contained in V defines the *observed* visibility matrix M, with $M(i, j) = 1$ iff $V(i, j) > 0$, and zero otherwise. Thus, M encodes the incomplete visibility information that is learned from observing traffic.

Hence, M is an approximation to T, with the property that if $M(i, j) = 1$, then $T(i, j) = 1$. However, if $M(i, j) = 0$, then $T(i, j)$ may be 0 or 1. Therefore, there are two types of zeros in the observed visibility matrix M: *true zeros* and *false zeros.* A true zero occurs when (i, j) is truly not visible; a false zero occurs when (i, j) is actually visible, but not communicating. In notation: $M(i, j)$ is a *true zero* if $M(i, j) = 0$ and $T(i, j) = 0$; it is a *false zero* if $M(i, j) = 0$ and $T(i, j) = 1$.

2.1 The VISIBILITY-INFERENCE problem

Our goal is to design a mechanism that outputs a matrix $\widehat{T}$ that is a better approximation of T than M is. We call $\widehat{T}$ the *predicted visibility matrix*, and we call the problem of inferring $\widehat{T}$ the VISIBILITY-INFERENCE problem. This is a classification problem that focuses on the zero-valued elements of M, which we denote by Z, (i.e., $Z = \{(i, j) \mid M(i, j) = 0\}$). For every $(i, j) \in Z$, the classifier decides whether (i, j) corresponds to a true zero or a false zero and sets $\widehat{T}(i, j) = 0$ or 1 respectively. In other words, our goal is to set the values of $\widehat{T}(Z)$ (i.e., the subset of the elements of $\widehat{T}$ that correspond to the indices in Z) in such a way that they agree with the corresponding values of $T(Z)$.

To fix terminology, we consider a classification of $(i, j) \in Z$ that outputs $\widehat{T}(i, j) = 0$ to be a *positive* classification – the element is predicted to be, in fact, a true zero. Correspondingly, $\widehat{T}(i, j) = 1$ is a *negative* classification – the element is predicted to be a false zero.

The performance of a classifier can then be expressed in terms of *True Positive Rate* (TPR) and the *False Positive Rate* (FPR) of the predicted matrix $\widehat{T}$. TPR is the classifier's accuracy on true zeros:

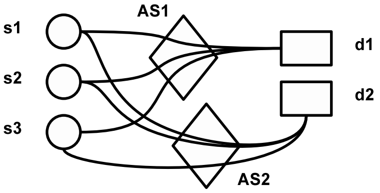


Figure 1: Groups of similar paths

$$\text{TPR} = \frac{\text{number of correctly classified true zeros}}{\text{total number of true zeros}}$$

while 1−FPR is the classifier's accuracy on false zeros:

$$\text{FPR} = \frac{\text{number of incorrectly classified false zeros}}{\text{total number of false zeros}}$$

Thus we can formally describe the VISIBILITY-INFERENCE problem as the task of forming a predicted matrix $\widehat{T}$ such that TPR is as close to 1 as possible and FPR is as close to 0 as possible.

The classifiers we develop will allow tradeoffs between TPR and FPR. To examine this tradeoff and thus assess the overall performance of the classifier, we use ROC (Receiver Operating Characteristic) curves [7]. The ROC curve of $\widehat{T}$ plots TPR versus FPR for different settings of the classifier. The ideal classifier operates at the upper-left point $(0, 1)$. In order to describe the performance of the classifier over its entire range of settings with a single number, we use the *area under the ROC curve of* $\widehat{T}$ (AUC) which takes values between 0 and 1. An ideal classifier (one which can operate at the $(0, 1)$ point) has an AUC of 1, and in general, the closer to 1, the better the prediction of $\widehat{T}$.

2.2 Our approach

Having defined our problem, we now discuss the basic intuition behind our approach. We start from the observation that at any given point in the Internet, there are groups of (i, j) pairs that are routed similarly. This general idea is shown in Figure 1. For example, in some region of the Internet (say, a set of nearby ASes AS_1, AS_2, ...) one may often find a group of paths that all pass through AS_1, and yet none passes through AS_2. Another group may all pass through AS_2 but none through AS_1. At the highest level, our strategy is to identify path groups, and if traffic is observed on some paths in the group, infer that all members of the group are likely to be visible.

More concretely, for any given source-destination pair (i, j), we look for other pairs that we believe to have a similar visibility status as (i, j). We then use any knowledge we have about the other pairs to form a visibility estimate for (i, j). Essentially, this is nearest-neighbor classification (with incomplete information). The principal challenge then becomes to properly identify which source-destination pairs are 'neighbors' to (i, j).

On a high level, our classifier takes a specific form: given some $(i, j) \in Z$,

1. Select an (i, j)-*descriptive submatrix* from M. This

[1] We work at the granularity of ASes, rather than ISPs; some implications of this are discussed in Section 8.

submatrix, denoted by $M(S_i, D_j)$, is defined by the set of sources S_i and destinations D_j.

2. Compute the value of some *structural property* of $M(S_i, D_j)$. We call this the *descriptive value* π_{ij}.

3. If the descriptive value is above a *threshold*, predict $\widehat{T}(i, j) = 1$; otherwise, predict $\widehat{T}(i, j) = 0$.

This approach performs classification of each $(i, j) \in Z$ independently. Although more complicated mechanisms could classify pairs jointly, our experimental evaluation indicates that the independent approach yields very accurate predictions.

The goal of the first step is to find a submatrix $M(S_i, D_j)$ whose elements have a similar visibility status with respect to the observer. In practice, we simplify this; we only require that sources in S_i are similar in some way, and that destinations in D_j are similar in some way, with respect to the observer. Of course, different kinds of similarity lead to different submatrices. We use two types of similarity, based on different information. The first is based on the *visibility information* – as encoded in the observed visibility matrix. The second is based on *proximity information* – as encoded in publicly available BGP tables. These two strategies are described in Sections 4 and 5, respectively.

The second step evaluates different *structural properties* of $M(S_i, D_j)$. These properties are discussed in detail in Sections 4 and 5; here we just note that these are simple properties like the number of nonzero elements in $M(S_i, D_j)$, the size of $M(S_i, D_j)$ or its density.

The third step classifies (i, j) based on whether its descriptive value π_{ij} is above a threshold. By varying this threshold, one can trade off TPR and FPR.

In practice, it is necessary to choose a threshold value to configure the classifier. If the classifier is robust, there will be a reasonably wide range of threshold values that yield good results, so its accuracy will not be highly sensitive to threshold settings. We show that this is the case for the classifiers we develop in Sections 4 and 5. In Section 6, we show how to actually choose a threshold value based on observable data, and we show that such automatically-chosen threshold results in a highly-accurate classifier.

3. DATASETS

To evaluate our methods, we need a large survey of ground-truth visibility matrices T. To obtain these matrices, we use a collection of BGP tables (collected on midnight UTC on December 6th, 2011) obtained from the Routeviews [14] and RIPE [12] repositories.

The full dataset is collected from 359 unique monitors; it consists of over 48 million AS paths, and contains 454,804 unique destination prefixes. (Note that not all BGP tables show paths to all prefixes.) Because these paths are the *active* paths at the time of collection, each path represents the sequence of ASes that traffic flows through when going from the particular monitor to the path's destination prefix.

Visibility Matrices. Using this data we construct visibility matrices for every AS appearing in the dataset – 23,511 ASes. In order for visibility matrices to be comparable across ASes, each matrix must be indexed by the same set of rows and columns. From the full dataset, we select a subset of monitors and a subset of prefixes such that new dataset contains the AS path from *every* monitor to *every* prefix. This results in 38 monitor ASes and 135,369 prefixes. For each of the 23,511 ASes, we construct a 0-1 visibility matrix of size 38 × 135,369. Thus, each visibility matrix has about 5.2 million entries; each matrix entry (i, j) records the visibility status of the pair (AS i, prefix j).

