

2017

On the dynamics of interdomain routing in the Internet

<https://hdl.handle.net/2144/27480>

"Downloaded from OpenBU. Boston University's institutional repository."

BOSTON UNIVERSITY
GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation

**ON THE DYNAMICS OF INTERDOMAIN ROUTING IN
THE INTERNET**

by

GIOVANNI VENTORIM COMARELA

B.S., Universidade Federal do Espírito Santo, 2010

M.S., Universidade Federal de Minas Gerais, 2012

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2017

© 2017 by
GIOVANNI VENTORIM CO-
MARELA
All rights reserved

Approved by

First Reader

Mark Crovella, PhD
Professor of Computer Science

Second Reader

Evimaria Terzi, PhD
Associate Professor of Computer Science

Third Reader

John Byers, PhD
Professor of Computer Science

From one thing, know ten thousand things.
Miyamoto Musashi

Acknowledgments

I would like to thank my advisor Mark Crovella. I am extremely fortunate for having the opportunity of working with him. Thank you, Professor, for the countless meetings, all the help, financial support, and mainly, for your patience. The many lessons I learned from you, technical or not, I will take for life. Special thanks to Evimaria Terzi for the help with a significant part of this dissertation; also to my dissertation's committee, for all the words of advice and incentive.

During these years I have made many friends in the lab. Thank you all for making my life much more fun. All the names would not fit on the page, but I have to mention the office mates, Bashir and Natali, for tolerating my odd personality; and of course, thank you, Larissa, for the candy, friendship, and for listening to my endless random complaints.

Finally, thank you to my family, especially my sister Bianca and my mother Sônia. From you, I have gotten nothing but positive words and support. Most importantly, my father Carlos; it has been many years since we were together, but the memories of you continue to guide me through life; I dedicate this work to you.

ON THE DYNAMICS OF INTERDOMAIN ROUTING IN THE INTERNET

GIOVANNI VENTORIM COMARELA

Boston University, Graduate School of Arts and Sciences, 2017

Major Professor: Mark Crovella, PhD
Professor of Computer Science

ABSTRACT

The routes used in the Internet’s interdomain routing system are a rich information source that could be exploited to answer a wide range of questions. However, analyzing routes is difficult, because the fundamental object of study is a set of paths. In this dissertation, we present new analysis tools – metrics and methods – for analyzing paths, and apply them to study interdomain routing in the Internet over long periods of time.

Our contributions are threefold. First, we build on an existing metric (Routing State Distance) to define a new metric that allows us to measure the similarity between two prefixes with respect to the state of the global routing system. Applying this metric over time yields a measure of how the set of paths to each prefix varies at a given timescale. Second, we present PathMiner, a system to extract large scale routing events from background noise and identify the AS (Autonomous System) or AS-link most likely responsible for the event. PathMiner is distinguished from previous work in its ability to identify and analyze large-scale events that may re-occur many times over long timescales. We show that it is scalable, being able to extract significant events from multiple years of routing data at a daily granularity.

Finally, we equip Routing State Distance with a new set of tools for identifying and characterizing unusually-routed ASes. At the micro level, we use our tools to identify clusters of ASes that have the most unusual routing at each time. We also show that analysis of individual ASes can expose business and engineering strategies of the organizations owning the ASes. These strategies are often related to content delivery or service replication. At the macro level, we show that the set of ASes with the most unusual routing defines discernible and interpretable phases of the Internet's evolution. Furthermore, we show that our tools can be used to provide a quantitative measure of the "flattening" of the Internet.

Contents

1	Introduction	1
1.1	Rate of change	2
1.2	High-impact events	4
1.3	Unusually-routed ASes	6
1.4	Roadmap	8
1.5	Related publications	8
2	Background: Interdomain Routing	10
2.1	Internet organization	10
2.2	The Border Gateway Protocol	11
2.3	AS relationships	12
3	Related Work	16
3.1	Analysis of the interdomain topology	16
3.2	Dynamics of interdomain routing	18
3.2.1	BGP instability measurement	18
3.2.2	Event detection and root cause analysis	19
3.2.3	Evolution and flattening of the Internet	20
3.3	Incompleteness of the known Internet topology	21
3.4	Routing State Distance	22
4	Studying Interdomain Routing over Long Timescales	24
4.1	Notation and Definitions	25
4.1.1	Multiple next-hop RSD	25

4.1.2	Temporal RSD	27
4.2	Dataset Description	28
4.3	Measurements	30
4.3.1	Analyzing TRSD	30
4.3.2	Analyzing Routing Changes	32
4.3.3	Contribution of Sources	36
4.4	Summary of the chapter	38
5	Identifying and Analyzing High-impact Routing Events	39
5.1	An Example	41
5.2	Notation and definitions	44
5.2.1	Notation	44
5.2.2	Problem definition	45
5.3	Dataset description	48
5.4	Extracting Events	53
5.4.1	2D Factorization	54
5.4.2	3D Factorization	56
5.5	Characterizing Events	60
5.6	Single Actor Analysis	64
5.6.1	Algorithm	64
5.6.2	Performance	67
5.6.3	Case studies	72
5.7	Summary of the chapter	78
6	Detecting Unusually-Routed ASes	80
6.1	Definitions	81
6.1.1	Path-based networks	81
6.1.2	RSD	83

