Stable and Practical AS Relationship Inference with ProbLink

Yuchen Jin, *University of Washington;* **Colin Scott,** *UC Berkeley;* **Amogh Dhamdhere,** *CAIDA;* **Vasileios Giotsas,** *Lancaster University;* **Arvind Krishnamurthy,** *University of Washington;* **Scott Shenker,** *UC Berkeley, ICSI*

https://www.usenix.org/conference/nsdi19/presentation/jin

This paper is included in the Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI '19).

February 26–28, 2019 • Boston, MA, USA

ISBN 978-1-931971-49-2

Open access to the Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI '19) is sponsored by

NetApp®

Stable and Practical AS Relationship Inference with ProbLink

Yuchen Jin[†] Colin Scott[‡] Amogh Dhamdhere[◇] Vasileios Giotsas[⋆]
Arvind Krishnamurthy[†] Scott Shenker[‡]
[†]*University of Washington* [‡]*UC Berkeley and ICSI* [◇]*CAIDA* [⋆]*University of Lancaster*

Abstract

Knowledge of the business relationships between Autonomous Systems (ASes) is essential to understanding the behavior of the Internet routing system. Despite significant progress in the development of relationship inference algorithms, the resulting inferences are impractical for many critical real-world applications, cannot offer adequate predictability in the configuration of routing policies, and suffer from inference oscillations. To achieve more practical and stable relationship inference, we first illuminate the root causes of the contradiction between these shortcomings and the near-perfect validation results for `AS-Rank`, the state-of-the-art relationship inference algorithm. Using a "naive" inference approach as a benchmark, we find that available validation datasets over-represent AS links with easier inference requirements. We identify which types of links are harder to infer and develop appropriate validation subsets to enable more representative evaluation.

We then develop a probabilistic algorithm, `ProbLink`, to overcome the challenges in inferring hard links, such as non-valley-free routing, limited visibility, and non-conventional peering practices. `ProbLink` reveals key AS-interconnection features derived from stochastically informative signals. Compared to `AS-Rank`, our approach reduces the error rate for all links by 1.6× and, importantly, by up to 6.1× for various types of *hard* links. We demonstrate the practical significance of our improvements by evaluating their impact on three applications. Compared to the current state-of-the-art, `ProbLink` increases the precision and recall of route leak detection by 4.1× and 3.4× respectively, reveals 27% more complex relationships, and increases the precision of predicting the impact of selective advertisements by 34%.

1 Introduction

The Internet, often referred to as a "network of networks", is composed of more than 60,000 Autonomous Systems (ASes). These ASes co-operate via the Border Gateway Protocol (BGP) to exchange routing information and obtain global reachability. The connections between ASes are shaped by business contracts that determine the economics and technical aspects of traffic exchange. For over 15 years, researchers have studied the problem of inferring the different types of relationships between ASes from publicly available BGP routing data. Relationship inferences are used for a wide range of applications and areas of research, such as detecting network congestion [13], identifying malicious ASes [36, 12], deploying incentive-compatible BGP security mechanisms [23, 11], protecting the integrity of anonymization [47, 34], optimizing video streaming [38, 18, 30], and understanding Internet governance and the ramifications of public policy proposals [39, 40, 30].

In this paper, we revisit the AS relationship inference problem. We find, as others have, that available relationship inference algorithms perform poorly in many critical applications [44, 6, 47, 43, 57]. We seek to understand why state-of-the-art algorithms are insufficient, despite extensive validation that indicates an error rate as low as 1%. In particular, we consider the sophisticated `AS-Rank` technique [41] which is carefully crafted using eleven deterministic heuristics. As a first step in assessing the performance of `AS-Rank`, we create a baseline benchmark algorithm, `CoreToLeaf`, that consists of three simple steps and only assumes valley-free paths through a core set of transit-free ASes. In spite of its simplicity, `CoreToLeaf` achieves accuracy that is almost as high as that of `AS-Rank`. At the same time, we evaluate `CoreToLeaf` and `AS-Rank` in practical applications—detection of route leaks and the analysis of selective advertisements—which reveals that the performance of both algorithms falls short of the needs of those applications.

`CoreToLeaf`'s high accuracy against the validation datasets implies that the majority of AS-links in the validation datasets are relatively easy to infer. Yet the sub-optimal performance of both algorithms in practical applications indicates that the small minority of AS-links that are difficult to infer are crucial for those applications. We select subsets of the validation dataset that contain AS links that we consider *hard*, and find that both the `CoreToLeaf` and `AS-Rank` techniques have substantially lower accuracy on these validation subsets (confirming where these current algorithms fall short).

We next examine the challenges in developing a more accurate AS relationship inference algorithm. We observe first that the attributes of a link (and those of the paths that traverse the link) that might be used by an AS-relationship inference algorithm are noisy and often have only a weak correlation with the link's relationship type. Second, many links appear in paths that likely violate the valley-free assumption made by existing algorithms. Third, existing algorithms are sensitive to the locations of the vantage points and the order in which the link relationships are inferred. An AS relationship inference technique must address the above challenges if it is to achieve higher accuracy for hard links.

We develop a *probabilistic AS relationship inference* algorithm, `ProbLink`, to address the above issues.

ProbLink provides a framework that allows for easy integration of many noisy but useful attributes into the relationship inference algorithm. ProbLink enables us to identify a set of link attributes that take into account not only observed paths but also information gleaned from the fact that certain paths are *not* observed. ProbLink allows for links to appear in paths that violate the valley-free property but attributes a lower probability to such occurrences. ProbLink uses an iterative algorithm that repeatedly infers link types based on statistical distributions of link attributes until the inferences reach a fixed point.

Our evaluation of ProbLink show that it achieves an error rate that is better than that of AS-Rank overall by 1.7×, and achieves 1.8-6.1× better error rate for various categories of *hard* links. We find that even the small improvement in overall accuracy brought by ProbLink has a significant impact when applied to real-world applications. Compared to the current state of the art, ProbLink increases the precision and recall of route leak detection by 4.1× and 3.4× respectively, reveals 27% more complex relationships, and increases the precision of predicting the impact of selective advertisements by 34%.

2 Background and Related Work

The Border Gateway Protocol (BGP) is the mechanism used by ASes to exchange reachability information. A BGP AS path is a sequence of ASes denoting the routing path that the first AS in the path prefers to reach a destination prefix. The last AS in the path is referred to as the "origin AS" of the prefix. Each AS uses a complex decision process to select the most preferred path toward each destination prefix [9]. BGP route collection infrastructure is operated by Routeviews [4] and RIPE NCC [3]. This consists of routers that peer with ASes that volunteer to provide their BGP routing advertisement feeds for research or operational reasons. A route collector is configured to obtain the *best paths* from ASes it peers with, the most preferred available path towards each destination starting from that AS. Route collectors typically peer with several ASes, and thus obtain multiple best paths to each destination prefix.

AS relationships fall into two broad categories: customer-provider (c2p) and settlement-free peering (p2p). In a c2p relationship, the customer AS pays the provider AS for reachability to/from the rest of the Internet. In a p2p relationship, two networks agree to exchange traffic destined to prefixes they or their customers own without an associated fee. In practice, AS relationships can span a spectrum of types between c2p and p2p. These *hybrid or complex relationships* can occur when two ASes have multiple contractual agreements, one for each geographical region where an interconnection exists [27]. *Sibling* relationships exist between distinct ASes that are owned by the same organization and can exchange traffic without any cost or routing restrictions.

The *customer cone* of an AS X is the set of ASes that X can reach using only p2c links. The size of the customer cone is an indication of the market power of an AS. A clique of *Tier-1 ASes* at the top of the Internet AS hierarchy are "transit-free", meaning that they have routes to all other networks on the Internet through customer or peering links without the need to pay for transit.

2.1 AS Relationship Inference Techniques

Beginning with the seminal work by Gao [21], most AS-relationship inference algorithms are based on the assumption that valid BGP paths are *valley-free*, i.e., a path consists of zero or more c2p links, followed by zero or one peering link, followed by zero or more p2c links. This assumption captures the economic incentives that (at least partially) determine traffic exchange between ASes: an AS should not intentionally advertise routes learned from a peer or provider to another peer or provider, since this "free transit" increases infrastructure costs but provides no remuneration.

Another observation made by Gao and others [55, 14, 15, 61] is that providers usually have a higher *node degree* (i.e., the number of ASes to which an AS node directly connects to) than customers, while peers usually have similar degrees. *Node degree* is the number of neighbors an AS directly connects to, irrespective of whether the neighbors are providers, peers, or customers Willinger et al. have shown that node degree is significantly biased by the fact that the available topological data reveals only a subset of the complete Internet topology, due to limited placement of vantage points adjacent to peer-to-peer AS links [60].

The state-of-the-art AS relationship inference technique, called the "AS-Rank" algorithm [41], makes three generally accepted assumptions: 1) there is a clique of large transit providers at the top of the hierarchy, 2) most customers purchase transit in order to be globally reachable, and 3) there are no cycles of p2c links. The AS-Rank algorithm takes 11 intricate steps to label each link as customer-provider (abbreviated as c2p or p2c depending on the directionality of the relationship) or peer-to-peer (p2p). An abbreviated version of the AS-Rank algorithm is available in Appendix A for ease of reference.

It is worth noting a few properties of the AS-Rank algorithm. First, AS-Rank uses the *transit degree* attribute as one of the main sources of information in determining relationship labels. *Transit degree* is the number of ASes that appear on either side of an AS in adjacent links of BGP paths, but it does not count neighbors for which the given AS does not transit traffic. Transit connectivity is easily observable by Route Collectors (except for backup or partial-transit links [29]), therefore it provides a more robust metric to describe an AS's prominence than node degree. Second, AS-Rank considers ASes and links in a specific order, using the *transit degree* information in certain cases (step 5) and not in others (step 7).

3 Input Datasets

3.1 BGP Paths

We collect BGP paths towards IPv4 prefixes from RouteViews [4] and RIPE RIS [3]. In September 2018, both projects operated 22 collectors, which in total connect with more than 1,000 vantage points (VP) worldwide. Each RouteViews and RIPE RIS collector dumps a snapshot of their Adj-RIB-out tables every 2 hours and every 8 hours respectively. For the purpose of evaluating the various algorithms over longitudinal data (as discussed in §4 and §7), we consider snapshots of BGP paths on the first day of April, August, and December (i.e., every four months) since 2006.