In a given observer AS's visibility matrix T, an entry (i, j) is 1 if the path from AS i to prefix j contains the observer AS, and 0 otherwise. Since we have all (5.2 million) active paths from every monitor AS to every prefix, we know with certainty the 0-1 value of every entry in T for every observer AS. Of course, these visibility matrices are only a portion of the complete visibility matrix for each observer, but each provides millions of ground-truth values for validating our methods.

For each ground-truth matrix T, we also need an observed visibility matrix M, which differs from T by having some 1s turned into (false) zeros. The number of false zeros in M is affected by the duration over which traffic is observed. Longer traffic observation periods result in smaller numbers of false zeros, as additional source-destination pairs generate observable traffic. Hence we study a range of false zeros, expressed as a fraction of visible elements (1s) in T: from 10% false zeros (corresponding to a long measurement period), to 95% false zeros (corresponding to very short measurement period). We generate false zeros by randomly flipping 1s in T to zeros; the result becomes the observed visibility matrix M. We call the fraction of 1s flipped in M the *flipping percentage.* When the flipping percentage is small (10%) classifiers have more information to work with; when the flipping percentage rises to 95% classifiers have very little information to work with and the classification problem is quite challenging.

Our bit-flipping strategy models the case where the traffic of each source-destination pair is independent of the others. While this reflects the basic unpredictability of traffic patterns, it is possible that correlations in traffic patterns could affect the accuracy of our classifier. For that reason, we also use another strategy: *destination-based* flipping. The goal is to model the situation where a particular destination prefix receives no traffic from any source – such as when there are no hosts provisioned with addresses from the prefix. In destination-based flipping, *all* of the 1s in an *entire* column are flipped to zero.

Observer AS Types. An important fact is that the arrangement of 1s in the visibility matrix of an AS shows distinct patterns that depend on the AS's topological location in the AS graph. For instance, in the visibility matrix of an AS that sits in the core of the graph, the 1-valued entries are scattered relatively uniformly; in contrast, for an AS that sits at the edge of the graph, the 1-valued entries are clustered in a small set of rows and columns. This is a natural consequence of the routing structure of the Internet.

Hence, as we will show in Sections 4 and 5, the topological location of an AS affects the relative performance of different classifiers. To distinguish ASes in different locations, we use two metrics: *node degree* and *k-shell number.* We compute these metrics using our BGP dataset; although these metrics are sensitive to missing edges in the BGP graph, we only use them for ranking ASes, not for quantitative comparisons.

Degree is the number of observed neighbors in the BGP graph derived from our set of paths. K-shell number measures the centrality of a node in a graph [1]. It is computed using the *k-core decomposition*, which separates the nodes of

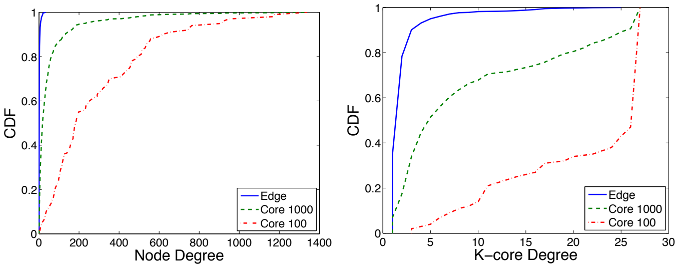


Figure 2: Properties of AS Sets Studied.

a graph into nested sets called 'shells.' As described in [5], this is a parameter-free way of characterizing the AS graph, and it corresponds to a natural notion of centrality.

Using these metrics, we define three sets of ASes to represent different topological positions. First, the `Core-100` set consists of the 100 ASes with highest k-shell number. Second, the `Core-1000` set consists of the 1000 ASes with the highest k-shell number (and so contains the `Core-100` set). Finally, the `Edge` set consists of 1000 ASes that have low degree and k-shell number. This set contains mainly stub ASes and ASes that are topologically close to stubs. This set is representative of 'typical' ASes in that almost 95% of the ASes in our dataset are stubs. The `Edge` set was constructed by randomly sampling among all the ASes not contained in the `Core-1000` set. Figure 2 shows the distribution of degree and k-shell numbers for these three sets. The Figure shows that the `Core-1000` set is intermediate in both respects between the `Edge` and `Core-100` sets, while the other two sets reflect extremes: boundary and center of the AS graph.

Traffic Matrix. Some of the experiments we report use knowledge of source-destination traffic volumes, i.e., the traffic matrix V. Traffic matrices are generally hard to obtain at the fine grain we work with in this paper (AS-prefix traffic volumes). However we were able to obtain **`netflow`** data from a Tier-1 provider (a member of the `Core-100` set) suitable for our purpose, and collected in the same timeframe as the BGP data. This consists of traffic volumes measured in bytes, over a duration of one day. We organized the flow data according to source AS and destination prefix, using the same row and column indices as our visibility matrices, into a traffic matrix V.

4. THE VISIBILITY-BASED METHOD

We start our exploration of problem solutions by examining the most straightforward approach to inferring visibility of unknown entries, namely, making use of the observed visibility matrix. We term this the *visibility-based* method. We first describe the visibility-based method by showing how it instantiates the general strategy described in Section 2.2; then, we examine the method's accuracy and applicability.

4.1 Method description

As described in Section 2.2, our general strategy seeks to find collections of source-destination pairs whose visibility status is likely to be similar to the target pair (i, j). In this section we use the positive information in M directly, and ask "How many sources and destinations have (positive) visibility that is similar to the target (i, j)?" To ask this question, we instantiate the generic method presented in Section 2.2 as follows:

Submatrix selection: Given $(i, j) \in Z$, the visibility-based method selects the (i, j)-descriptive submatrix as:

$$\begin{aligned} S_i &= \{i\} \cup \{i' \mid M(i', j) = 1\} \quad \text{and} \\ D_j &= \{j\} \cup \{j' \mid M(i, j') = 1\}. \end{aligned}$$

That is, the set S_i consists of i as well as all the sources that have have been observed to send traffic to destination j. Similarly, the set D_j contains j as well as all the destinations that have been observed to receive traffic from source i. Then S_i and D_j define the subtmatrix $M(S_i, D_j)$ that will be used to predict whether $M(i, j)$ is a true or a false zero.

The intuition behind this method is as follows. Referring back to Figure 1, imagine a set of sources $\tilde{S} = \{s_1, s_2, ...\}$ that have similar behavior with respect to the observer. That is, paths from sources $\tilde{S}$ to an arbitrary destination d all either go through the observer, or not. Then the pattern of 1s on the rows $s \in \tilde{S}$ of M will be similar. Likewise, if there are destinations $\tilde{D}$ with similar behavior, the patterns of 1's in the $\tilde{D}$ columns will be similar. Hence, by choosing S_i and D_j according to the above rules, we select a submatrix which will tend to be large and contain many 1s when the target (i, j) is a false zero.

Structural properties: There are a number of ways we might test the (i, j)-descriptive submatrix $M(S_i, D_j)$ to see whether it is large and contains a large number of 1s. These correspond to the following structural properties of $M(S_i, D_j)$:

- Size: If $|S_i| = s$ and $|D_j| = d$, the Size of $M(S_i, D_j)$ is simply $s \times d$.
- Sum: The Sum of $M(S_i, D_j)$ is the number of ones that appear in $M(S_i, D_j)$.
- Density: The Density of $M(S_i, D_j)$ is the ratio of the Sum to the Size of $M(S_i, D_j)$.