6.2	Data Preparation	85
6.3	Unusually-Routed ASes	87
6.3.1	Problem definition	87
6.3.2	Algorithm	89
6.3.3	Results	90
6.4	Individual ASes	93
6.5	Evolution of Unusual ASes	99
6.5.1	Problem definition	101
6.5.2	Algorithm	101
6.5.3	Results	102
6.6	Global Path Characteristics	104
6.7	Data Considerations	108
6.7.1	Monitor locations	108
6.7.2	Peering links	111
6.8	Summary of the chapter	113
7	Conclusions	115
A	Algorithm for clustering 2D-events	118
B	Algorithm for selecting sources and destinations	119
C	Proof of Proposition 1	121
D	Algorithm for segmentation	124
E	Computing $q_{r,t}$	128
F	Additional data cleaning	130
G	Proof of Proposition 2	132
	References	133

List of Tables

4.1	TRSD computation for Figure 4-1.	28
4.2	Summary of the four datasets.	29
5.1	Summary of dataset.	49
5.2	Summary of the sampled tensors of routing changes.	52
5.3	Summary of (λ, ν) -events found by PathMiner.	60
5.4	Description of top-5 events of each year.	63
5.5	Single actor analysis of Top-5 events of each year.	68
5.6	Three different rankings for the Top-20 Single Actors	71
F.1	ASes that we removed from segmentation analysis	130

List of Figures

2·1	Example of possible AS interconnections and relationships. Solid arrows are for customer-provider (from customer to provider), and dotted lines for peering relationships.	13
4·1	Differences between routing decisions towards prefix p , owned by AS x_p , at: (a) time t_i ; and (b) time t_{i+n} . ASes b and e change their next-hop choices, while f is a new AS in the network.	27
4·2	Average TRSD time series ($r(i, n)$ versus i) for different values of n and datasets: (a) 2005, daily; (b) 2008, daily; (c) 2011, daily; and 2005 – 2012, monthly. Time series are remarkably stable, and a considerable amount of changes are not undone quickly.	31
4·3	Time series for sustained changes, $s(i, k)$, and churn, $c(i, k)$, for the 2011, daily dataset: (a) $s(i, k)$, $k = 7$ days; (b) $c(i, k)$, $k = 7$ days; and (c) weekly average of $s(i, 7)$ and $c(i, 7)$ – error bars are 95% confidence intervals. Sustained changes contribute more to TRSD, and most of the changes happen during the working days of the week.	34
4·4	Time series for sustained changes, $s(i, k)$, and churn, $c(i, k)$, for the 2005 – 2012, monthly dataset, and $k = 2$. Even on a monthly timescale, sustained changes contribute more to TRSD.	35

4.5	TRSD contribution per AS, 2011, daily dataset: (a) 01/01/11, for $n = 1, 2, 7, 30$; (b) Fraction of top contributors for $n = 1$; and (c) Number of days that each AS top contributor appears at Figure 4.5b, for $n = 1, 2, 7, 30$. Few ASes are responsible for most of TRSD.	37
5.1	Stages of PathMiner.	40
5.2	Summary of path changes in the network towards two prefixes (hosted at gray nodes) from Apr-30-2013 to May-01-2013.	42
5.3	Next-hop changes towards the two prefixes of Figure 5.2 during 2013. An observed next-hop change, by AS j , towards the subject prefix at day k implies value 1, black dot, for the element (j, k) . Subfigures are: (a) prefix 1, all changes; (b) Prefix 2, all changes; (c) Prefix 1, signal; and (d) Prefix 2, signal.	43
5.4	Distributions of the total number of next-hop changes: (a) per prefix; (b) per AS; and (c) per pair (AS, prefix), for 2011 only. In (c) value at (i, j) indicates the total number of next-hop changes from the first j ASes towards first i prefixes (ASes and prefixes are sorted, in decreasing order, by their number of changes).	51
5.5	Density <i>versus</i> Volume for 2D (λ, ν) -events obtained from slices of tensor \mathcal{C} , 2013 only. Prefix (horizontal) slices in (a), AS (lateral) slices in (b), and time (frontal) slices in (c). Experiments performed using Algorithm 1 with $\lambda = 0.7$, $\nu = 100$ and $\epsilon = 1\%$	57

5.6	Basic statistics about (λ, ν) -events found by PathMiner ($\lambda = 0.8$ and $\nu = 100$). Time series of: (a) total number of next-hop changes in each yearly sample; (b) percentage of these changes captured in events by PathMiner; (c) number of events found in each day. Complementary CDFs of: volume of events – volume is defined as the product of the number of prefixes, ASes and days in an event; (e) density of events – density is the number of changes captured by an event over its volume; (f) product volume \times density of events. Event scatterplots (2013 only): (g) number of prefixes <i>versus</i> volume; (h) number of ASes <i>versus</i> volume; (i) number of days <i>versus</i> volume.	61
5.7	Summary of our methodology for Single Actor identification: (a) CDF of ΔF , computed over each day of each event; (b) CDF of the fraction of days that the same AS (or AS-link) is reported as actor (has maximum ΔF score) over all days of the event; (c) scatterplot of the number of days within an event <i>versus</i> the fraction of days that the same AS (AS-link) is reported as actor – same variable as in (b). . . .	70
5.8	Representation of case study I: (a) path changes between Feb-08-2009 and Feb-09-2009; and (b) path changes between Feb-10-2009 and Feb-11-2009	74
5.9	Representation of case study II: (a) path changes between Jan-10-2012 and Jan-11-2012; and (b) Path changes between Jan-11-2012 and Jan-12-2012.	76
5.10	Representation of case study III: (a) Path changes between Jan-11-2005 and Jan-12-2005; and (b) Path changes between Jan-12-2005 and Jan-13-2005.	77