After collecting BGP paths, we parse them to remove duplicated ASes that result from BGP path *prepending*. We also filter out paths with AS loops, i.e., when an ASN appears more than once and is separated by at least one other ASN. We also sanitize the BGP paths by removing paths containing reserved ASes [51]. Loops and reserved ASes showing up in a path are artifacts of route poisoning [8, 35].

3.2 Sibling Relationships

We use CAIDA's AS-to-organization mapping dataset [10], which is derived from WHOIS data, to identify sibling links. This dataset provides quarterly information starting from 2009. We infer links between ASes that are operated by the same organization as sibling relationships.

3.3 IXP List

ASes often establish p2p relationships over shared switching fabric provided by IXPs. To facilitate dense peering connectivity, IXPs provide BGP Route Servers over which ASes establish many-to-many (multilateral) interconnections. To enable layer-3 connectivity Route Servers typically have their own ASN, but according to best practices it should be filtered-out from the AS path since the Route Server does not participate in the routing decision process [33]. However, for debugging reasons, some IXP members append the Route Server ASN in the BGP path. We sanitize BGP paths to remove Route Server ASNs since essentially the peering links are between the IXP members, and not between the IXP and ASes. To collect a list of AS Numbers (ASNs) used by IXP Route Servers, we query PeeringDB [2] for networks of type "Route Server" and extract the ASN. We augment this list by consulting the Euro-IX IXP Service Matrix [1] and extracting the Peering LAN ASN and Route Server ASN for each IXP. There were 172 IXP ASes in this list on 12/01/2017.

3.4 Validation Dataset

AS operators frequently encode the relationship type with their neighbors directly in their prefix advertisements using BGP Communities, an optional transitive BGP attribute used to attach metadata on BGP paths. While the use of communities attribute is not standardized, many ASes publicly document the meaning of their BGP communities on websites and in IRR databases, enabling us to assemble a dictionary of BGP communities that denote relationship type. We used a dictionary of 1286 community values from 224 different ASes to construct a set of relationships from BGP data starting quarterly from April 2006 to April 2017. Similar to prior work [41], we treat this dataset as "best-effort" validation to evaluate existing inference techniques and our proposed approaches.

Date (MM/DD/YYYY)	# links in validation set	# links in total	Percentage
04/01/2012	7,833	117,872	6.6%
04/01/2013	11,644	133,459	8.7%
04/01/2014	44,875	159,678	28.1%
04/01/2015	47,036	176,791	26.6%
04/01/2016	52,931	204,309	25.9%
04/01/2017	56,326	213,441	26.4%

Table 1: Size of the validation dataset vs. all links observed from all VPs.

Table 1 shows the size of this validation dataset over the past 6 years. The coverage of our validation dataset increased from 6.6% in 2012 to about 26% of the observed links in recent years, due to the increasing popularity of BGP communities and the deployment of additional VPs that allow more communities to propagate to BGP collectors. As prior work has pointed out, links involving *Tier-1* ASes and VP ASes are over-represented, because public data on BGP communities mostly comes from large ASes [41], while communities from non-VP ASes may be stripped out during the propagation of BGP routes. However, unlike prior work, we take these biases into consideration during our evaluation.

As noted above, the use of BGP communities has become increasingly popular [24], raising the question of whether we can eventually exclusively rely on communities to extract relationships without the need for an inference algorithm. Even with prevalent use of BGP communities however, we would face two important limitations. While communities are by default a transitive attribute, in practice operators often strip out community tags before propagating advertisements to neighbors. Indeed, if all the communities in our dictionary were transitively propagated to our BGP collectors, our validation dataset should have over 58% coverage of the visible AS links. Instead, as shown in Table 1 our coverage is less than 30% for the past 6 years. A second limitation with communities is partial availability of publicly available documentation of those attributes. Despite our best efforts to maximize the number of interpretable community values via automated web scraping and text processing tools, we are only able to find authoritative documentation on the meaning of 35% of visible community values.

4 Establishing a Benchmark for Hard Links

As explained in the previous section, the evaluation dataset is extensive but biased toward specific types of links. It is important to understand if the links over-represented in the val-

idation dataset are easier to infer correctly, compared to the under-represented links, which may skew the overall evaluation results. To this end we develop `CoreToLeaf`, a very simple algorithm that allows us to understand which links are easy to infer. `CoreToLeaf` uses only the *valley-free* assumption and the list of Tier-1 ASes to infer relationships. We show that the inference accuracy of this algorithm is almost as high as that of the more sophisticated `AS-Rank` algorithm elaborated in §2.1, while the accuracy of both algorithms suffer for certain categories of links. Our findings reveal that indeed certain types of under-represented links in the evaluation dataset are harder to infer, possibly inflating the overall accuracy of past work. We address this issue by constructing distinct validation sub-datasets as benchmarks for hard links.

4.1 The `CoreToLeaf` Algorithm

`CoreToLeaf` starts by inferring a clique of Tier-1 ASes using the same inference method as `AS-Rank`. For each path that traverses a Tier-1, we skip the first link after the Tier-1 and label all succeeding links as p2c. For example, if AS_2 is a clique member in a BGP path "AS_1, AS_2, AS_3, AS_4, AS_5, AS_6", we infer links $<AS_4$ - $AS_5>$ and $<AS_5$ - $AS_6>$ as p2c. We skip inferring the relationship for $<AS_2$ - $AS_3>$ because it could either be a p2c or a p2p, but all subsequent links need to be p2c assuming that the path is *valley-free*. (Note that if AS_1 is a clique member, we would have labeled $<AS_2$ - $AS_3>$ also as a p2c link.) Finally, we label all remaining unclassified links as p2p.

In the step of labeling p2c links, a link could be labeled more than once if it shows up in multiple paths. In some cases, a link could be labeled as a p2c in some path and as a c2p when traversing a different path. We label this link as a "conflict" link when we encounter such an inconsistency.

Note that `CoreToLeaf` does not take into account degree or transit degree information, nor does it use paths that do not go through Tier-1s. This is in contrast to other traditional algorithms; for example, Gao's algorithm [21] considers all paths, identifies the AS with the highest node degree in each AS path and treats it as the top provider, and then labels AS pairs before it as c2p or sibling and AS pairs behind it as p2c or sibling. The rationale behind `CoreToLeaf` is simply that there is greater certainty that it is customer routes that are being transitively exposed to Tier-1s and that there is less likelihood of paths being exported to Tier-1s due to complex peering mechanisms.

4.2 Evaluation

We evaluate this extremely simple algorithm against our validation dataset on 04/01/2017, which contains 23,528 p2p links and 32,798 p2c links (corresponding to 26.4% of the visible topology). Table 2 compares the *precision* (true positives / (true positives + false positives)) and *recall* (true positives / (true positives + false negatives)) of `CoreToLeaf` and `AS-Rank`. Surprisingly, `CoreToLeaf` achieves high precision and recall for both p2c and p2p links (comparable to `AS-Rank`), with higher *precision* on p2c relationships (98.9% compared to 97.8%), and a small fraction of links labeled as 'conflict'.

Algorithm	p2c		p2p		Conflict
	Precision (%)	Recall (%)	Precision (%)	Recall (%)	(%)
`CoreToLeaf`	98.9	95.8	95.0	98.8	0.12
`AS-Rank`	97.8	97.5	98.8	98.9	0

Table 2: Precision and recall of `CoreToLeaf` and `AS-Rank`.

The 1.1% mistakenly inferred p2c links and the links which `CoreToLeaf` labels as 'conflict' are due to *valley-free* violation, which we quantify later in §5.2. Since the step of labeling p2c links uses just paths through Tier-1s, it fails to capture 4.2% (95.8% *recall*) of the actual p2c links. Consequently, these links are inferred as p2p in the third step and results in a 5.0% error rate for links labeled as p2p.

The accuracy of `CoreToLeaf` and `AS-Rank` seem quite high, but they perform sub-optimally when applied to real-world applications. Route leaks constitute a type of prevalent routing incident that can cause significant disruptions to Internet routing [54, 28]. In §8, we describe how we can use inferred relationships to detect route leaks and evaluate the effectiveness of `AS-Rank` inferences. Only 19% of the route leaks detected using `AS-Rank` were real route leaks, and almost 80% of the real leaks were missed. We observe relatively poor performance for two more applications we tested, as discussed in detail in §8. The high application-level error rates illustrate that a better AS relationship algorithm is needed for real-world applications.

4.3 Identifying Hard Links

The surprisingly high accuracy obtained by `CoreToLeaf` has many implications. First, it indicates that simple techniques might suffice for inferring the types of many of the links in the validation dataset. Second, it underscores the need for more comprehensive validation datasets that would be more representative of AS links beyond those associated with Tier-1 and VP ASes. Third, in the absence of more comprehensive validation datasets, one way to make progress on improving and evaluating AS relationship inference algorithms is to identify specific types of links for which the current algorithms do not work well. We therefore now attempt to extract collections of *hard* links from the overall validation dataset based on the inference performance of `CoreToLeaf` and `AS-Rank`.

We feed a large set of features of every link in the validation dataset along with information on whether a link was labeled as "inferred correctly" and "inferred incorrectly" by `CoreToLeaf` and `AS-Rank` into a *gradient boosted decision tree* [20], and calculate the feature importance for accurate predictions for the two algorithms. The feature importance calculation and results are presented in Appendix C. We extract the following five categories of "hard" links suggested

by the feature importance analysis and the CoreToLeaf algorithm.

1) Links with max node degrees smaller than 100. The feature importance analysis shows that CoreToLeaf and AS-Rank do not have high accuracy for links whose endpoint ASes both have small node degrees.

2) Links observed by more than 50 but less than 100 VPs. The feature importance analysis also reveals that links observed by at least 50 VPs but not more than 100 VPs are hard to infer correctly. The reason is that p2p links are often observed by few VPs and transit links are often observed by many VPs, so it is hard to distinguish the link types for the range in the middle.

3) Non-VP and non-Tier1 links. In general, a link that is directly connected to a VP or a Tier-1 is likely to appear in many BGP paths, and the AS inference algorithm is likely to have access to more information regarding the link. Moreover, most of our validation dataset are links that are connected to a VP or Tier-1 AS, so we want to specifically analyze the performance of inference algorithms on the "underrepresented" links in our validation dataset.