Classification criterion: The intuition behind this method suggests that when the (i, j)-descriptive submatrix is small, or contains few 1s, then (i, j) is likely a true zero; otherwise, it is likely a false zero. In practice we set a threshold β and our classifier becomes:

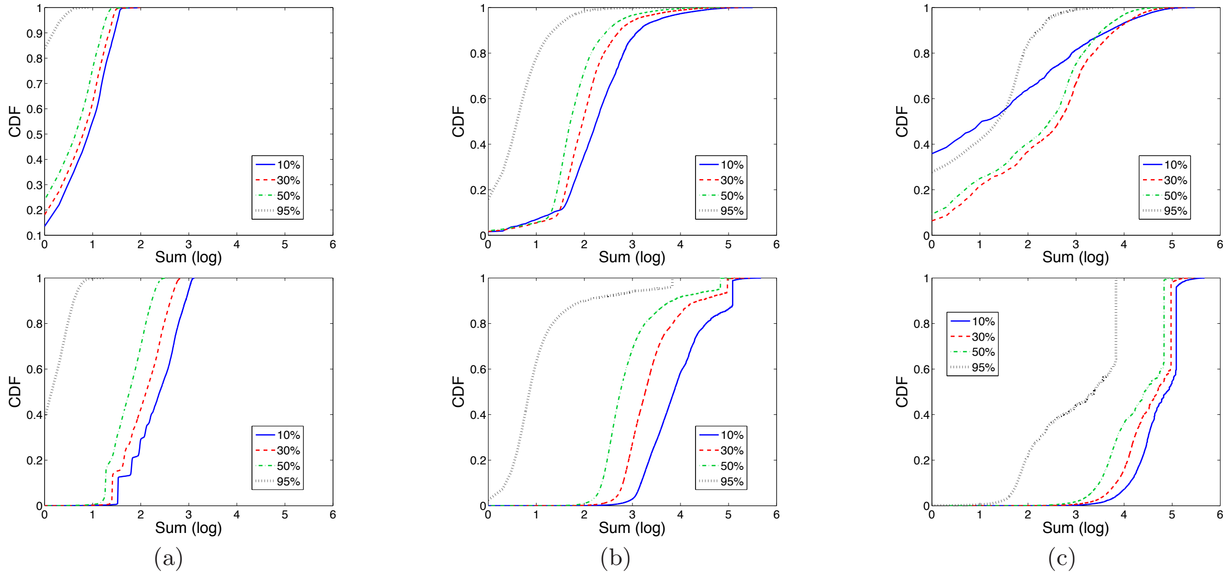
$$\widehat{T}(i, j) = \begin{cases} 1 \text{ if } \pi_{ij} > \beta & \text{(False Zero)} \\ 0 \text{ if } \pi_{ij} \le \beta & \text{(True Zero).} \end{cases} \qquad (1)$$

As already noted, in order for the classifier to be robust, there should be a significant region of β values over which TPR is close to 1 and FPR is close to 0.

4.2 Experimental results

Our experimental evaluation starts with the ground truth visibility matrices as described in Section 3. We test the classifier in each case on an equal number of true and false zeros, randomly selected – we test as many as possible while keeping the numbers equal, up to a limit of 2000 of each type. This balancing of test cases allows us to examine both TPR and FPR at the same resolution, and avoids bias stemming from the much larger set of true zeros than false zeros.

Descriptive power of the Sum property. To interpret the performance of the visibility-based method, it is helpful



(a)

(b)

(c)

Figure 3: Submatrix Sum distribution for (a) Edge, (b) Core-1000, (c) Core-100. Upper: True Zeros; Lower: False Zeros. Scale on x axis is base 10 log.

to start by looking at the Sum values for submatrices, comparing the case for true and false zeros. (We concentrate on the Sum metric for reasons explained below.) Here we use knowledge of the true and false zeros, obtained from ground truth, in order to gain insight on the method (of course, actual performance results do not use this knowledge).

Figure 3 shows the CDF of Sum values on a log (base 10) scale for true zeros (upper) and false zeros (lower). To interpret these figures in the context of our classification problem, one can visualize a classification threshold drawn as a single vertical line through both an upper and lower plot. On the upper plot, a good classifier will place the majority of the distribution to the left of the line (yielding low FPR); on the lower plot, the majority should be to the right of the line (yielding high TPR).

The figure shows that the best case is for the Edge networks, shown on the left. Here it can be seen than there is a significant range of thresholds – values between about 10 and 50 – which almost completely separate the true and false zeros, in all cases except the extreme 95% case. Even in the 95% case, there is a significant opportunity for separating the two classes (at a different threshold).

The Core-1000 networks are also generally separable at certain thresholds, but the Core-100 networks less so. Examining the reasons for the difference between the Edge and Core networks, we find the following. Networks in the Edge class are typically 'stubs' in the AS graph (or close to stubs). The paths passing through a stub network are typically only those with sources or destinations in the set of prefixes that are announced by the network. Likewise, paths passing through a network that is near the edge, but not a stub, are primarily those that have sources or destinations announced by an AS in the network's customer cone. These situations correspond to a visibility matrix that is quite sparse, but having a few rows and columns that are almost completely filled.

In such cases, the Sum and Size properties behave similarly: for a true zero, both return very small values (Figure 3(a) upper) while for a false zero, both capture the 1s in the same row and column as (i, j) resulting in large values (Figure 3(a) lower). This explains why we only show results for Sum – the results for Size are very similar, and the results for Density are poor because the density is nearly constant.

In contrast, for the highest-tier networks (Core-100, Figure 3(c)) the arrangement of 1s in M is much more complex because of the spreading of paths through the network core. In such an AS, many pairs are visibile, so the likelihood of unrelated pairs having similar visibility is much higher than for Edge networks.

Classification accuracy. The differences between Edge and Core networks are reflected in the performance of the classifier. Figure 4 shows an 'aggregate ROC' curve across all networks in each set. The aggregate ROC is a composite constructed by collecting results for all (i, j) pairs tested in all networks. While this curve does not reflect the performance of any actual network (those results are next), it serves to give an overall sense of the classifier's performance.

The results show that it is possible to achieve excellent performance in the case of the Edge networks. Figure 4 shows a TPR of over 90% when FPR is zero, and an FPR of about 20% when TPR is 100%, for moderate flipping percentages. Thus, either the true or the false zeros can be labeled essentially perfectly, with small error for the other class. However, the picture is not so good for Core-1000 and Core-100 networks. For those networks, achieving TPR greater than about 95% requires a relatively high FPR, as high as 50% or more, even for moderate flipping percentages.

To give better insight into how the classifier performs in individual networks, we turn to Figure 5. In the Figure the AUC metric is used to judge the classifier's performance on each network, and the resulting CDF of AUC across all networks is presented. The Figure shows that for moderate flipping percentages the AUC for Edge networks is almost

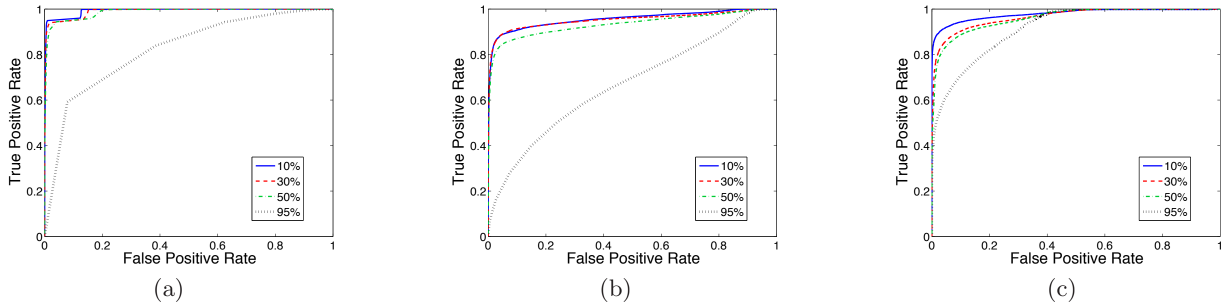


Figure 4: Aggregate ROC Curves for (a) `Edge`, (b) `Core-1000`, (c) `Core-100`.