6.1	A path-based network over nodes $V = \{a, b, c, d, e, f\}$, broken out to separately show the paths towards each square node. A path is a sequence of directed edges with same line style. Examples of path sets are: $P(f, d) = \{fcbad\}$, $P(b, b) = \{b\}$, and $P(d, c) = \{dbac, dbc\}$. Those path sets entail respectively the following next-hop multisets: $N(f, d) = \{c\}$, $N(b, b) = \{b\}$, and $N(d, c) = \{b, b\}$	82
6.2	December 2014 dataset. (upper) Density of dendrogram split values across the dataset; (bottom) Dendrogram of the top-40 clusters of unusual ASes. AS descriptions are shortened from http://bgp.potaroo.net/cidr/autnums.html	91
6.3	Columns correspond to ASes operated by Google (AS15169), GoDaddy (AS26496), Ams-IX (AS1200) and K-Root Server (AS25152). The top row shows Average RSD (Δ_t) time series; the red triangles at the bottom indicate when the AS was in one of the top-40 groups of unusual ASes. The middle row shows Temporal RSD ($\tau_{t,t'}$) comparing all possible pairs t and t' over the 13 years of study. The bottom row shows next-hop distribution (η_t) of the top 10 next hops used to reach the corresponding AS.	95

6·4	Phase-based analysis of unusual ASes for the past 13 years. Each month t is characterized by its set of top-45 unusual AS clusters, denoted by X_t . (A) O_d , overlap distance between sets of all unusual ASes at different times. (B) Segmentation scores $\log(q_{r,t})$, showing the relative quality of t as a segment boundary. Red arrows show actual segment boundaries determined by Problem 3. (C) Mean Squared Error of the segmentation model, given the data for different r and k values. (D) J_d , Jaccard distance between sets of unusual ASes after using segment representatives to filter noise.	100
6·5	Segment representatives reflecting the evolution of unusually-routed ASes.	103
6·6	(A) Global RSD over the 13-year period of study. The line is fit using weighted least squares (using the variance of pairwise RSD at each time). (B) Global RSD for the six ITER configurations of our simulation study. Configurations 1 and 6 represent hierarchical and mesh-like structures respectively. Error bars are 95% confidence intervals. . . .	106
6·7	RSD contribution for different AS categories and parameter configurations. Error bars are 95% confidence intervals – Omitted for the leftmost figure to improve visibility.	110

List of Abbreviations

AS	Autonomous System
BGP	Border Gateway Protocol
CDN	Content Delivery Network
DNS	Domain Name System
IP	Internet Protocol
ISP	Internet Service Provider
IXP	Internet eXchange Point
MRSD	Multiple Next-hop Routing State Distance
PoP	Point of Presence
RFC	Request For Comments
RIB	Routing Information Base
RSD	Routing State Distance
TRSD	Temporal Routing State Distance

Chapter 1

Introduction

The Internet has become a major pillar of today's society. However, guaranteeing the proper functioning of the Internet, and guaranteeing that its evolution will happen with the same principles under which it was created, is challenging. In this context, the challenges are both technical and organizational. The former is related to the growing number of users, new applications with different requirements, and new types of devices. The latter refers to the fact that the Internet is not a single and isolated system, but a collection of networks belonging to many organizations, whose goals and views may differ significantly.

Each network, also called AS (Autonomous System), is used by its owner for business purposes; these can include providing Internet access or connectivity, content or cloud services, and infrastructure services. Based on these business goals, organizations establish connections with each other, and on top of those connections, routing policies are created in order to constrain the set of paths over which data is allowed to flow. Hence, the set of ASes and AS paths can be seen as a path-based network, in which the paths are determined by algorithmic computations realized in BGP (Border Gateway Protocol), the standard interdomain routing protocol of the Internet, combined with policies driven by business strategies. Because Internet routing is a complex process, driven by commercial as well as engineering concerns, it is both important and difficult to fully characterize.

There are many aspects of the interdomain routing system that are important to

understand, including its stability, scalability, and security. However, a particularly difficult problem is to understand the overall structure of interdomain routing and how it changes, and evolves, over time. The immense size, complexity, and continuous growth of the system make it challenging to gain a useful understanding of the nature of routing changes.

Therefore, the general goal of this dissertation is to provide means, tools, and methods to characterize the dynamics of the interdomain routing system of the Internet. Our contributions are threefold: first, we propose a metric to expose rates of changes; second, we propose a tool to identify and analyze high-impact routing events; and third, we show how to detect unusually routed ASes in the Internet. Each one of these parts is summarized in the next three sections.