4) Unlabeled stub-clique links in CoreToLeaf. A stub AS connects with only one other AS through which it gains access to the entire Internet. A stub-clique link is a link whose one endpoint is a stub AS and the other endpoint is in the Tier-1 clique. In other words, the clique member is the only AS to which the stub AS connects. These links typically have very high transit degree difference, which is an important feature as shown by the feature importance analysis.

In CoreToLeaf, a stub-clique link <X, Y> (where X is a stub AS and Y is a clique AS) is inferred as a c2p *iff* there is a path containing an AS triplet "Z, Y, X" where Z is also a clique AS. We call the set of stub-clique links that are not inferred as c2p in the second step of CoreToLeaf (i.e., they are inferred as p2p in the later step) as "unlabeled stub-clique links".

In step 9 of AS-Rank, stub-clique links are classified as c2p by default based on the assumption that stub networks are extremely unlikely to meet the peering requirements of clique members. We believe this assumption should be revisited with the trend of "Internet flattening", as peering relationships between high-tier ASes and low-tier ASes are becoming more prevalent [22].

5) Conflicts in CoreToLeaf. Recall that CoreToLeaf labels some links as "conflicts". These links appear to behave as p2c on some paths and c2p on others, and the main reason for this is violations of *valley-free* routing. We believe that this set of links is difficult to analyze because the two endpoints are likely to have unconventional routing policies.

Table 3 shows the error rates of inferences made by CoreToLeaf and AS-Rank on each category of hard links on 04/01/2016. We observe both algorithms yield more errors than their inferences on normal links, especially on unlabeled stub-clique links. Furthermore, the fraction of every category of hard links in the validation dataset is less than that in the overall links, especially for the "Max node degree < 100" category. This indicates that the validation dataset is skewed to *easy* links. In addition to the entire validation dataset, we will use these more specific datasets for evaluating the AS inference algorithms in the subsequent sections.

Category	CoreToLeaf (%)	AS-Rank (%)	Fraction all links	Fraction validation
Max node degree <100	13.7	8.6	16.1%	1.7%
Observed by 50-100 VPs	4.7	9.3	9.9%	8.1%
Non-VP & Non-Tier1	5.3	9.0	24.2%	11.6%
Unlabeled Stub-clique	95.5	33.4	0.3%	0.1%
Conflict	100.0	8.1	0.24%	0.16%

Table 3: Error rates of CoreToLeaf and AS-Rank on *hard* links on 04/01/2016. The fraction of each category of *hard* links that is in overall links vs. in the validation dataset shows that *hard* links are underrepresented in the validation dataset.

5 Challenges With AS Relationship Inference

In this section, we identify three main challenges with AS relationship inference, and describe how they hamper existing inference techniques. This analysis helps inform the design of a probabilistic algorithm for AS relationship inference.

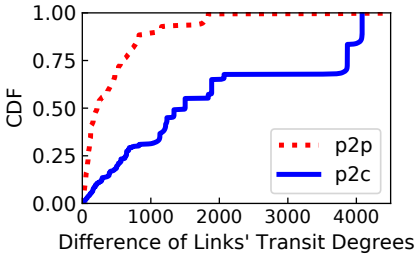
5.1 Degree Inversion

An AS inference algorithm can use any observed attribute associated with a link, its two endpoint ASes, AS links, and end-to-end paths that traverse the link in order to determine the link type. However, most attributes have stochastic information value, as we will illustrate below for AS degree.

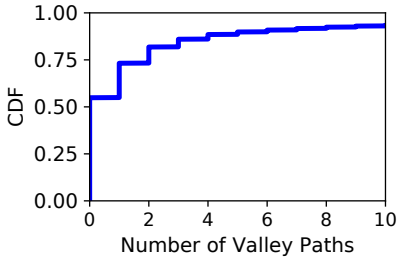
Many existing techniques for inferring AS relationships make three assumptions: highest-degree ASes sit at the top of the routing hierarchy; peering ASes have similar degrees; and providers have larger degree than customers [21, 41, 16].

Over the past four years, the top two nodes with the largest transit and node degrees (as observed through BGP feeds from available VPs) have consistently been *AS*6939 (Hurricane Electric) and *AS*174 (Cogent Communications). However, both of these ASes are not Tier-1 ASes [59], so the assumption that the ASes with the highest degrees sit on top of the routing hierarchy is not universally valid. This fact influences the accuracy of some inference approaches since a key step in these approaches is identifying a clique of Tier-1 ASes at the top of the hierarchy [21, 41].

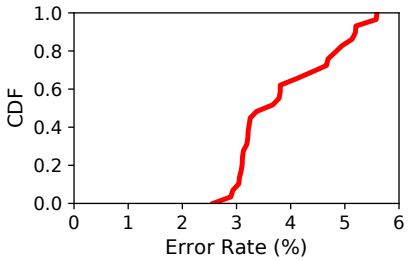
Figure 1a plots a CDF of the absolute transit degree differences of different link types. Transit degrees of ASes are computed from BGP paths observed on 04/01/2017, and link types are derived from the validation dataset on 04/01/2017 as described in §3.4. The validation dataset includes 55,016 links, 30,859 of which are p2c links and 24,157 are p2p links. We see that even though p2c links usually have larger degree differences than p2p links, over 14% of the p2p links have absolute transit degree differences larger than 1000, making many p2p links indistinguishable from p2c links in terms of transit/node degree difference.



(a) CDF of absolute transit degree difference.



(b) CDF of the number of paths that violate the valley-free property traversing each link.



(c) CDF of `AS-Rank`'s error rates on 30 consecutive 1-day BGP snapshots from April 1, 2016 to April 30, 2016.

Figure 1: Analysis of transit degree difference, valley-free violations, and error rates of `AS-Rank`.

According to this observation, the existence of substantial differences in node/transit degrees between peering ASes is common. This phenomenon is explained in part by the fact that, during recent years, large content providers such as Google, Akamai, and Microsoft, which usually have high degrees, are more willing to peer with large numbers of lower tier ASes to get free and more efficient traffic exchange [56, 31]. This trend is referred to as the "flattening" of the Internet [22], and it significantly influences the AS relationship inference techniques that differentiate peers from providers or customers based on transit/node degree differences, or rank ASes in decreasing order by degrees and label links based on the order in which ASes are considered (as is the case with `AS-Rank`).

5.2 Violation of Valley-Free Property

Next, we study the prevalence of *valley-free* violations, which is the culprit behind mistakenly inferred p2c links and 'conflict' links in `CoreToLeaf`.

3% of the BGP paths violate *valley-freeness* in the `AS-Rank` inference on 04/01/2012. We find this level of *valley-free* violations is persistent over the various snapshots in our study. Figure 1b shows a CDF of the number of paths that violate the valley-free property for the links in BGP paths on 04/01/2012. 47% of the links in the AS topology are traversed by paths that violate the valley-free property. This statistic is consistent with prior work that analyzes the prevalence of valley-free violations, and it is a result of the deliberate BGP policies of ASes that use unconventional economic models [26].

The existence of these violations has certain implications for AS relationship inference. First, a robust inference algorithm has to take into account the structure of all paths traversing a given link. Second, it might have to revisit and update the inference made for a given link after inferring the types of neighboring links.

5.3 Current Techniques are Sensitive to VP and Snapshot Selection

We observe high variation in accuracy when applying the `AS-Rank` algorithm to consecutive snapshots of BGP paths. Figure 1c plots a CDF of `AS-Rank`'s error rates ($1 - accuracy$) on 30 consecutive 1-day BGP snapshots in April, 2016. As shown in Appendix D.1, AS-Ranks accuracy is also quite sensitive to the VP selections.

The reason for the `AS-Rank` algorithm's sensitivity to snapshot and VP selections lies in the first step of its inference algorithm that identifies the Tier-1 clique and the subsequent steps that labels links in a particular order starting with the Tier-1 ASes. `AS-Rank` first finds the biggest clique from the AS-links involving the largest ten ASes by transit degree, then visits the rest of the ASes top-to-bottom, and adds an AS to the clique if it connects with all the members in the current clique. It then labels p2c links using path segments that radiate from the Tier-1 clique. Errors that creep into the clique determination step have a significant impact on the order in which AS links are analyzed and labeled. See Appendix D.2 for detailed discussions.

6 Probabilistic AS Relationship Inference

In this section, we present a new AS relationship inference algorithm, `ProbLink`, that is designed to address the challenges discussed above. First, `ProbLink` is a probabilistic algorithm that enables the use of link attributes with stochastic information value. Second, in determining a link's type, `ProbLink` simultaneously takes into account all information regarding the links and the paths that traverse it, and provides a framework for integrating conflicting information (e.g., paths that violate the valley-free property). Third, `ProbLink` does not prescribe a specific order in which ASes and links are considered, but rather continually updates the link type inferences and iterates till it reaches a fixed point in terms of the underlying stochastic distributions.

Crucially, our algorithm provides a framework for integrating various link attributes that might help infer a link's type. We therefore first design a set of link features or attributes that provide noisy but still informative signals regarding the AS relationships. In particular, we design features that capture routing behavior in terms of both observed and unobserved routes as well as integrate information regarding a link's endpoints. We note that many of the features used in our algorithm are distinct from that of prior tech-

niques, which mostly use only "valley-freeness" or "node/-transit degree" features. We then describe how we use these features to build a probabilistic inference model.

6.1 Overview

Our algorithm starts with an initial classification of links based on the inference result of `CoreToLeaf`, so each link has deterministic relationship probabilities at the beginning. More concretely, if `CoreToLeaf` labels L as a p2p link, we will convert it to $P(\mathsf{L} = p2p) = 1.0$, $P(\mathsf{L} = p2c) = 0.0$, $P(\mathsf{L} = c2p) = 0.0$ and provide that as the input to our algorithm. Note that `ProbLink` is essentially a meta-inference algorithm that can be bootstrapped by outcomes of any algorithm. Its performance is independent on the bootstrapping algorithm we choose, which we evaluate later in §7.1.