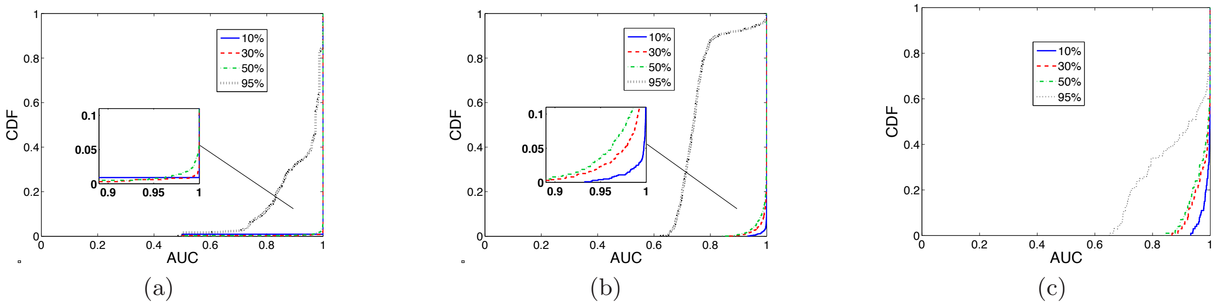


Figure 5: CDFs of AUC for (a) `Edge`, (b) `Core-1000`, (c) `Core-100`.

always very close to 1, while for `Core-1000` and `Core-100` networks, up to half of the AUCs are less than 1.

These results show that for `Edge` networks, the visibility method is quite accurate. However, the same is not true for `Core` networks; such networks apparently require a more sophisticated approach. We develop such an approach in the next section.

5. THE PROXIMITY-BASED METHOD

The challenge presented by `Core` networks is that observed visibility patterns are not sufficiently helpful in finding source-destination pairs that are routed similarly to the (i, j) target. In this section we develop a new distance metric for prefixes (sources and destinations) that helps infer visibility in such networks. We compute this metric using the limited amount of BGP state observable in publicly available datasets (as described in Section 3). We refer to this approach as the *proximity* method.

5.1 Routing-state distance

Intuitively, we seek a measure of distance (or dissimilarity) between prefixes. Our intent is that if the distance between prefixes is low, then we expect that paths to or from those prefixes have a similar visibility status for an arbitrary observer.

A natural measure of distance in this setting would be *hop distance.* Let prefixes p_1 and p_2 be announced by AS_1 and AS_2 respectively. Then the hop distance between p_1 and p_2 is simply the length of the path (e.g., shortest path) between AS_1 and AS_2 in the AS graph.

Unfortunately, this is a poor metric to use. One way to see this is simply to note that most ASes are very close in hop distance. For example, Figure 6(a) shows the distribution of hop distances for a random sample of 1000 AS-AS pairs in our data. Almost half of the hop distances are 1, 2, or 3. Such a metric clearly hides significant routing differences across prefix pairs.

What is needed is a metric that measures whether two prefixes are 'routed similarly in general.' To that end we define *routing-state distance (RSD).* First we define RSD formally, then we describe how we compute it in practice.

RSD: Definition. We start with a connected graph $G = (V, E)$. We make the following assumption: for each source-destination pair $(x_1, x_2) \in \{V \times V\}$ we assume that there is a unique node $x_3 = \mathit{nexthop}(x_1, x_2)$, and that by following the $\mathit{nexthop}(\cdot, x_2)$ function recursively, one will eventually reach x_2. We assume as well that $\mathit{nexthop}(x, x) = x$. Thus, $\mathit{nexthop}(x_1, x_2)$ is the next node on the unique path from x_1 to x_2; and that path stops at x_2.

We then define:

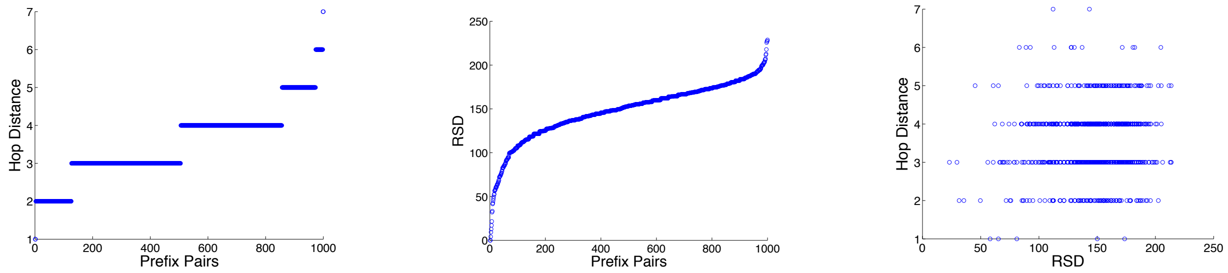
$$\mathit{routestate}(x) = \\ \langle \mathit{nexthop}(x_1, x), \mathit{nexthop}(x_2, x), ..., \mathit{nexthop}(x_{|V|}, x)\rangle$$

and:

$$\mathit{rsd}(x_1, x_2) = \#\{x_i \mid \mathit{nexthop}(x_i, x_1) \neq \mathit{nexthop}(x_i, x_2)\}$$

Intuitively, $\mathit{routestate}(x)$ tells us what the 'direction' is to x from each node in the graph. Further, $\mathit{rsd}(x_1, x_2)$ is the number of positions where the vectors $\mathit{routestate}(x_1)$ and $\mathit{routestate}(x_2)$ differ. Thus, two nodes which appear to most other nodes to be in the same 'direction' are considered close under the *rsd* metric.

Referring again to Figure 1, assume that the next hop from d_1 to each s_i is AS1, and that the next hop from d_2 to each s_i is AS2. Then the nodes s_1, s_2, s_3 would be considered close to each other under the *rsd* metric. That is because d_1 considers all the s_i to be in the same direction, as does d_2.



(a) (b) (c)

Figure 6: Distance distribution: (a) hop distance (b) $\widehat{rsd}$ (c) Comparison.

RSD has a number of good properties. It is in fact a metric (it is symmetric, positive definite, and obeys the triangle inequality). Further, it can be applied to many different situations, as long as the notion of a unique next hop is satisfied (as for example when using shortest paths, or for many routing situations). And, unlike hop-distance, it is fine-grained – taking on a wide range of values (from 0 to $|V|$).

RSD and BGP. We will compute the RSD between pairs of ASes, and then extend RSD to prefixes by noting the AS that announces the prefix. The general idea is to use the (comparatively small) set of publicly observable BGP paths to compute RSD. However, computing RSD from BGP data raises some implementation considerations. There are two main issues: (1) we cannot observe the *nexthop* function for every AS pair in the Internet; and (2) for some AS pairs the *nexthop* function is not uniquely defined.

The first issue concerns the fact that the publicly available BGP data consists (essentially) of paths from a set of monitor ASes to a large collection of prefixes. For any given AS pair (x_1, x_2), these paths may not contain information about $nexthop(x_1, x_2)$. We address this by *approximating* RSD. We make the following observation: some ASes have a much larger impact on RSD than others. For example, a stub AS has a highly uniform contribution to each *routestate* vector. If the stub AS x has a small set of providers, then for many AS pairs (x_1, x_2), $nexthop(x, x_1) = nexthop(x, x_2)$. Hence *most ASes contribute little information to RSD.*

Thus, we conclude that we can approximate RSD using a subset of all ASes, in particular those ASes with many neighbors; these ASes contribute the most information to RSD. We call these ASes the *basis set.* We select the basis set by identifying 77 ASes with the largest number of neighbors in our data. Happily, such ASes tend to appear on many paths in the publicly available data; hence we can often find $nexthop(x_1, x_2)$ when x_1 is in the basis set.