1.1 Rate of change

As a first step, we are interested in characterizing the *rate of change* of the routing system and the dynamics of its change. One reason that temporal change of the routing system has not been extensively studied to date is that good methods and metrics to study it have not existed. Accordingly, one contribution that we make is to propose a metric that can be used to capture the dynamics of routing change, and show how to apply it to available BGP data. The key idea behind our approach is that the entire state of the global routing system at any time can be captured as a large set of tuples that express the next-hop decisions made by each AS with respect to each prefix. From this starting point, we define a natural measure of change for this set of next-hop decisions, and then show the utility of the measure.

This kind of approach to the study of interdomain routing moves away from studying the system in terms of an AS topology. Rather than focusing on an imperfectly understood topology, we focus on the fundamental decisions that are made by each

AS, namely, its next-hop choices. Further, our approach also avoids the problems of studying routing in terms of BGP update messages, which are very difficult to interpret globally. Analyzing BGP updates is problematic, requiring special methods and heuristics, because BGP traffic represents a variety of effects (table dumps as well as route changes) and because many messages do not result in actual changes to the routing systems (e.g., updates that do not trigger new next-hop decisions in the recipient).

The new metric we develop is based on the general concept of *Routing State Distance* (RSD), studied in prior work. However, we show that RSD as originally used is insufficient to study the dynamics of a system that changes over time. Hence, we develop a significant improvement to RSD, called *Multiple Next-hop Routing State Distance* (MRSD). MRSD has the advantage of being applicable in a much wider set of routing systems than the original RSD, while still capturing the same concept. Further, whereas the original concept of RSD was developed to compare different prefixes, in this study we apply MRSD to compare routing to the *same* prefix at *different times*. We refer to the application of MRSD over time as *Temporal RSD* (TRSD).

We use TRSD to address our motivating questions by applying it to a large corpus of publicly available BGP data. Our results reveal some interesting aspects of Internet routing. We show that approximately 1% of all visible next-hop decisions change each day, and that this rate is about 10% at the timescale of one month and 50% at the timescale of 2 years. We show that these values are remarkably constant over the period investigated, despite the immense change and growth of the network during our period of study. Furthermore, we show a decomposition of the daily TRSD time series in two components, intended to capture the difference between sustained (policy-driven) changes in routing versus churn (temporary changes, e.g., due to equipment

failures). About 2/3 of changes on a daily basis fall into the sustained category, with the remaining 1/3 classified as churn. We show that sustained changes show a strong weekly periodicity, with the majority of sustained changes made in the workweek. On the other hand, routing churn is better described as noise without a strong periodic component. Finally, we also study the relative rate of routing changes across different ASes, where we show that this rate is very long-tailed, i.e., that a small fraction of ASes is responsible for the vast majority of changes to next-hop decisions.

1.2 High-impact events

Each routing change made by an individual AS is in response to some discrete event, such as a link failure or addition, a peer's route announcement or withdrawal, or a policy change. However, the complexity of the resulting dynamics means that the causal relationship between routing changes in different parts of the system is notoriously difficult to tease out.

In this part of the dissertation, we present PathMiner, a system for identifying large-scale changes to the state of the routing system that are caused by individual events, and for narrowing down network elements (ASes or links) responsible for the set of changes. By 'large-scale', we mean routing changes that involve many ASes and prefixes, and may re-occur at multiple times.

The central idea behind PathMiner is that when a set of ASes change their next-hop decisions to a set of prefixes in a coordinated fashion, especially when those same changes are repeated at multiple points in time, then it is very likely that the coordinated activity is ultimately caused by actions taken by a single AS or link. This is an application of Occam's Razor: when a large set of ASes all change their next-hop decisions for a large set of prefixes, it is unlikely to be a coincidence. Rather, the simplest explanation is that all the changes were ultimately triggered by the action

of a single ‘actor’ (AS or link). Furthermore, as the size of the AS set and prefix set involved grows, causation by a single actor becomes even more likely. Hence, PathMiner looks for significant spatio-temporal patterns in BGP routing, extracts them from background noise, and identifies the network element most likely to be responsible for generating the pattern.

We formalize the concept of high-impact routing events, and show how to translate the discovery of such events into the Boolean Tensor Factorization problem. Therefore, the first component of PathMiner is a new algorithm for Boolean Tensor Factorization that is well suited for the kind of data that is derived from network routing changes.

The second component of PathMiner identifies the single actor that is responsible for each event. This second step crucially depends on the fact that the first step extracts a set of coordinated routing changes. The key insight is that over the set of all paths that participate in the routing changes, the network element having the highest precision and recall as a classifier for changed paths is most likely the single actor responsible for the event.

We validate PathMiner by manually inspecting the extracted events and actors. For this we developed an automated tool for graphical reconstruction of the event, which depicts the changes made to the subgraph that is induced by the set of ASes and prefixes involved in the event. While manual inspection is time-consuming and imperfect, we know of no alternative, since existing systems for root-cause analysis are not capable of working with historical data, nor with sets of large-scale routing changes. Our validation finds that the actors identified for each event almost always agree with the manual analysis.