For each feature, `ProbLink` computes the conditional probability distribution based on observed data and the initial set of relationship types attributed to links. In each iteration, we update the probabilities of each link's types ($P(\mathsf{L} = p2p)$, $P(\mathsf{L} = p2c)$, $P(\mathsf{L} = c2p)$) by running our probabilistic algorithm described in §6.4, and recompute the distributions of features using the updated probability values of each link. We repeat this process until convergence, i.e., the percentage of links that change labels between each iteration drops below a small threshold.

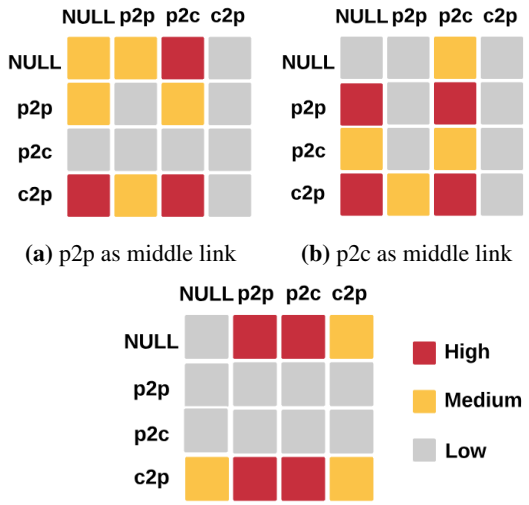
6.2 Clique Inference

We first attempt to infer the ASes that are at the top of the hierarchy, namely Tier-1 ASes, because it is used to derive features employed by `ProbLink`. Tier-1 ASes should have the largest *customer cones* [41], so estimating the customer cones is the core of doing clique detection.

First, we find top N ASes in terms of transit degree, denoted as D.[1] These ASes are either Tier-1 or Tier-2 ASes because of the large number of neighbours to which they provide transit. Then, we estimate the customer cone size of each AS in the graph by determining the average number of destination ASes (last hops) for which an element of D uses this AS as part of a route. This is an effective way of estimating the customer cone size because, irrespective of whether a node $d \in D$ is a Tier-1 or a Tier-2 AS, if it reaches a destination t through an AS $x \in D$, then t is likely to be in x's customer cone.

Second, we find the maximal clique C with largest estimated customer cone size sum in D. Then, we test every other AS in order by estimated customer cone size to add members to C. An AS is added to C if it has links with every other AS in C. If there are three consecutive members (X-Y-Z) in C showing up in paths, disconnect the edge between X and Z even though X and Z are connected in some paths, because no AS path should have three consecutive clique ASes. Finally, we find the maximal clique in C as the inferred clique.

[1] Any value of N between 10–40 does not affect the final result.



(a) p2p as middle link **(b)** p2c as middle link

(c) c2p as middle link

Figure 2: Conditional probability distribution for the triplet feature describing $P(previous, next \mid middle)$. Probability values in the ranges of > 0.1, $[0.01, 0.1]$, and < 0.01, are categorized as high, medium, and low respectively in the figure.

6.3 Feature Design

An AS link can be characterized by the following three attributes: (A) The structure of paths that use the link; (B) The structure of paths that *do not* use the link; (C) Properties of the ASes on each side of the link. We carefully design six features that correspond to these three types of attributes.

Triplet feature (Type A). The triplet feature considers link triplets that appear in paths and attributes probabilistic values for the relationships of the first and the last links given the relationship of the middle link. Suppose three consecutive links "L1 - L - L2" show up in a BGP path, where L1, L, L2 are three links (AS pairs). "L1 - L - L2" is called a link triplet. We break down each BGP path in a snapshot into link triplets, and, for the first and the last links in each path, we insert a " NULL" link in front of and behind it. For example: a BGP path "8793 6939 1103 198499" is decomposed into 3 link triplets: "NULL - <8793, 6939> - <6939, 1103>", "<8793, 6939> - <6939, 1103> - <1103, 198499>", and "<6939, 1103> - <1103, 198499> - NULL". As a consequence, each link in the BGP paths appears as a middle link in at least one link triplet. We take into account sibling relationships (described in §3.2) by skipping sibling links, i.e. treating the two ASes connected by a sibling link as a single AS, when constructing triplets.

The goal of the triplet feature is to model *valley-freeness* in a probabilistic way. For each middle link type, we compute the probability of the link type of its adjacent previous link and next link. If we put the link type of the previous link along the y-axis and the link type of the next link along the x-axis, we get a matrix view as shown in Figure 2, which is computed from `CoreToLeaf` initial labels. Each cube in

the matrix represents a probability that is categorized as high, medium, and low depending on it being in the range of > 0.1, $[0.01, 0.1]$, or < 0.01. For example, we can see from Figure 2a that when the middle link is of type p2p, the previous and next links are most likely to be <NULL, p2c>, <c2p, NULL> and <c2p, p2c>, but its previous link is unlikely to be p2c no matter what its next link's type is.

Non-path feature (Type B). In addition to observed routes, unobserved routes also provide some information regarding AS relationships. The *non-path feature* describes the probability of how many adjacent p2p or p2c links a link has, but none of them appear before this link on any of the paths. This feature is designed to capture the property that a link is unlikely to be a p2c link if it has many adjacent p2p/p2c links and none of them appear as a previous link on any of the paths containing the link.

Similar to the triplet feature, the non-path feature also models *valley-freeness* in a probabilistic way. The non-path feature is not necessarily applicable to all links. When a link does follow a p2p or p2c link or if a link does not have any adjacent p2p or p2c links, the non-path feature does not play a role in inferring the link's type.

Distance to clique feature (Type C). The *distance to clique* feature can be used to capture the observations that high-tier ASes are closer in distance (AS hops) to clique ASes than low-tier ASes, and that ASes in the same tier are likely to be peers, while high-tier ASes tend to be providers of low-tier ASes.

We first create an undirected graph by adding AS links as edges, and then compute the shortest path from each AS towards each clique member using Dijkstra's algorithm. For each AS, we compute its average distance to each member in the clique set, round it to a multiple of 0.1, and denote this value as dist(AS). We represent each link $<AS_1, AS_2>$ in the graph by a distance to clique tuple "dist(AS1), dist(AS2)".

Vantage point feature (Type C). The number of VPs observing a link also suggests the link type. The vantage point feature captures the likelihood of a certain number of VPs with at least one path traversing a particular link given its link type. This feature naturally folds in the following intuition: p2c links are more likely to be seen by more VPs compared to p2p and c2p links. This feature considers path directions. For example, let's consider a link L (<AS1, AS2>) where AS1 is the provider of AS2. `ProbLink` computes probabilities separately for both directions by counting how many paths traverse L in the direction of <AS1, AS2>, and how many paths traverse L in the direction of <AS2, AS1>

To evaluate the informational value of this feature, we analyze the number of VPs that observe a given link and correlate that with the link's type computed by `CoreToLeaf`. Figure 3 shows the CDF of the number of VPs that observe a link for each link type. We observe that 93% of p2p links and 90% of c2p links are observed by ≤ 10 VPs, while 98% of p2c links are seen by more than 10 VPs.

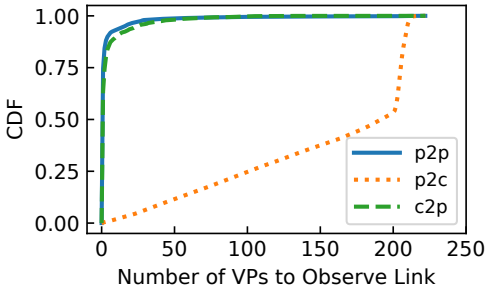


Figure 3: The visibility of each link type derived from `CoreToLeaf` inference results on 04/01/2017 BGP paths.

Co-located IXP and co-located private peering facility feature (Type C). The *co-located IXP* and *co-located peering facility facility* information is extracted from PeeringDB [2]. These features are based on the intuition that the more IXPs or facilities two ASes are co-located in, the more likely they are peering with each other. Based on the validation data, 90% of transit links do not have any co-located IXPs or facilities, while more than 70% p2p links have at least one co-located IXP or facility.

6.4 Inference Algorithm

We begin by reviewing the Naïve Bayes classifier. Given a link type variable C (which can be p2p, p2c, c2p) and a feature vector f_1 through f_n, *Bayes' theorem* states the following relationship:

$$P(C \mid f_1, ..., f_n) = \frac{P(C, f_1, ..., f_n)}{P(f_1, ..., f_n)} \tag{1}$$

By assuming that each feature f_i is conditionally independent of every other feature:

$$P(f_i \mid C, f_1, ..., f_{i-1}, f_{i+1}, ..., f_n) = P(f_i \mid C) \tag{2}$$

Using the *chain rule* to rewrite the numerator of Eq. 1:

$$P(C, f_1, ..., f_n) = P(C) \prod_{i=1}^{n} P(f_i \mid C) \tag{3}$$

So,

$$P(C \mid f_1, ..., f_n) = \frac{P(C) \prod_{i=1}^{n} P(f_i \mid C)}{P(f_1, ..., f_n)} \tag{4}$$

Since the denominator $P(f_1, ..., f_n)$ does not depend on the class C, the Naïve Bayes classifier assigns a link being a type $\hat{C}$ by the following function:

$$\hat{C} = \arg\max_{C} P(C) \prod_{i=1}^{n} P(f_i \mid C) \tag{5}$$

The inputs to `ProbLink` are BGP paths, link triplets extracted from these BGP paths, and initial relationship labels for each link as inferred by a bootstrapping algorithm. Algorithm 1 shows the pseudocode of `ProbLink`.