To address the case when $nexthop(x_1, x_2)$ is not available (for AS x_1 in the basis set), we performed extensive studies of the effect of missing *nexthop* information on RSD. Space does not permit a full summary of our findings; we note only that *proportional approximation* yields results that work well in practice. This approximation extends the *routestate* vector to include 'don't know' elements. We then define $\widehat{rsd}(x_1, x_2)$ as the fraction of *known* positions in which $routestate(x_1)$ and $routestate(x_2)$ differ, times the number of ASes in the basis set. This normalizes $\widehat{rsd}$ so that it always ranges between zero and the size of the basis set.

The second issue is that for some AS pairs there is more than one next hop. This happens when an AS uses detailed routing policies (such as hot-potato routing) that are not strictly per-neighbor. That is, traffic destined for the same prefix may take different next hops depending on where it enters the network. We address this problem the same way as in [10], i.e., by introducing the notion of 'quasi-routers.' Using the algorithm in [10] we divide each AS in the basis set into a minimal set of quasi-routers such that for each (quasi-router, prefix) pair there is a unique next hop AS. This expands the size of the basis set from 77 to 243.

Having addressed these two issues, we can compute $\widehat{rsd}$ for each pair of prefixes in our dataset. In Figure 6(b) we show the RSD values for the same set of prefix pairs as in Figure 6(a) (sorted in increasing order in both cases). The steep slope on the left of the figure shows that RSD can make fine distinctions between prefix pairs. Furthermore, it is clear that RSD is not simply another measure of hop distance: Figure 6(c) shows a scatterplot of RSD versus hop distance, indicating that there is little relationship between the two metrics. A last observation about RSD applied to prefixes is that it can be interpreted as a generalization of the notion of BGP atoms [3]. A BGP atom is a set of prefixes which are routed the same way everywhere in the Internet; so a BGP atom is a prefix set in which each prefix pair has an RSD of zero.

5.2 Method description

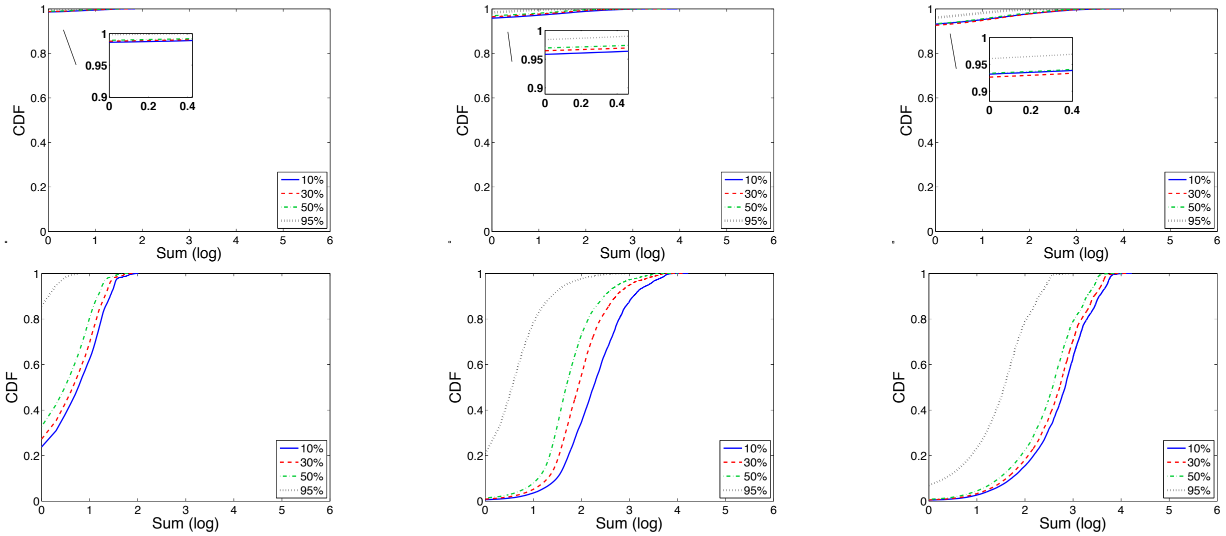
Using RSD, we again instantiate the generic method presented in Section 2.2.

Submatrix selection: Given $(i, j) \in Z$ and threshold τ, the proximity method selects the (i, j)-descriptive submatrix by selecting rows and columns from M that correspond to sources and destinations that are close to i and j respectively. That is,

$$\begin{aligned} S_i &= \{i\} \cup \{i' \mid \widehat{rsd}(i', i) \leq \tau\} \quad \text{and} \\ D_j &= \{j\} \cup \{j' \mid \widehat{rsd}(j', j) \leq \tau\}. \end{aligned}$$

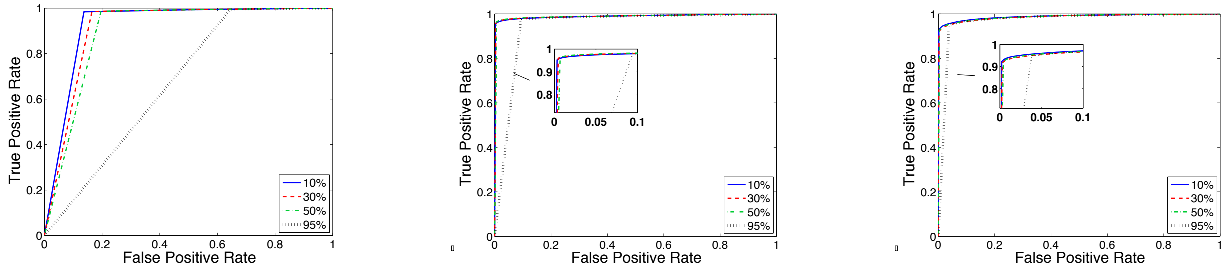
We explored various threshold values τ. Space does not permit reviewing those results, but we found that $\tau = 50$ works well in practice. As can be seen in Figure 6(b), this means that we are placing a relatively small subset of rows and columns into $M(S_i, D_j)$.

Structural properties: As in the visibility-based approach we again compute the Size, Sum and the Density of the $M(S_i, D_j)$ as input to the classifier. In this case, Sum and Density prove to be equally effective discriminating features; we report only results for Sum.



(a) (b) (c)

Figure 7: Submatrix Sum distribution for (a) `Edge`, (b) `Core-1000`, (c) `Core-100`. Upper: True Zeros; Lower: False Zeros.



(a) (b) (c)

Figure 8: Aggregate ROC Curves for: (a) `Edge`, (b) `Core-1000`, (c) `Core-100`.

Classification criterion: The intuition behind the proximity method suggests that the target element (i, j) should have similar visibility to the paths captured in the $M(S_i, D_j)$ submatrix. Both Sum and Density are large when the paths captured in $M(S_i, D_j)$ are visible; hence we use the same classification criterion as for the visibility method (Equation (1)).