Using PathMiner we perform an initial analysis of the last 9 years of interdomain routing data, sampled at a daily granularity. We show that PathMiner is capable

of extracting large events, some of which involve over 100,000 coordinated routing changes. Taken together, these events constitute between 10% and 20% of all visible routing changes over time in the datasets that we analyzed. Individual events can involve tens to hundreds of ASes and prefixes, and occur tens to hundreds of times in our data. For most of these events, PathMiner is able to identify a single actor (or a small set of actors) that is likely responsible for ultimately causing this coordinated activity.

One of the main contributions of our work is to provide evidence that large-scale events do exist and they also re-occur over long periods of time. Specifically, PathMiner exposes the existence of regions of the AS-level Internet that have similar dynamics towards sets of prefixes. To the best of our knowledge PathMiner, is the first tool capable of exposing such facets of the Internet at a global scale. From an engineering point of view, such information may be valuable for network administrators, when making changes in their systems, by providing a historical view of events related to similar actions.

1.3 Unusually-routed ASes

In this part of the dissertation, our questions can be broadly grouped into *micro* and *macro* levels. At the micro level, we are interested in identifying and understanding *unusually-routed* ASes – reached through a set of paths that can be considered *unusual*, when compared to the remaining of the network. Furthermore, we are interested in how business and engineering decisions of individual ASes are reflected in their decisions to adopt unusual routing structures.

We develop tools for answering these questions, and using them we show that unusually-routed ASes are very likely to be economically important. Further, we show that these unusually-routed ASes form *clusters*, and that the clusters often

consist of ASes owned by the same organization. When an AS is unusually-routed, it tends to be highly-connected. However, the unusual aspect of such an AS (or group of ASes) is not simply that it is highly-connected (a network property); it is that paths to the AS do not make typical use of the Internet’s hierarchy (a path property). Our analysis of individual ASes shows that there are a variety of reasons why an AS may employ unusual routing, and allows us to track how individual ASes adopt unusual routing strategies over time, including infrastructure build-out for content delivery and anycast.

Driven by the results at the micro level we ask questions related to the macro level as well. Here we seek to understand the high-level evolution of the set of all paths in the network over time. We approach this two ways. First, we start from the observation that unusually-routed ASes are often significant Internet businesses. Hence, we identify the set of unusually-routed nodes at each point in time, and use those sets to identify phases of Internet evolution. We show that a segmentation of unusually-routed AS sets yields five phases over 13 years. Digging into the kinds of ASes in each phase, we see that the most unusually routed ASes have shifted over time: from those delivering network operations services, to those involved in content delivery, to those involved in cloud services and domain registry.

The second question at the macro level is driven by the observation that many (but not all) unusually-routed nodes are contributing to the so-called ‘flattening’ of the Internet. This is the trend of major ASes to move away from hierarchical routing and towards more mesh-like routing. To explore this, we develop metrics to measure quantitatively the process of Internet flattening over time. We show evidence suggesting that Internet flattening has been taking place fairly consistently over the 13-year period of our study, predating the first reports in the literature. We back up our measurements with theoretical and simulation results, and we discuss their

robustness due to missing data (i.e., unknown AS-links).

1.4 Roadmap

The remaining of this dissertation is organized as follows:

- Chapter 2: contains a brief description of the interdomain routing system of the Internet;
- Chapter 3: discusses how this dissertation differs from related work;
- Chapter 4: presents the first part of the dissertation, containing the definition of TRSD, and how we used it to analyze the Internet;
- Chapter 5: presents the second part of the dissertation, where we describe PathMiner;
- Chapter 6: presents the third part of the dissertation, where we detect unusual routing strategies, and show how they correlate with the evolution of the Internet;
- Chapter 7: summarizes the content of the dissertation; discusses implications, directions for future work, and final remarks.

1.5 Related publications

The work presented in this dissertation is related to three publications, which are listed below. With regard the most recent one, this dissertation contains material that is not present in the original publication.

Giovanni Comarela, Evimaria Terzi and Mark Crovella. *Detecting Unusually-routed ASes: Methods and Applications*. ACM IMC 2016.

Giovanni Comarela and Mark Crovella. *Identifying and Analyzing High Impact Routing Events with PathMiner*. ACM IMC 2014.

Giovanni Comarela, Gonca Gürsun, and Mark Crovella. *Studying Interdomain*

Routing over Long Timescales. ACM IMC 2013.

Chapter 2

Background: Interdomain Routing

The goal of this chapter is to provide background information for a better understanding of the next chapters. A reader familiar with concepts related to interdomain routing may proceed to Chapter 3. Section 2.1 gives a general idea of how the Internet is organized, highlighting the main differences between intra-domain and interdomain routing. Section 2.2 highlights some aspects of BGP (Border Gateway Protocol), and finally, Section 2.3 explains the most common types of AS (Autonomous Systems) relationships that can arise in the Internet. We do not aim at teaching BGP in this text. For more information, the reader can consult the works of Kurose and Ross (2012), Tanenbaum and Wetherall (2010), and Stewart (1998).

2.1 Internet organization

The Internet is commonly referred as a network of networks. Each network may be owned by a different organization and exists with a different purpose (e.g., education, content delivery and provision of Internet services). Each organization has autonomy over its networks, and for that reason, such networks are referred as Autonomous Systems (AS or ASes in the plural).