Algorithm 1: `ProbLink`: probabilistic AS relationship inference algorithm based on Naïve Bayes

Input : 1) BGP paths → link triplets
2) Initial AS relationships R
3) Feature vector f = [Triplet, Non-path, Distance to clique, VP, Co-located IXP, Co-located facility]
Output: Inferred probabilities of each link being p2p, p2c, c2p

```
/* Loop until convergence                                          */
1  while R − last(R) > ε do
       /* Compute conditional distribution of each feature        */
2      foreach feature f_i in feature vector f do
3          P(f_i | C) = P(f_i,C)/P(C) = (N(f_i,C)+α)/(N(C)+αd)
4      foreach link L do
5          all ← N(p2p)+N(p2c)+N(c2p)
6          P(L = p2p) ← P(p2p) = N(p2p)/N(all)
7          P(L = p2c) ← P(p2c) = N(p2c)/N(all)
8          P(L = c2p) ← P(c2p) = N(c2p)/N(all)
9          foreach feature f_i in feature vector f do
10             P(L = p2p) *= P(f_i | p2p)
11             P(L = p2c) *= P(f_i | p2c)
12             P(L = c2p) *= P(f_i | c2p)
13         sum = P(L = p2p)+P(L = p2c)+P(L = c2p)
14         P(L = p2p) ← P(L = p2p)/sum
15         P(L = p2c) ← P(L = p2c)/sum
16         P(L = c2p) ← P(L = c2p)/sum
           /* Update link's type                                  */
17         R = arg max_C P(L = C)
```

First, the algorithm calculates probabilities for each feature, conditional on the link type C (C in {p2p, p2c, c2p}) by accumulating probability values (line 2-3 in Algorithm 1). The parameter α is a smoothing parameter, which prevents a feature with examples in only one class from forcing the probability estimate to be 0 or 1. In our implementation, we use *Laplace (Add-1) Smoothing* [42], which sets the smoothing parameter to 1. The algorithm then assigns probability that link L is of each type by the prior probability distribution $P(C)$, which is the proportion of each link type in the data (line 5-8 in Algorithm 1). Then, it goes through each feature and multiplies the probability that link L is of each type by the conditional probability of the feature given each link type (line 9-12). In the end, the final probability of the link L of being each type is calculated by the fraction of each type's probability over the sum of probabilities of all possible link types (lines 14-16 in Algorithm 1). We then update L's type by picking the link type with the largest probability (line 17). We repeat this process of link type inference and updating probability distributions of features until convergence, i.e., the percentage of links that change labels between each iteration drops below a small threshold. The algorithm usually converges within four iterations.

Algorithm Design Choice: We considered alternative approaches for prediction, such as supervised learning based on a training set of labeled link types from community attributes (say using boosted trees). However, since the size of the ground truth from community attributes has not increased in recent years and since our analysis in §4.3 shows that the validation dataset is skewed and is only partially representative of the overall Internet, a supervised learning approach would be affected by the biases in the training set.

We instead adopted an unsupervised approach that uses EM. In particular, `ProbLink` falls in the category of techniques that use the *expectation maximization algorithm for parameter estimation in Naïve Bayes classifers* [46]. In particular, Expectation Maximization (EM) is the iterative technique used to separate out classes from a mixture, and Naïve Bayes is the classification technique used in each iteration. EM is suitable when there are hidden classes, and the observed feature distributions are a mixture of the feature distributions of different classes. In each iteration of EM, the algorithm groups together elements that are classified together and derives the feature parameters of each class. Recall that in our setting, the feature parameters are estimates of the probability of different types of links given a particular topological feature, namely, the probability of the middle link type given previous/next link type, the probability of a certain number of VPs observing a link given each link type, and so on.

Our approach and the underlying techniques have the following implications. First, there is no ground truth in any stage of the algorithm. Second, EM does not work when the classification technique is a black-box or a non-parametric technique (such as a neural network or a boosted decision tree) since it is hard for EM to converge to a stable set of black-box parameters. In fact, in the case of decision trees, convergence would mean that not only the values in the tree nodes do not change, but also the structure of the trees remains stable across iterations. This convergence requirement is hard to satisfy, and hence non-parametric techniques such as decision trees are ill-suited for EM in spite of their high classification accuracy. Naïve Bayes, on the other hand, is a parametric technique that has been shown to work well even when there is correlation between the features used for prediction. Crucially, when Naïve Bayes is used as the classification technique, EM can converge and attribute a stable set of parameters to be used by Naïve Bayes.

It is worth noting that Naïve Bayes makes the assumption that all features are conditionally independent. This independence assumption rarely holds in practical situations, including our own. Nevertheless, despite violating the independence assumption, the classification decisions made by Naïve Bayes are often of high quality [62, 53, 63, 58]. Moreover, in our context, what is needed is a parametric classification technique as opposed to a prediction technique that attributes precise probabilities for the different classes. The

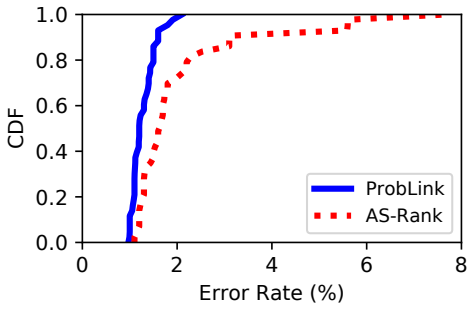


Figure 4: CDF of error rates of ProbLink and AS-Rank on the snapshots of BGP paths in the past 6 years.

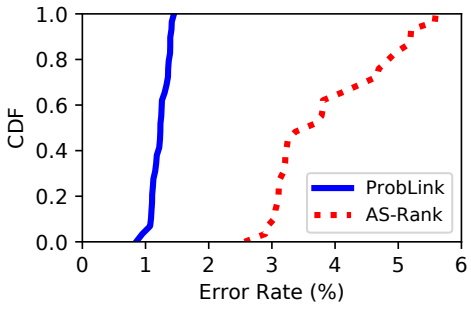


Figure 5: CDF of error rates of ProbLink and AS-Rank on 30 consecutive 1-day snapshots from April 1, 2016 to April 10, 2016.

Naïve Bayes classifier is appropriate for such settings, but the lack of conditional independence does have the downside that we cannot use the derived probability values as a confidence measure [7].

7 Evaluation

We now evaluate our probabilistic inference algorithm, ProbLink, from three aspects and show that:

- ProbLink consistently achieves low error rates across many years, reducing the average error rate of AS-Rank for all links by 1.7×, and attaining 1.8-6.1× better error rates for the different categories of *hard* links.
- ProbLink is not dependent on its bootstrapping algorithm, and it is stable with respect to snapshot and VP selection.
- Each feature used in ProbLink is meaningful and eliminating any of them harms the overall inference accuracy.

7.1 Accuracy

To evaluate the accuracy of ProbLink, we assemble daily snapshots of BGP paths on the first five days of April, August, and December (i.e., every four months) over the past 6 years. We apply our algorithm against these snapshots and compare it with the AS-Rank algorithm over this time period.

Figure 4 compares the error rates of inferences made by ProbLink and those made by AS-Rank. Our probabilistic inference algorithm consistently yields a low error rate smaller than 2%, reducing the average error rate of AS-Rank for all links from 2.1% to 1.2%.

Category	AS-Rank(%)	ProbLink(%)
Observed by 50-100 VPs	8.8	1.5
Non-VP & non-Tier1	4.4	1.7
Unlabeled Stub-clique	33.6	5.5
Conflict	6.8	3.8

Table 4: Average error rates on *hard* links. ProbLink achieves 5.9×, 2.6×, 6.1×, and 1.8× better error rate for the links observed by between 50 and 100 VPs, non-VP & non-Tier1 links, unlabeled stub-clique, and conflict than AS-Rank respectively.

Feature excluded	Error rate
None	1.5%
Triplet	2.4%
Non-path	1.7%
Distance to Clique	1.7%
VP	4.3%
Co-located IXP and peering facility	1.8%

Table 5: Error rates of ProbLink with all features turning on and without each feature in turn against 04/01/2017 BGP paths.

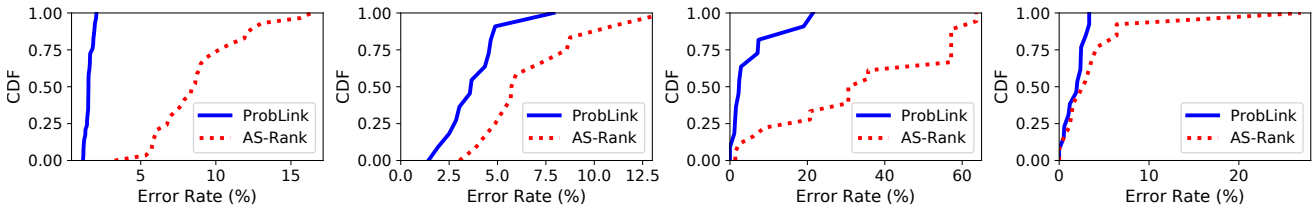
Figure 5 shows a comparison between error rates of ProbLink and AS-Rank on 30 consecutive snapshots of BGP paths during April 2016. The max and average error rates across these days for ProbLink are 1.4% and 1.2%, while the error rate of AS-Rank ranges from 2.6% to 5.6%, with an average error rate of 3.9%. ProbLink is not sensitive to the specific set of paths used in a snapshot, and it achieves uniformly low error rates in spite of clique inference inaccuracies that adversely impact AS-Rank.

Figure 6 plots the CDFs of error rates of inferences made by ProbLink and AS-Rank on the four categories of *hard* links identified in §4.3. Not only does our algorithm yield much smaller error rates, but it also has less variation than AS-Rank. Table 4 lists the average error rate of our algorithm and AS-Rank for links *observed by 50 to 100 VPs*, *non-VP and non-Tier1* links, *stub-clique* links, and *conflict* links. Our probabilistic algorithm reduces the error rate on the four categories by a factor of 5.9, 2.6, 6.1, and 1.8 respectively compared to AS-Rank.

ProbLink is not dependent on the initial labels provided by the bootstrapping algorithm. Bootstrapping with CoreToLeaf and AS-Rank only results in 0.15% overall accuracy difference on average. Completely random initial assignment would not work well because our algorithm attempts to separate mixture distributions with some underlying properties. Our algorithm should, however, be robust to various types of initial label assignments as long as there is some weak correlation between the initial labeling and the actual assignments.

7.2 Feature Importance Analysis

Eliminating any of the features used by ProbLink results in lower accuracy. We next run ProbLink with each feature excluded in order to show that each feature adds value. Ta-



(a) *Observed by between 50 and 100 VPs* links. **(b)** *Conflict* links. **(c)** *Unlabeled stub-clique* links. **(d)** *Non-VP and Non-Tier1* links.

Figure 6: CDFs of error rates of `ProbLink` vs. `CoreToLeaf` on *hard* links in a period of 30 days in April 2016.

ble 5 lists the error rates of `ProbLink` after turning off each feature one-by-one against the 04/01/2017 snapshot of BGP paths.