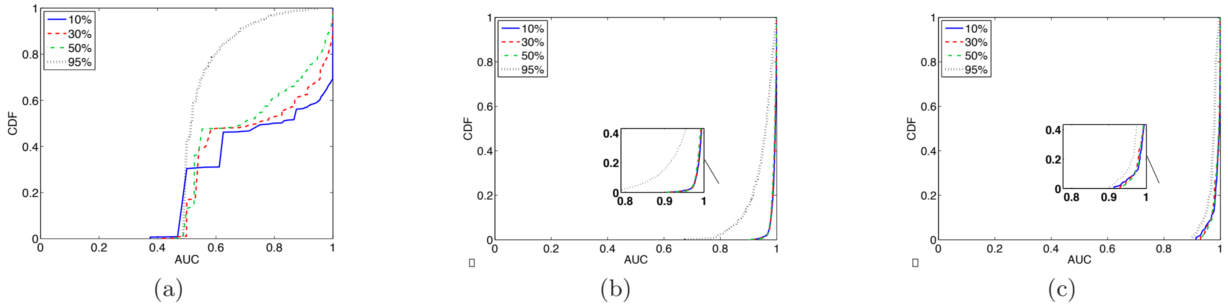
5.3 Experimental results

We again start by examining the distribution of Sum values for submatrices $M(S_i, D_j)$, which are shown in Figure 7. The upper row of plots shows that in each set of ASes, the Sum metric is almost always zero when (i, j) is a true zero (i.e., more than 95% of the time). This makes sense, since the paths captured in $M(S_i, D_j)$ are similar to the path (i, j) in terms of routing behavior. For the case of false zeros, $M(S_i, D_j)$ is typically nonzero, but with different behavior for the `Edge` versus `Core` networks. The `Core-1000` and `Core-100` true-zero submatrices are easily separated from the false-zero submatrices. For moderate flipping levels, a threshold value anywhere in the range 1-10 is effective. However for the `Edge` networks, there is no similarly good threshold.

This can be understood as follows. The proximity method is not effective for `Edge` networks because it typically selects only a small set of sources and destinations for inclusion in $M(S_i, D_j)$ – those that are close to i and j in terms of $\widehat{rsd}$. Since `Edge` networks have sparse visibility matrices in general, the small size of $M(S_i, D_j)$ makes it possible that most or all 1s are flipped to zeros, leading to false zeros that are misclassified as true zeros. On the other hand, comparison of the top and bottom of Figure 7(b) and (c) shows that RSD is able to pick out those prefixes that are similar in routing behavior when the network is in the core and has denser visibility matrices, and this is what is needed for accurate classification.

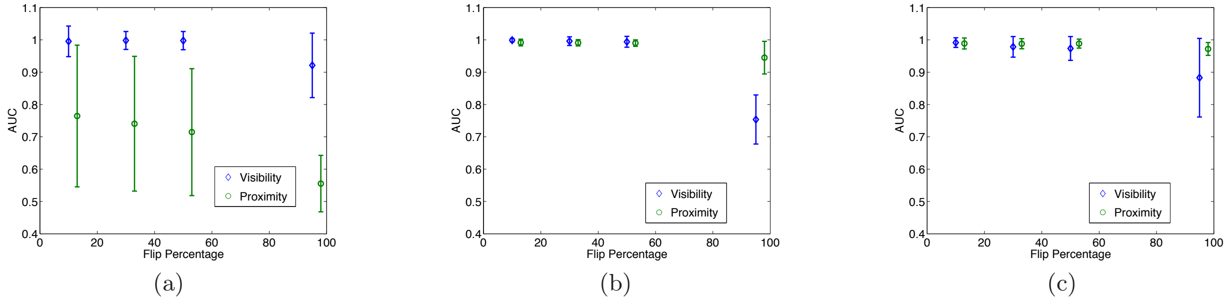
The accuracy of classification in terms of aggregate ROC curves is shown in Figure 8. The figure shows that classification in general is very accurate for `Core` networks: for moderate flipping levels, accuracy is as high as 95% TPR with zero FPR. For `Edge` networks, as expected, classification accuracy is generally poorer, with an unavoidably nonzero FPR.

Figure 9 summarizes AUCs over all networks. For `Edge` networks, a significant number of AUCs are in the 0.5 range. An AUC of 0.5 is typically characteristic of a straight-line ROC curve from the origin to the upper-right corner. This happens when there is essentially no discriminatory power in



(a) (b) (c)

Figure 9: CDFs of AUC for (a) `Edge`, (b) `Core-1000`, (c) `Core-100`.



(a) (b) (c)

Figure 10: AUC Means for (a) `Edge`, (b) `Core-1000`, (c) `Core-100`.

the classifier. On the other hand, for `Core` networks, AUCs are generally quite high, typically 0.95 or higher.

5.4 Comparison

The AUC means for both methods are shown in Figure 10. Figure 10(a) shows that for `Edge` networks, the visibility-based method is vastly preferable, and maintains an accurate classification rate even for flipping percentages as high as 95%. However, Figures 10(b) and (c) show that the visibility-based method breaks down, especially for high flipping percentages, for `Core` networks. Since the VISIBILITY-INFERENCE problem is likely to be of more interest to `Core` networks in general, this motivates the effort to develop a more refined method targeted specifically to `Core` networks.

Figure 10 shows that this more refined method (proximity-based) is in fact clearly effective in both `Core-1000` and `Core-100` networks. For those networks, even at very high flip percentages, the proximity-based method using RSD is capable of AUCs that are nearly always close to 1.

6. CLASSIFICATION

The results in Sections 4 and 5 demonstrated both methods have high TPR and low FPR for large range of threshold values. Nonetheless, in practice we need to choose a threshold value (without access to ground-truth data). Here, we show that such a value can be picked automatically, by exploiting the information hidden in the 1s of the observed visibility matrix M.

The key idea is that, intuitively, we expect the descriptive submatrices of the 1-valued entries of M to be similar to the descriptive submatrices of the false-zero entries of M. Therefore, we can use the distribution of the characteristic values of the former, in order to choose a threshold that will correctly classify the latter.

To demonstrate the effectiveness of this strategy, we perform the following experiment. For each network in `Edge`, with visibility matrix M, we choose entries $M(i,j) = 1$ and form their corresponding (i,j)-descriptive submatrices using the methods described in Section 4.1. Then, we set the classification threshold to the low end of the distribution of SUM values obtained. Based on experience with the classifier, we selected the 3^{rd} percentile of the distribution as a reasonable choice.

The leftmost part of Table 1 shows the the average (and standard deviation) of TPR and FPR of the visibility-based classifier across all the networks in `Edge` and for all four flipping percentages. The results show that the 3^{rd}-percentile rule for picking the threshold value leads to average TPR of almost 1 and very small FPR for flip percentages up to 50%. Even for 95% flipping percentage, the average TPR drops only to 0.85 and the average FPR increases only to 0.17.

The last four columns of Table 1 show the same results for the `Core-1000` and `Core-100` networks. These results are obtained by performing the same experiment in those networks, with the only difference that the (i,j)-descriptive submatrices of the 1-valued entries of M were obtained using the proximity-based method described in Section 5.2. We use the proximity-based method since it was identified to outperform visibility-based classification for these networks.

The results show again that a threshold set to the 3^{rd}-percentile of the sum of the descriptive submatrices of the 1-valued entries yields excellent average TPR and FPR for all flip percentages. Overall, these experiments confirm our intuition that the descriptive submatrices of the 1-valued entries of M are similar to the submatrices of the false zeros, and that using this information to select the classifier threshold leads to excellent prediction accuracy.

The results above are for the random bit-flipping strat-

Table 1: Mean and standard deviation of TPR and FPR for Edge, Core-1000 and Core-100; 3^{rd}-percentile Sum threshold.

	Edge		Core-1000		Core-100	
Flip %	TPR	FPR	TPR	FPR	TPR	FPR
10%	0.99 (0.062)	0.032 (0.15)	0.98 (0.033)	0.03 (0.04)	0.95 (0.067)	0.027 (0.022)
30%	0.99 (0.061)	0.045 (0.15)	0.98 (0.035)	0.03 (0.04)	0.95 (0.071)	0.028 (0.021)
50%	0.99 (0.067)	0.061 (0.17)	0.98 (0.033)	0.03 (0.05)	0.95 (0.064)	0.034 (0.025)
95%	0.85 (0.18)	0.08 (0.18)	0.98 (0.027)	0.21 (0.18)	0.96 (0.054)	0.069 (0.046)

Table 2: Mean and standard deviation of TPR and FPR for destination-based flipping on Edge and Core-100.