Organizations have the freedom to decide how information (i.e. packets) flows inside their AS. In other words, they can choose hardware configuration, interconnection topology, and routing protocols. With regard to the latter, the general goal is to optimize some performance metric related to the cost of a packet being sent from a

source to a destination, both inside the AS. Popular routing protocols currently used in the Internet are RIP (Routing Information Protocol), OSPF (Open Shortest Path First) and IS-IS (Intermediate System to Intermediate System).

A different challenge arises when source and destination of a packet belong to distinct ASes, i.e., distinct organizations. In this case, the path traversed by a packet depends not only on performance metrics of each AS, but also on the way that ASes connect to each other and their routing policies. More specifically, due to routing policies, it is possible that the best path (according to some performance metric) is not among the set of available paths.

The task of deciding the possible AS paths is called interdomain routing, and currently, the protocol used for such task in the Internet is called BGP (Border Gateway Protocol).

2.2 The Border Gateway Protocol

Currently, BGP, as described in the RFC 4271 (Rekhter et al., 2006) and extensions, is the standard interdomain routing protocol in the Internet. It has two main goals. First, BGP is a scalable solution for global reachability, i.e., it ensures that every public subnetwork (composed by a collection of IP addresses – or IP prefixes) is visible and reachable from every AS.

Second, BGP allows ASes to express routing policies and preferences, which cannot be captured by intra-domain routing protocols. Simple examples of motivation for routing policies are:

- AS x pays ASes y and z for Internet connection. Therefore, x does not want to route traffic from y to z (and vice-versa). Otherwise, x would be paying to provide a service that does not generate any benefit to itself;
- AS x does not want its traffic routed through its competitors;

- preference of sending traffic through ASes that charge less for transit services;
- a country may be willing to have its national security traffic not flowing through enemy nations.

Routing through BGP is performed by propagation of information from each AS to its neighbors. In a high level, each AS learns reachability information from neighbor ASes. Such information is composed of at least three items: destination subnet; AS path, i.e., the sequence of ASes that must be traversed to reach the AS that owns the destination subnet; and the router's interface that must be reached in the neighboring AS in order to start the AS path. Then, based on the current routes toward the subnet and on routing policies, the appropriate routes towards the destination are determined. Finally, the determined routes can be announced to AS neighbors, again, depending on the policies established by the AS administrator. If the AS decides for the announcement, then it adds its own AS number to the AS-path.

There are no strict rules in the Internet saying how ASes should connect to each other. In general, the interconnection process happens through bilateral or multilateral agreements, which can be implemented in different types of facilities, e.g., PoPs (Points of Presence) or IXPs (Internet eXchange Points). Many factors may influence how an AS selects interconnection strategies and routing policies. In the next section, we discuss some policies that are well-known in the literature and especially relevant for a better understanding of Chapter 6.

2.3 AS relationships

The most basic relationship between two ASes is *customer-provider*. In such case, an AS x pays another AS y , which is already connected to the Internet, for transit services. In general, x is smaller (e.g., in terms of geographical reach, network size, and traffic amount) than y , and x depends on y for:

- communication with subnetworks in the Internet that x could not reach (including addresses from y); and
- exposure to the Internet (including y) of subnetworks owned by x .

In Figure 2-1, for example, AS F is a customer of D , and since F has only one provider, it is said it is a *single-homed* AS. AS D on the other hand, is said to be *multi-homed*, because it has more than one provider, namely ASes A and B .

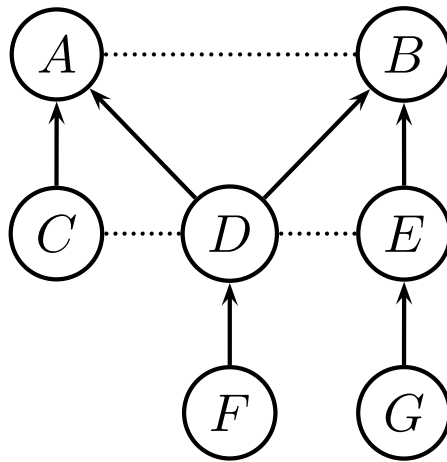


Figure 2-1: Example of possible AS interconnections and relationships. Solid arrows are for customer-provider (from customer to provider), and dotted lines for peering relationships.

Another common type of AS relationship is *peering*. In this case, ASes x and y have a settlement-free agreement, where they exchange traffic, but do not pay each other. This type of relationship is more likely to happen when:

- x and y have the opportunity to connect to each other; and
- the amount of traffic that x needs to send to y is approximately the same amount that y needs to send to x .

When such conditions are met, it is not in the best interest of x or y to pay providers in order to exchange traffic. Instead, it makes more economic sense for them to connect directly to each other, thus bypassing transit providers and saving money.

However, it is not interesting for y to carry traffic between its providers and x . If y were to proceed in that way, it would be spending money in order to provide a free service for x .

In recent years, peering relationships have also become common in a different type of situation. Whenever possible, large content providers peer with other networks in order to avoid paying huge amounts of money to transit providers for the sake of content diffusion. For the same reason, this peering opportunity is also welcome from the point of view of many other networks, e.g., small transit providers and access networks. However, despite having the same motivation, the situation is not the same as presented in the last paragraph. In fact, the amount of traffic sent between peers is not symmetric, i.e., the majority of the content flows from the content provider to the other networks.