Excluding any feature in `ProbLink` results in a higher error rate, which suggests that each feature adds some value. Among all the features, excluding the VP feature harms the overall accuracy the most, indicating that the visibility of BGP paths from many vantage points is a crucial attribute for inferring link types. We believe that integrating more features can further improve `ProbLink`'s accuracy.

8 Practical Applications

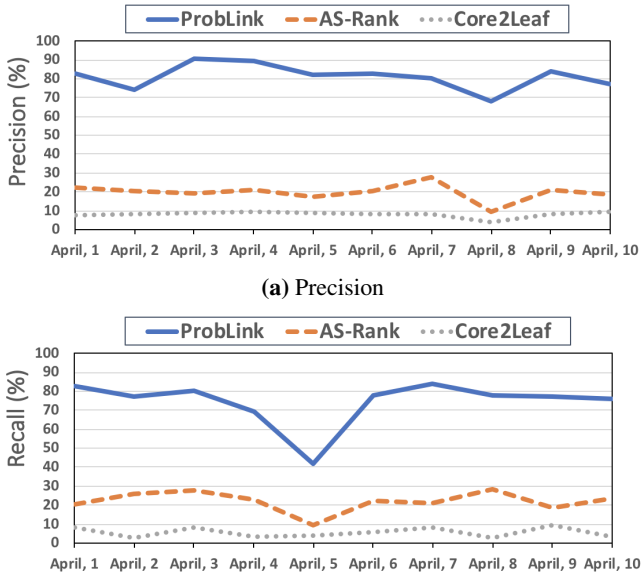
8.1 Route Leak Detection

Route leaks are a class of common routing incidents that can cause large Internet service disruptions [54]. They are caused by violations of the policies among the ASes involved. For instance, on November 5, 2012, a Google peer Moratel (*AS*23947) improperly advertised Google routes to its provider, causing Moratel's providers to select the leaked routes as the preferred ones destined to Google. As Moratel could not handle such large traffic volumes, Google's services went offline in parts of Asia for half an hour [28].

A conventional method for detecting route leaks is through checking valley-free violations in BGP paths. Mauch built a routing leak detection system based on this intuition by searching for valley paths containing three or more major networks with known relationships [32].

In the same spirit, we build a route leak detection system by detecting valley-free violations in paths based on the link relationship inference results of `ProbLink`. It is worth noting that a large fraction (more than 50%) of the valley-free violations do not result from route leaks but intended policies from ASes that are research/educational or IXPs [26]. Such ASes often establish a special type of AS relationship called indirect peering, where an AS functions as an intermediate link between two other ASes who wish to peer but go through intermediate ASes. Therefore, we ignore a path if it contains research/educational or IXP ASes when detecting route leaks.

To evaluate the performance of `ProbLink` and other AS relationship inference techniques on route leak detection, we use only those links for which we have validation data in BGP paths. For example, suppose a path has link relationships "* * p2p * c2p", where * is unknown in the validation dataset. Even though a relationship inference algorithm should have predictions on the unknown links, we just detect route leaks by its predictions on the two known links in order to compare against the validation dataset. Figure 7 compares the precision and recall of `ProbLink`, `AS-Rank`, and `CoreToLeaf`, against the real route leaks implied by the validation dataset across 10 days in April 2016. The average precision for `ProbLink`, `AS-Rank`, and `CoreToLeaf` is 81.1%, 19.8%, and 8.1%, respectively; and the average recall for `ProbLink`, `AS-Rank`, and `CoreToLeaf` is 76.2%, 22.1%, and 5.6%, respectively.

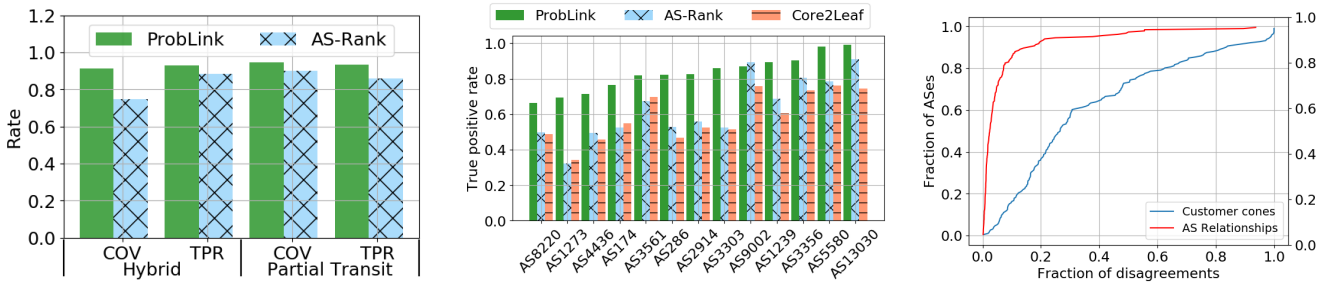


(a) Precision

(b) Recall

Figure 7: Evaluation of route leak detection across 10 days.

While `ProbLink` significantly improves the state-of-the-art in route leak identification, the number of false-positive inferences is still relatively high (almost 20%) when used as a stand-alone inference heuristic. That said, `ProbLink` shows that valley-free violations are a strong signal for the occurrence of route leaks. This indicator facilitates post processing and can be used as part of a composite detection mechanism that combines multiple sources of information, such as the detection of abrupt changes in per-prefix traffic levels [5].



(a) Comparison of coverage (COV) and True Positive Rate (TPR) between `ProbLink` and `AS-Rank` in complex relationship inference.

(b) True positive rate per relationship inference algorithm for the prediction of the impact of selective advertisement policies.

(c) Disagreements in AS relationships and ASes in the customer cone of the 200 ASes with the largest customer cones.

Figure 8: Evaluation of complex relationships inference and the prediction of path changes due to selective prefix advertisements.

8.2 Inference of Complex Relationships

AS relationships may be more complex than the traditional p2c/p2p model. Such complex agreements may take the form of a *hybrid* relationship with different relationship type for different Points-of-Presence (PoPs) or a *partial-transit* relationship, in which a provider offers transit only toward its peers and customers, but not its providers, or restricts transit to a specific geographic region [15, 19]. The state-of-the-art algorithm for inferring complex relationships (`CR` algorithm) takes as input a set of conventional relationships and iteratively refines them by combining active traceroute measurements with geolocation data to discover the PoP-level propagation patterns of inter-domain paths [25]. Due to the high measurement cost in terms of traceroute queries required to infer complex relationships, `CR` utilizes customer cones to optimize the allocation of queries to traceroute probes and maximize the discovery of hybrid relationships within the limited querying budgets used by platforms such as RIPE Atlas [52]. Therefore, the quality of the p2p/p2c relationships can affect the precision, accuracy, and coverage of complex relationship inference.

To test the performance of `ProbLink` for complex relationship inference, we implemented the `CR` algorithm and executed two 2-day measurement campaigns over the RIPE Atlas platform, on 2018/09/06 and 2018/09/08 using `ProbLink` and `AS-Rank` respectively. For each measurement round, we allocated the maximum permissible number of measurement credits, which resulted in 125,529 traceroute queries from 7,870 Atlas probes. `CR+ProbLink` inferred 1,308 hybrid relationships and 3,163 partial transit links, while `CR+AS-Rank` inferred 1,029 hybrid relationships and 3,009 partial transit links. We evaluated these inferences against our validation dataset, which includes 346 hybrid links and 402 partial-transit links. As shown in Figure 8a, combining `CR` with `ProbLink` not only improves the True Positive Rate (TPR) of the algorithm both for hybrid and partial transit relationships, but, importantly, we significantly expand its coverage (COV) by capturing 91% of the hybrid and 95% of the partial transit relationships, compared to 76% and 90%, respectively, for `CR+AS-Rank`. Overall, `CR+ProbLink` discovers 27% more hybrid relationships than `CR+AS-Rank`.

8.3 Predicting the Impact of Selective Advertisements

The ability to predict the impact of traffic engineering policies on the active BGP paths can be valuable to network operators, as it would limit the need for trial-and-error experimentation, allow the configuration of more predictable and stable routing policies, and minimize the risk of propagating unintended routes [45]. However, past works have shown that the existing AS relationship datasets have poor predictive capabilities, making them impractical for such purposes. In this section, we evaluate the impact of `ProbLink`'s improved relationship inferences in predicting the outcome of a selective advertisement. Selective advertisement is a popular traffic engineering technique used by AS operators to achieve traffic load balancing, by advertising certain routes only to a subset of their inter-domain neighbors [48].

To predict the impact of selective advertisement "in the wild", we first need to explicitly capture the activation and the scope of such policies. We detect selectively advertised prefixes by utilizing route redistribution BGP Communities, which are increasingly utilized to implement selective prefix advertisement [49, 17]. In particular, many providers define an array of Community values that can be set by their customers, to allow them to control whether the provider should propagate or not a route to a specific peer or group of peers. For instance, if *AS*9002 (RETN) receives a prefix advertisement from a customer annotated with the BGP Community `9002:65535`, then RETN will propagate this route only to its customers, but not its peers or providers [50]. Redistribution Communities can further limit the scope of the prefix advertisement by determining a location for which the redistribution policy will be applied. For instance, when the Community `286:49` is applied on a prefix, *AS*286 (KPN) will not advertise this prefix to its US peers [37]. By parsing WHOIS records and NOC websites, we compile a dictionary of Community values that define one of the following types of selective route redistribution:

- Do not announce route to neighbors of type R.
- Do not announce route to neighbors of type R at L.

- Announce route only to neighbors of type R.
- Announce route only to neighbors of type R at L.

R indicates relationship type (customer, provider, peer) and L indicates a city-level or country-level location identifier. In total, we extracted 644 Community values from 152 ASes.