Edge		Core-100	
TPR	FPR	TPR	FPR
1.0 (0)	0.98 (0.11)	0.78 (0.30)	0.027 (0.17)

egy, corresponding to the case where traffic for each source-destination pair is independent. As described in Section 3, we also consider the case of correlated bit-flipping, in which all of the 1s for a specific destination (column) are flipped to zero. For each AS in the Edge and Core-100 datasets, we applied this stategy 1000 times to a randomly chosen column. We report results for only the zeros in the chosen column, because classification accuracy of zeros outside of the chosen column is not significantly changed.

Table 2 shows the resulting classification accuracy, again using the visibility-based method for Edge networks and the proximity-based method for Core-100 networks. It shows that for Edge networks, essentially all zeros are classified as true zeros. This occurs because of the nature of the visibility-based method. In that method, the 1s in the same column and rows as the zero $(i.j)$ determine the size of the submatrix. If the entire column is set to zero, the submatrix is necessarily too small to identify any false zeros. Note that classification accuracy for the rest of the matrix is not strongly affected, however.

The situation is quite different for the proximity-based method operating on Core-100 networks. Here the classifier performs quite well, with a high TPR and a FPR of almost zero. That is, even when traffic patterns are such that a destination prefix receives no traffic at all, the traffic patterns in similarly-routed prefixes are sufficient to accurately separate true and false zeros.

7. APPLICATION: TM COMPLETION

In Section 1, we described a number of reasons one might seek a solution to the Visibility-Inference problem. Here we demonstrate an example in depth: improving the accuracy of *traffic matrix completion.*

The problem. Traffic matrix (TM) completion refers to estimating traffic volumes that are not directly measurable, whether due to missing or dropped data [16], lack of visibility [2], or other reasons. The general notion builds on work in statistical signal processing [4] that has identified sufficient conditions and appropriate algorithms for estimating missing elements of a partially-observed matrix.

TM completion starts with a partially-observed traffic matrix V. The *known* (i.e., observed) entries of V are the set Ω. Using Ω one seeks to estimate the remaining, unknown elements of V. If ground truth is not available, TM completion accuracy can be assessed through cross-validation, i.e., by holding a subset $R \subset \Omega$ out as a validation set. Then, $\Omega \setminus R$ is given as input to the matrix-completion method, which predicts the entire matrix $\widehat{V}$, including elements R. Accuracy is measured by the Normalized Mean Absolute Error (NMAE) on the validation set R:

$$\text{NMAE}(R) = \frac{\sum_{(i,j)\in R} |\, V(i,j) - \widehat{V}(i,j)\,|}{\sum_{(i,j)\in R} V(i,j)}.$$

Prior work on TM completion has been forced to treat zeros in V as missing values (e.g., [2]). This is because, as we have discussed throughout this paper, lack of observed traffic for an element (i,j) is ambiguous: it may or may not reflect a valid traffic measurement. Thus, a large amount of information – all false zeros in V, which represent valid traffic measurements – is thrown away.[2] We demonstrate the significance of this loss, and the improvement possible with our classification methods, using LMaFit [15], a well-known matrix-completion algorithm.

Data. We use the traffic matrix V described in Section 3; it comes from a large Tier-1 provider that is a member of the Core-100 set. We also use the corresponding ground-truth matrix T for the same network, for validation (only). The only preprocessing we perform is to remove any rows and columns that are fully-zero in V, since it is well understood that matrix completion cannot estimate elements in such rows and columns. After this filtering step, we wind up with versions of V and T having 28 rows (source ASes) and 6198 columns (destination prefixes); V has a density of about 4%.

Improving TM completion accuracy. To evaluate the benefit of solving Visibility-Inference in this setting, we consider four possible cases for TM completion:

Ground Truth (GT): TM completion using perfect knowledge of false zeros (valid zero-valued traffic measurements). So for GT the set of known entries given to LMaFit consists of $\Omega \setminus R$ plus the known false zero entries as given by T.

Visibility (Vis): TM completion after labeling false zeros via the visibility-based method. So for Vis the set of known entries given to LMaFit consists of $\Omega \setminus R$ plus the false zeros identified by the visibility-based method.

Proximity (Prox): TM completion after labeling false zeros via the proximity-based method. Input to LMaFit

[2] We extend the notion of true and false zeros to V by identifying them with the true and false zeros of M. This means that a false zero in V is actually a *valid* traffic measurement; a true zero is not.

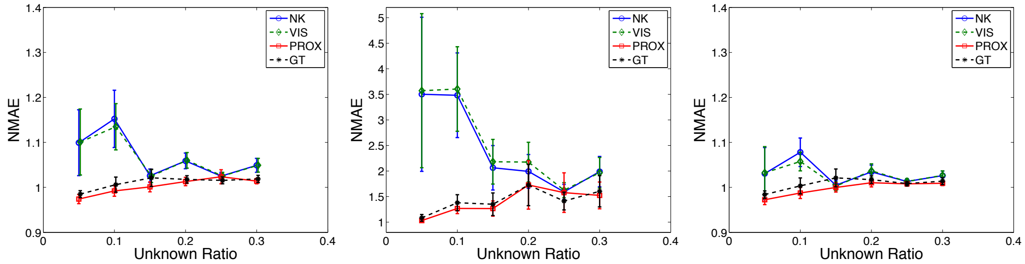


Figure 11: Accuracy of matrix completion for GT, Vis, Prox and NK for (a) all, (b) small, (c) large unknown elements in R.

is $\Omega \setminus R$ plus the false zeros identified by the proximity-based method.

No Knowledge (NK): TM completion with no knowledge of false zeros, i.e., as it has been done previously in the literature. In this case, the set of known entries given to `LMaFit` consists only of $\Omega \setminus R$.

Figure 11 shows the results, in which the fraction of Ω that is held out in R is varied from 0.05 to 0.3 (denoted on the plots as the *unknown ratio*). All results are averages over 20 different cross-validation sets R. We expect the results to depend on the size of the TM elements being estimated; additional knowledge of zero-values has a bigger influence on estimation of small elements. Hence, in Figure 11(a) we report results for all values in R, while in Figure 11(b) we show results for small values (entries smaller than the median) and in Figure 11(c) we show results for large values.

The results show that knowledge of false zeros can significantly improve the accuracy of TM completion. Comparing NK and GT one sees that accuracy can be improved as much as a factor of 3.5 when predicting small values. With respect to our classifiers, as we expect, Prox outperforms Vis since this is a `Core-100` AS. Most significantly, we observe that labeling false zeros using Prox yields improvements that are essentially the same as those obtained using ground truth.

8. DISCUSSION AND RELATED WORK

To the best of our knowledge, we are the first to define and address the Visibility-Inference problem, whose goal is inferring visibility from traffic. Therefore, we are not aware of any existing methods that solve exactly the same problem. Despite the unique question prompting our work, our methods have connections to much previous work. Here, we discuss those connections as well as the limitations of our approach.

BGP paths. The analysis of the properties of observed BGP paths has a long history, initially enabled by the Routeviews project [14]. Our work draws on the notion of BGP Atoms [3], which resulted from the observation that some sets of prefixes are routed identically everywhere in the Internet. The specifics of how routing policies affect observed BGP paths have been studied extensively [10]; these studies informed our work and guided our adaptation of RSD to BGP data.