The intuition behind the process of establishing interdomain routing policies followed in the last paragraphs is known (and formalized) in the literature as the Gao-Rexford rules (Gao and Rexford, 2001). Basically, such rules state that routes learned from customers should be advertised to peers and providers, while routes learned from peers and providers should only be advertised to customers. Moreover, they state that customer's routes should be preferred over peer's routes, which should be preferred over provider's routes. Gao and Rexford show that there are economic and technical reasons for ASes to follow the Gao-Rexford rules.

Unfortunately, AS paths obtained by using the Gao-Rexford rules are not always the shortest (or optimal with regard to some performance metric). For instance, in Figure 2-1, the Gao-Rexford path from AS C to AS E is $A - B - E$, while the shortest path is given by $D - E$. The reason that makes the shortest path unavailable in this case is that AS D should not announce to AS C (a peer) a route learned from AS E (another peer of AS D).

Based on the previous concepts, next, we present some relevant definitions. The set of Ases that can be reached from an AS x through paths which traverse only edges from providers to customers is defined as the *customer cone* of x . Also, an AS x is defined as a *stub* AS if all its interdomain traffic originates at x , or has x as a destination. In Figure 2-1, assuming a Gao-Rexford routing model, the customer cone of B is $\{D, E, F, G\}$, and C , F and G are stub ASes.

Chapter 3

Related Work

There is a considerable amount of literature about many aspects of BGP due to its importance to the global Internet. In this chapter, we position our work and contributions with regard to similar problems and related areas.

3.1 Analysis of the interdomain topology

One of the main characteristics of this dissertation is the way we choose to represent the state of the interdomain routing system. More specifically, we move away from a graph-based towards a path-based representation. Such shift is important because a modeling the system as a path-based network allows us to capture more of its details, and to define problems and algorithms more precisely.

A number of studies have analyzed the AS-level Internet by using graph metrics in static or dynamic fashion (Edwards et al., 2012; Carmi et al., 2007; Mahadevan et al., 2006; Gregori et al., 2013). The main idea is to model the interdomain topology as graph $G(V, E)$, where V is the set of ASes and $(x, y) \in E$ when ASes x and y connect to each other. One classical example is the work by Faloutsos et al. (1999), where it was shown that the structure of the Internet, as a graph, had several characteristics following power-law distributions. Later, insights from such power-law relationships were used by many researchers in order to create topology generators that were representative for the Internet (e.g., the work of Medina et al. (2001)).

Using graph methods to analyze the interdomain topology can provide many in-

sights about the underlying structure over which data flows. However, it does not take into account the economic relationships between ASes, nor the paths that data is actually allowed to traverse (Roughan et al., 2011).

Another common way to look at the interdomain topology is to consider a graph in which the edges are annotated with relationships. In other words, each edge of the graph is labeled, most commonly, as a *peering* or *customer-provider* edge. Then, under the assumption of the Gao-Rexford routing model, it is possible to infer BGP paths, and use them in a variety of scenarios. Unfortunately, labeling the edges of the graph is not easy, because AS relationships need to be inferred from real data (Luckie et al., 2013; Gao, 2001). Moreover, recently, it has been shown that AS relationships have become more complex and harder to infer (Giotsas et al., 2014, 2015), and that common routing assumptions, like the Gao-Rexford model, do not explain the Internet paths as well as they did years ago (Anwar et al., 2015).

A different, but related, line of thought states that in order to properly understand and model the structure of the interdomain topology (and its formation process), it is necessary to start from first principles. In other words, it is necessary to understand what each network demands, what it can offer, its business strategies, its opportunities for interconnection, etc. Works in that direction include Lodhi et al. (2012, 2014) and Dhamdhere and Dovrolis (2010).

Our work is agnostic to graph-based representations and AS relationships. We recognize that the state of the routing system can be described as the set of paths used in the network at a given time. Then we develop methods to mine and understand such set of paths, and how it changes over short and long timescales. It is important to mention that some studies have carefully analyzed AS paths, to characterize their behavior (Broido and claffy, 2001; Rexford et al., 2002) or to understand routing policies (Mühlbauer et al., 2007, 2006). In contrast to those papers, our work provides

metrics and methods that are useful in analyzing path-based networks in general, and focuses on different questions.

3.2 Dynamics of interdomain routing

The state of the routing system of the Internet is not fixed, and creating means to understand its dynamics better is one of the main goals of this dissertation. In this context, routing changes can be perceived in different granularities and timescales. In this section we position our work with regard to three aspects of the dynamics of the interdomain Internet: in Section 3.2.1 we cover BGP instability measurements; in Section 3.2.2 we look at works related to high-impact events and root cause analysis; and in Section 3.2.3 we show the literature related to long-term evolution – focusing on one specific phenomenon, the flattening of the Internet.

3.2.1 BGP instability measurement

Considerable prior work has looked at the stability (or instability) of the interdomain routing system (Papadimitriou et al., 2011; Labovitz et al., 1998; Rexford et al., 2002; Li et al., 2007; Elmokashfi et al., 2012; Livadariu et al., 2016). On one hand, these studies are related to ours in also performing long timescale studies of the BGP system. However, these studies do not emphasize the global evolution of routing *decisions*, but focus mainly on the dynamics of routing *traffic*. They focus on characterization and analysis of BGP messages: how to understand BGP traffic dynamics over time, and how to characterize stability of BGP routing for specific destinations. Our focus, especially in Chapter 4, is on the changes of next-hop decisions made throughout the global routing system.