After compiling our Communities dictionary, we monitor the BGP messages of the corresponding ASes to capture BGP Updates annotated with one of the redistribution Communities. Let us assume we observe a BGP Update for a destination prefix d annotated with a BGP Community C, which instructs AS_C to propagate p *only* to its neighbors of type R. We calculate which ASes will have to change their paths as follows: We first parse the BGP paths right before C was applied on the prefix d, and we collect all the paths P_{ALL} to d that traversed AS_C. Then, based on the inferred relationships, we find the paths $P_{R'} \subseteq P_{ALL}$ in which AS_C advertises the route toward d to a neighbor with relationship type $R' \neq R$. Since the Community C allows the prefix announcement only to neighbors with relationship type R, we infer that the paths $P_{R'}$ will be withdrawn, and the corresponding ASes in these paths will choose a different path. When C also defines a geographic scope for the prefix advertisement in addition to the relationship type, we use the techniques described in [25] to map the city-level location of AS interconnections, and calculate the affected paths in a similar manner. We validate our inferences by observing the withdrawn paths after C was applied on the path. We consider as false positive any AS $a \in P_{R'}$ that did not withdraw its path 15 minutes after we observed the BGP Update with C. We do not consider false negatives, as an AS may change its path to d for different reasons, and this change may simply coincide with the application of the Community C on the same prefix.

Figure 8b shows our validation results after executing the above experiment for the first week of April 2016. During that period, we found 480 prefixes tagged with redistribution communities defined by 13 ASes. Overall, 83% of `ProbLink`'s predictions were correct, compared to 62% for `AS-Rank` and 59% for `CoreToLeaf`. `ProbLink` outperformed `AS-Rank` for every AS except $AS9002$, and in some cases (e.g. $AS1273$), the true positive rate was 2x higher compared to the other algorithms. These results are surprising given that less than 4% of the relationship inferences differ between `ProbLink` and `AS-Rank`. To understand the significant improvement achieved by `ProbLink`, we investigate the impact of the relationship disagreements between the two algorithms on the customer cones obtained using the *Provider/Peer Observed* methodology proposed in [41]. We focus on the ASes with at least 100 ASes in their downstream path. For each of these ASes we calculate the fraction of their relationships and the fraction of their customer cones that disagree between `ProbLink` and `AS-Rank`. As shown in Figure 8c, while less than 10% of the ASes had more than 20% relationship mismatches, over 60% of the ASes had at least 20% difference in their customer cones. This finding highlights the fact that even a few incorrect relationship inferences can lead to significant differences in properties of the resulting downstream paths and substantial deviations in the predictive capabilities of `ProbLink` and `AS-Rank`.

9 Conclusion

We revisit the AS relationship problem and inference techniques. We first develop a simple inference algorithm that achieves accuracy comparable to that of the state-of-the-art inference technique, `AS-Rank`, indicating that the types of most links in validation datasets are relatively easy to infer. We then construct different subsets of the validation dataset that might be considered *hard* and use these as benchmarks for evaluating improvements in AS relationship inference. Further, we observe that many of the features that can be used by inference techniques are of a stochastic nature, so we present a probabilistic AS relationship inference algorithm that provides a framework for easy integration of many noisy but useful attributes into the relationship inference algorithm. We show that this probabilistic algorithm is more accurate and less sensitive to the locations of vantage points and BGP paths compared to the state-of-the-art algorithms.

Acknowledgments

We would like to thank the anonymous NSDI reviewers and our shepherd Andreas Haeberlen for their valuable feedback. This research was partially supported by the National Science Foundation under Grants CNS-1614717 and CNS-1513847.

References

[1] IXP Service Matrix. https://www.euro-ix.net/en/tools/ixp-service-matrix/.

[2] PeeringDB. https://www.peeringdb.com/.

[3] RIPE (RIS). http://www.ripe.net/ris/.

[4] U. Oregon Route Views Project. http://www.routeviews.org/.

[5] B. Al-Musawi, P. Branch, and G. Armitage. BGP anomaly detection techniques: A survey. *IEEE Communications Surveys Tutorials*, 19(1):377–396, Firstquarter 2017.

[6] R. Anwar, H. Niaz, D. Choffnes, Í. Cunha, P. Gill, and E. Katz-Bassett. Investigating interdomain routing policies in the wild. In *Proceedings of the 2015 Internet Measurement Conference*, pages 71–77. ACM, 2015.

[7] P. Bennett. Assessing the calibration of Naive Bayes' posterior estimates. Technical report, September 2000. Computer Science Department, School of Computer Science, Carnegie Mellon University.

[8] R. Bush, O. Maennel, M. Roughan, and S. Uhlig. Internet optometry: assessing the broken glasses in internet reachability. IMC '09.

[9] M. Caesar and J. Rexford. BGP routing policies in ISP networks. *Netwrk. Mag. of Global Internetwkg.*, 19(6):5–11, Nov. 2005.

[10] CAIDA. Inferred AS to organization mapping dataset. http://www.caida.org/data/as-organizations/.

[11] A. Cohen, Y. Gilad, A. Herzberg, and M. Schapira. Jumpstarting BGP security with path-end validation. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 342–355. ACM, 2016.

[12] G. Comarela, E. Terzi, and M. Crovella. Detecting unusually-routed ASes: methods and applications. In *Proceedings of the 2016 Internet Measurement Conference*, IMC '16, pages 445–459, New York, NY, USA, 2016. ACM.

[13] A. Dhamdhere, D. D. Clark, A. Gamero-Garrido, M. Luckie, R. K. P. Mok, G. Akiwate, K. Gogia, V. Bajpai, A. C. Snoeren, and K. Claffy. Inferring persistent interdomain congestion. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, pages 1–15, New York, NY, USA, 2018. ACM.

[14] G. Di Battista, M. Patrignani, and M. Pizzonia. Computing the types of the relationships between autonomous systems. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 1, pages 156–165. IEEE, 2003.

[15] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, G. Riley, et al. AS relationships: Inference and validation. SIGCOMM '07, 2007.

[16] X. Dimitropoulos, D. Krioukov, B. Huffaker, G. Riley, et al. Inferring AS relationships: Dead end or lively beginning? In *International Workshop on Experimental and Efficient Algorithms*, pages 113–125. Springer, 2005.

[17] B. Donnet and O. Bonaventure. On BGP communities. *ACM SIGCOMM Computer Communication Review*, 38(2):55–59, 2008.

[18] Q. Fan, H. Yin, G. Min, P. Yang, Y. Luo, Y. Lyu, H. Huang, and L. Jiao. Video delivery networks: Challenges, solutions and future directions. *Computers & Electrical Engineering*, 66:332–341, 2018.

[19] P. Faratin, D. D. Clark, S. Bauer, W. Lehr, P. W. Gilmore, and A. Berger. The growing complexity of internet interconnection. *Communications & Strategies*, (72):51, 2008.

[20] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[21] L. Gao. On inferring autonomous system relationships in the internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745, Dec 2001.

[22] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. The flattening internet topology: Natural evolution, unsightly barnacles or contrived collapse? In *International Conference on Passive and Active Network Measurement*, pages 1–10. Springer, 2008.

[23] P. Gill, M. Schapira, and S. Goldberg. Let the market drive deployment: A strategy for transitioning to BGP security. In *SIGCOMM '11*, volume 41, pages 14–25. ACM, 2011.

[24] V. Giotsas, C. Dietzel, G. Smaragdakis, A. Feldmann, A. Berger, and E. Aben. Detecting peering infrastructure outages in the wild. In *SIGCOMM '17*, pages 446–459, 2017.

[25] V. Giotsas, M. Luckie, B. Huffaker, and k. claffy. Inferring complex AS relationships. In *Internet Measurement Conference (IMC)*, pages 23–30, Nov 2014.

[26] V. Giotsas and S. Zhou. Valley-free violation in internet routinganalysis based on BGP community data. In *Communications (ICC), 2012 IEEE International Conference on*, pages 1193–1197. IEEE, 2012.

[27] V. Giotsas, S. Zhou, M. Luckie, and k. claffy. Inferring multilateral peering. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '13, pages 247–258, New York, NY, USA, 2013. ACM.

[28] S. Goldberg. Why is it taking so long to secure Internet routing? *Communications of the ACM*, 57(10):56–63, 2014.

[29] Y. He, G. Siganos, M. Faloutsos, and S. Krishnamurthy. Lord of the links: a framework for discovering missing links in the Internet topology. *IEEE/ACM Transactions on Networking (ToN)*, 17(2):391–404, 2009.

[30] G. Huston. The death of transit and the future internet. Second ITU Workshop on Network 2030, December 2018.

[31] G. Inc. Google peering policy. https://peering.google.com.

[32] M. Jared. BGP routing leak detection system. https://puck.nether.net/bgp/leakinfo.cgi.

[33] E. Jasinska, N. Hilliard, R. Raszuk, and B. N. RFC 7947: Internet exchange BGP route server. https://tools.ietf.org/html/rfc7947.

[34] J. Juen, A. Johnson, A. Das, N. Borisov, and M. Caesar. Defending tor from network adversaries: A case study of network path prediction. *Proceedings on Privacy Enhancing Technologies*, 2015(2):171–187, 2015.

[35] E. Katz-Bassett, D. R. Choffnes, Í. Cunha, C. Scott, T. Anderson, and A. Krishnamurthy. Machiavellian routing: improving Internet availability with BGP poisoning. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, page 11. ACM, 2011.

[36] M. Konte, R. Perdisci, and N. Feamster. Aswatch: An AS reputation system to expose bulletproof hosting ases. In *ACM SIGCOMM Computer Communication Review*, volume 45, pages 625–638. ACM, 2015.

[37] KPN NOC. BGP Communities For AS286. https://as286.net/AS286-communities.html, September 2018.

[38] S. Kuenzer, A. Ivanov, F. Manco, J. Mendes, Y. Volchkov, F. Schmidt, K. Yasukata, M. Honda, and F. Huici. Unikernels everywhere: The case for elastic CDNs. *SIGPLAN Not.*, 52(7):15–29, Apr. 2017.

[39] P. Laskowski and J. Chuang. Network monitors and contracting systems: Competition and innovation. SIGCOMM '06, pages 183–194, 2006.

[40] A. H. Lodhi. *The economics of Internet peering interconnections*. PhD thesis, Georgia Institute of Technology, 2014.

[41] M. Luckie, B. Huffaker, A. Dhamdhere, V. Giotsas, et al. AS relationships, customer cones, and validation. IMC '13.

[42] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[43] R. Mazloum, M.-O. Buob, J. Auge, B. Baynat, D. Rossi, and T. Friedman. Violation of interdomain routing assumptions. In *International Conference on Passive and Active Network Measurement*, pages 173–182. Springer, 2014.