Data Limitations. It is well-known that publicly available BGP tables provide an incomplete view of the AS graph. This has been reported widely, and recently reviewed in [13]. In this regard, it is important to note that our results are not based on the AS graph; in fact, the way we use BGP data does not introduce ambiguities due to missing links. As we describe in Section 3, we construct visibility matrices over only the set of sources and destinations for which we have every active path between a source and a destination. Thus, we omit source-destination pairs for which our BGP data may be missing links, and we can have high confidence in the 0-1 status of each element of each ground-truth T matrix. While we cannot rule out inaccuracies due to configuration errors, false BGP advertisements, or path changes that have not yet reached the monitors, we believe that these issues have negligible effect on our results. In the few cases where we discuss the AS graph (Section 3) we only take from it qualitative, not quantitative observations.

The experiments we perform make the assumption that paths visible in BGP match those that are actually taken by traffic. It is known that this is not always the case [8]. However, as [8] notes, “the two AS paths usually match.” That paper shows that mismatches are rare — a few percent at most. Hence we don’t expect that these mismatches dramatically change our results.

Scaling Considerations. As noted in Section 2, we consider observers to be ASes rather than ISPs. We don’t believe this has a major impact on our results. While ISPs can certainly merge visibility and traffic information obtained from multiple ASes that they operate, such merging does not seem to fundamentally change the nature or difficulty of the Visibility-Inference problem. On the other hand, studying visibility at the AS level is natural because the AS is the granularity at which BGP selects routes.

Another issue concerns the scale of the full problem. The matrices we work with, although they have over 5 million elements, are small compared to a full Internet-wide matrix. A conservative estimate of of the size of a full AS-prefix matrix is on the order of 5 billion elements. For our Tier-1 provider, the fraction of nonzero traffic elements in V is about 0.1%. On the other hand, that provider’s data was based on a sampling rate of 1/1000; hence very many small elements of V were lost due to the sampling process. So for the case of this provider, we can estimate that a day’s traffic occupies on the order of 5 million nonzero elements in V, and probably much more.

It is not clear the extent to which routing changes affect the estimation process. The notion of visibility itself can be attributed to an instant in time, during which routing changes are assumed to be negligible. However, the need to collect traffic to populate M introduces sensitivity to routing changes. The effect of route changes is to inflate the number of 1s in M. The sensitivity of the inference process to this inflation needs more study; we only note here that a small fraction of prefixes are responsible for most route changes; and those prefixes receive comparatively little traffic [11].

Additional Approaches. Finally, we ask whether there are other ways ISPs could attack this problem.

One direction, as mentioned in Section 2, is for ISPs to make note of the routes that they learn through BGP. How many entries in $\widehat{T}$ could the observer fill in using BGP-learned routes? From BGP, the observer learns a path to each destination from each of its neighbors. Each path as well contains sub-paths. This implies that the number of $\widehat{T}$ elements that can be learned in this way for any given destination is equal to the number of unique ASes found in all neighbor paths to the destination. Experience with BGP suggests that this is not usually a large number, except for the small set of ASes with very high degree. For example, for each of the Routeviews monitors, the average number of elements per destination that can be learned by such a method is ∼40. In comparison, for ASes in the `Core-1000` set, the average number of elements per destination potentially visible through traffic observation is ∼500. Thus, we believe that BGP-learned paths are likely to add only incremental improvement to estimation of $\widehat{T}$.

Another approach would be for ISPs to use information about the *volume* of traffic observed. The general idea is to assume a distributional model for traffic – one which includes a nonzero probability that a traffic element is zero. One then models the observed data as a mixture of the values taken from two sources: the assumed traffic distribution (which generates false zeros), and an additional source of (true) zeros. Such 'zero-inflated' models are used, for example in ecology, in settings loosely analogous to the VISIBILITY-INFERENCE problem [9]. Taking this approach requires imposition of modeling assumptions, with all the difficulties that accompany it: addressing the model selection problem, estimating parameters, and assessing confidence in the results. However the potential exists to estimate the *number* of false zeros via this sort of method. While this approach presents many hurdles, a considerable amount of theory has been developed around how to do this in various settings, and some issues relevant to traffic models are being addressed [6].

9. CONCLUSIONS

We started from the simple question 'what routes pass through a network?' We showed that using the network's traffic data to answer this question is equivalent to identifying source-destination pairs that are *not* communicating at a given time.

Answering these questions prompted us to look for ways to identify sets of source-destination paths that are *routed similarly in general.* This can be thought of as an considerable generalization of the notion of BGP atoms: rather than groups of prefixes that are routed *identically*, we look for groups of paths that are routed *similarly*. Surprisingly, we show that such groups of paths can be identified in a large set of representative locations in the Internet.

A key enabling idea has been the definition of a new distance metric for network prefixes: routing-state distance. Using it we were able to extrapolate from the relatively small amount of information available in publicly accessible BGP tables to estimate routing similarity between any two prefix pairs in the Internet.

We realized these ideas in the form of a family of classifiers; applying these classifiers to traffic measurements we showed that one can generally answer our motivating question with a high degree of accuracy.

While a number of challenges remain and considerable additional work is warranted, we are hopeful that the methods we develop here can improve knowledge of route visibility and further inform both operators and researchers.

Acknowledgements. This work was supported by NSF grants CNS-1017529, CNS-0905565, CNS-1018266, CNS-1012910, CNS-1117039, by a GAANN Fellowship, and by grants from Microsoft, Yahoo!, and Google. The authors thank the SIGCOMM referees and shepherd for their help in improving the paper.

10. REFERENCES

[1] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani. K-core decomposition: a tool for the visualization of large scale networks. Technical report, Arxiv, 2005.

[2] V. Bharti, P. Kankar, L. Setia, G. Gürsun, A. Lakhina, and M. Crovella. Inferring invisible traffic. In *CoNEXT*, Philadelphia, PA, 2010.

[3] A. Broido and k. claffy. Analysis of RouteViews BGP data: policy atoms. In *NRDM*, 2001.

[4] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Found. Comput. Math.*, 9(6):717–772, 2009.

[5] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, and E. Shir. A model of internet topology using k-shell decomposition. In *PNAS*, 2007.

[6] D.-L. Couturier and M.-P. Victoria-Feser. Zero-inflated truncated generalized Pareto distribution for the analysis of radio audience data. *Annals of Applied Statistics*, 2010.

[7] T. Fawcett. An introduction to ROC analysis. *Pattern Recogn. Lett.*, 2006.

[8] Z. M. Mao, J. Rexford, J. Wang, and R. H. Katz. Towards an accurate AS-level traceroute tool. In *Proceedings of ACM SIGCOMM*, August 2003.

[9] T. G. Martin, B. A. Wintle, J. R. Rhodes, P. M. Kuhnert, S. A. Field, S. J. Low-Choy, A. J. Tyre, and H. P. Possingham. Zero tolerance ecology: improving ecological inference by modelling the source of zero observations. *Ecology Letters*, 2005.

[10] W. Mühlbauer, S. Uhlig, B. Fu, M. Meulle, and O. Maennel. In search for an appropriate granularity to model routing policies. In *SIGCOMM*, 2007.

[11] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang. BGP routing stability of popular destinations. In *SIGCOMM Workshop on Internet measurment*, 2002.

[12] RIPE routing information service raw data project. `http://www.ripe.net/data-tools/stats/ris/ris-raw-data`.

[13] M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush. 10 lessons from 10 years of measuring and modeling the internet's autonomous systems. *IEEE Journal on Selected Areas in Communications*, 2011.

[14] University of Oregon route views project. `http://www.routeviews.org/`.

[15] Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. Technical report, Rice University, 2010.

[16] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu. Spatio-temporal compressive sensing and internet traffic matrices. *SIGCOMM Comput. Commun. Rev.*, 2009.