[44] A. Mitseva, A. Panchenko, and T. Engel. The state of affairs in BGP security: A survey of attacks and defenses. *Computer Communications*, 124:45 – 60, 2018.

[45] W. Mühlbauer, A. Feldmann, O. Maennel, M. Roughan, and S. Uhlig. Building an AS-topology model that captures route diversity. SIGCOMM '06, pages 195–206, 2006.

[46] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine learning*, 39(2-3):103–134, 2000.

[47] R. Nithyanand, O. Starov, P. Gill, A. Zair, and M. Schapira. Measuring and mitigating AS-level adversaries against Tor. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*, 2016.

[48] B. Quoitin, C. Pelsser, L. Swinnen, O. Bonaventure, and S. Uhlig. Interdomain traffic engineering with BGP. *IEEE Communications Magazine*, 41(5):122–128, May 2003.

[49] B. Quoitin, S. Tandel, S. Uhlig, and O. Bonaventure. Interdomain traffic engineering with redistribution communities. *Computer Communications*, 27(4):355–363, 2004.

[50] RETN NOC. BGP communities For AS9002. http://retn.net/support/bgp-communities/, September 2018.

[51] RFC. RFC 6996: Autonomous System (AS) reservation for private use. https://tools.ietf.org/html/rfc6996.

[52] RIPE Atlas. User-defined measurements - rate limits. https://atlas.ripe.net/docs/udm/#rate-limits.

[53] I. Rish et al. An empirical study of the Naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York, 2001.

[54] K. Sriram, D. Montgomery, D. McPherson, E. Osterweil, and B. Dickson. Problem definition and classification of BGP route leaks. https://tools.ietf.org/html/rfc7908.

[55] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the Internet hierarchy from multiple vantage points. IEEE, 2002.

[56] A. Technologies. Akamai Network partnerships. https://www.akamai.com/us/en/products/network-operator/akamai-network-partnerships.jsp.

[57] J. S. Varghese and L. Ruan. Computing customer cones of peering networks. In *Proceedings of the 2016 Applied Networking Research Workshop*, pages 35–37. ACM, 2016.

[58] Q. Wang, G. M. Garrity, J. M. Tiedje, and J. R. Cole. Naive Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy. *Applied and environmental microbiology*, 73(16):5261–5267, 2007.

[59] Wikipedia. List of Tier-2 ASes. https://en.wikipedia.org/wiki/Tier_2_network.

[60] W. Willinger and M. Roughan. Internet topology research redux. *ACM SIGCOMM eBook: Recent Advances in Networking*, 2013.

[61] J. Xia and L. Gao. On the evaluation of AS relationship inferences. In *Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE*, volume 3, pages 1373–1377. IEEE, 2004.

[62] H. Zhang. The optimality of Naive Bayes. In V. Barr and Z. Markov, editors, *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2004)*. AAAI Press, 2004.

[63] H. Zhang and J. Su. Naive Bayesian classifiers for ranking. In *European conference on machine learning*, pages 501–512. Springer, 2004.

A AS-Rank Algorithm

The `AS-Rank` algorithm takes the following 11 steps to infer the relationship of each link.

1. Discard or sanitize paths with artifacts.
2. Sort ASes in decreasing order of computed transit degree, then node degree.
3. Infer a transit-free clique (i.e., Tier-1) ASes at top of AS hierarchy and label the links between every pair of ASes in the clique as p2p links.
4. Discard poisoned paths.
5. Visit ASes in order of the ranking in (2), and label a link as c2p if its previous link in a BGP path is composed of two clique members, or if its previous link in a BGP path is already labeled as c2p.
6. Infer c2p relationships from VPs inferred to be announcing no provider routes.
7. Infer c2p relationships for ASes where the customer has a larger transit degree.
8. Infer customers for ASes with no providers.
9. Infer c2p relationships between stub ASes and clique ASes.
10. Infer c2p relationships where adjacent links have no relationship inferred.
11. Infer all other links left as p2p.

Feature label	Meaning
f1	Number of VPs which observe a link
f2	Max distance to Tier-1
f3	Min distance to Tier-1
f4	Max node degree
f5	Min node degree
f6	Node degree difference
f7	Max transit degree
f8	Min transit degree
f9	Transit degree difference

Table 6: Features fed into GBDT

B AS-Rank Clique Inference

The clique inference algorithm in `AS-Rank` works as the following:

1. Find the top 10 ASes by transit degree.
2. If there are three consecutive members (X-Y-Z) in the top 10 ASes showing up in paths, and there are more than 5 ASes downstream from X Y Z (to make sure that the paths containing three consecutive members are not poisoned), disconnect the edge between X and Z even though X and Z are connected in some paths.
3. Find the largest clique in terms of transit degree sum among the top 10 ASes, denoted as *C*.
4. Visit the rest ASes top to down by transit degree, add an AS Z to *C* if Z has links with all members in *C*.
5. Similar to Step 2: If there are three consecutive members (X-Y-Z) in C showing up in paths, and there are more than 5 ASes downstream from X Y Z, disconnect the edge between X and Z.
6. Find the largest clique in *C* in terms of transit degree sum as the final inferred clique.

C Feature Importance Computation

Gradient boosting [20] is a widely used machine learning technique for solving classification problems. In gradient boosting, *GBDT* (Gradient boosting decision trees) produces a prediction model in the form of an ensemble of multiple decision trees. It is straightforward to retrieve importance scores for features when constructing *GBDT*. An importance score (F score) describes the number of times a feature is used to split the data across all trees. The more a feature is used to make key decisions with decision trees, the higher its importance score.

To decide what features can distinguish *hard* links from *easy* links in the Internet, we first split the validation dataset into two halves. The set of links which `CoreToLeaf` or `AS-Rank` infers incorrectly are labeled as "hard", while those which are inferred correctly are labeled with "easy". Then, we feed the features listed in Table 6 of links along with their labels into the *GBDT* and calculate the importance score corresponding to each feature.

Figure 9 plots the importance score of each feature divided by the sum of all features' scores. We can tell the features

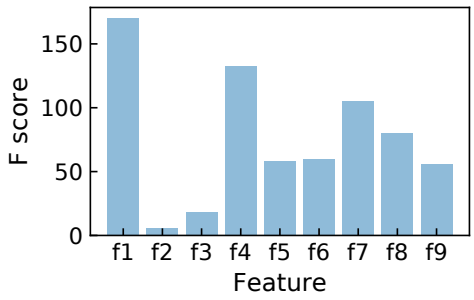


Figure 9: Feature importance scores provided by gradient boosting tree.

f1, f4, f7, f8, f9 are the most important ones, so we translate them into the various categories of features to characterize "hard" links in §4.3.

D AS-Rank Sensitivity Analysis

D.1 Sensitivity to VP Selection

Each vantage point provides its own view of the Internet AS-level topology and the flow of traffic from the VP to rest of the Internet. VPs are located in different places, belong to different tiers, and they themselves have different import and export policies.

Even though the number of VPs have been growing over time, VPs are free to join or leave the set of public collectors, so the selection of VPs we have access to is arbitrary, biased, and under flux. A good AS relationship inference algorithm should not be sensitive to the selection of these VPs.

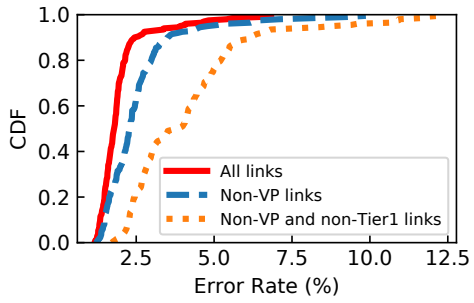


Figure 10: CDF of AS-Rank's error rates on paths seen from 200 different half-VP sets.

We run the AS-Rank algorithm repeatedly, 200 times, against the 04/01/2017 BGP snapshot. For each of these 200 executions, we choose a random VP subset consisting of half of all available VPs (which we denote as V below) and give as input to the AS-Rank algorithm only the BGP paths visible to the VPs in that subset. Figure 10 plots CDFs of AS-Rank's error rates using paths from these VP subsets. In the plot, we examine all links in the ground-truth dataset, links in the ground-truth dataset except links that directly connect with V (i.e., non-VP links), and the links in the ground-truth dataset except V's links as well as Tier-1 links (i.e., non-VP and non-Tier1 links). The inference error rates on overall links range from 1.2% to 6.9%, and the error rates on *non-VP and non-Tier1* links range from 1.8% to 12.3%. AS-Rank's accuracy is thus quite sensitive to the VP selections, especially for links which are relatively difficult to infer (i.e., not VP or Tier1 links).

D.2 Root Cause Analysis of AS-Rank Sensitivity

To illustrate the issue described in §5.3, let's consider two VP sets, V1 and V2, drawn from our 200 executions. AS-Rank's inference accuracy from V1 is low, while its inference accuracy from V2 is high. The largest ten ASes differ for different sets of VPs because the transit degrees of ASes are determined by paths observed by the VPs. For example, the 9th largest AS (AS2914) observed by V2 is the 12th largest AS observed by V1, so AS2914, which is a real Tier-1 AS, shows up in the clique chosen from the top 10 ASes using V2's paths, but it does not show up in the clique chosen using V1's paths.

For V1, the AS-Rank algorithm determines the maximum clique with the largest transit degree (from the top 10 ASes) to be "AS3356, AS6939, AS8220, AS9002, AS43531". AS43531 is not considered for V2 due to a relatively lower measurement of its transit degree, and it is not a high-tier AS in reality. This affects the subsequent expansion of the clique, wherein ASes are considered in order by degree and added to the clique if they have connections to all members of the clique. So, in V1's execution, all added members are required to have a direct link with AS43531, and AS1764, AS8767, AS12389, AS12552, AS20485, AS25091, AS33891, AS43531, AS57724 are therefore all added to the clique, even though they are all low-tier ASes.

In a nutshell, the clique inference of AS-Rank algorithm is sensitive to the top 10 largest ASes ranked by transit degrees, which are determined by the selection of VPs and the selection of snapshots. Further, the clique membership determines the order in which links are analyzed by AS-Rank, impacts the computation of *customer cones* for each clique member (i.e., the set of ASes that a clique AS can reach using p2c links), and impacts the overall accuracy of the algorithm.