3.2.2 Event detection and root cause analysis

In contrast to work quantifying and characterizing path and topology changes, there is also a vast amount of literature investigating specific events and their causes. Studying such events is not only important to understand the day-to-day natural changes of the network, but also to detect and learn how to protect the network from threats, such as hijacking (Khare et al., 2012; Wählisch et al., 2012; Shi et al., 2012) and rerouting (Arnbak and Goldberg, 2015).

Most work on event detection relies on BGP update messages in order to analyze path changes. For example, Wu et al. (2005) proposed a methodology to identify high impact BGP routing changes. However, their scope is different from ours, since in their context the term *high impact* is related to the impact on the network traffic leaving a specific ISP (Internet Service Provider) and not in terms of changes propagating throughout the network. Lad et al. (2004, 2006) proposed a way to identify temporal event boundaries in the stream of update messages and a visualization tool that allows users to narrow down and infer root causes.

Glass et al. (2010) use tensor factorization techniques to infer events in the stream of BGP updates. However, since they use a path-based representation they are restricted to a small set of monitors. Moreover, since they work with data obtained at the granularity of minutes their strategy is not able to identify large scale events involving hundreds of sources, possibly months apart.

In Chapter 5 we propose PathMiner, a system to identify and analyze high-impact events in the interdomain routing system. PathMiner does not rely on inferring the state of the routing system by using update messages. In fact, this is a hard task, since in most cases internal AS policies are unknown. Moreover, it is important to mention that working with BGP updates demands extra processing in order to clean the data. In our approach, we use BGP RIBs (Routing Information Base), i.e., snapshots of the

interdomain routing system, which avoids the complex process of update messages cleaning and allows PathMiner to scale over long timescales.

The second component of PathMiner consists of a technique to identify (or at least narrow down) possible ASes (or links) triggering large scale events. Although sounding related to root cause analysis, as in Javed et al. (2013) and Feldmann et al. (2004), it is important to remark that our requirements and assumptions differ significantly. On one hand, general root cause identification systems are real time, work with BGP update messages (sometimes with information coming from external sources) and are interested in identifying causes of any path changes. On the other hand, differently of PathMiner, those systems are not capable, of identifying large events and narrowing down causes using just routing tables.

3.2.3 Evolution and flattening of the Internet

A different aspect of the dynamics of the interdomain routing systems is related to how its state changes and evolves over long timescales. Dhamdhere and Dovrolis (2008, 2011) and Dhamdhere et al. (2012) analyzed the growth of the Internet over a period of more than 10 years, focusing on the economic roles played by each AS, the types of relationship between ASes, and differences across geographical regions. Our work, particularly in Chapter 4, complements those studies by looking at similarly long timescales, but focusing on more basic questions such as the rate of change in routing, the nature of routing churn, and the ASes responsible for the most routing changes.

One phenomenon with regard to the evolution of the interdomain routing system is known as the *flattening of the Internet*, which is the trend of many, including large, ASes to move away from hierarchical routing and towards more mesh-like routing. The main advantage of the hierarchical organization was to allow a global and cost-effective network (Calvert et al., 1997; Subramanian et al., 2002; Siganos et al., 2006;

Carmi et al., 2007). However, seeking lower latencies and to cut costs with transit providers, many ASes engaged in different interconnection strategies, resulting in the shift of the routing structure of the Internet.

The flattening of the Internet has been the subject of a number of previous studies. Gill et al. (2008) were the first to document the phenomenon, using active measurements from 50 nodes. Xiang et al. (2011) explore a dataset composed of AS paths collected from 2005 to 2009, in order to show further evidence that the Internet is flattening. Labovitz et al. (2010) document flattening using a larger set of networks, and focus on changes in traffic patterns. Luckie et al. (2013) show that the peering density of customer cones is increasing, which translates into a shift away from a tree-like structure. In the case of one specific AS that is contributing to flattening, Calder et al. (2013) and Chiu et al. (2015) showed that Google front-end servers are present all over the globe, and that Google’s AS had more than 5000 peers by March 2015. Finally, Ager et al. (2012) show that the popularization of IXPs is another important driver of the flattening.

In contrast to these efforts, we focus on developing metrics to directly analyze the paths in a path-based network, and using these metrics we quantify flattening effects at the macro level (across all paths) as well as the micro level (with respect to individual ASes).

3.3 Incompleteness of the known Internet topology

Most of the analyses performed throughout this dissertation rely on publicly available sources of data, which do not contain all Internet paths (or even complete AS-level topology). A considerable amount of work has exposed the problem of missing information (links and nodes) in common sources of data related to the Internet topology (Oliveira et al., 2010, 2008a,b). Although the problem is well-known in the networking

community, solving it is not an easy task. In fact, many researchers have approached the problem in different ways (Chen et al., 2009; Augustin et al., 2009; He et al., 2009; Gregori et al., 2012, 2015; Khan et al., 2013). The challenge arises because many organizations treat their routing tables with secrecy. Hence, there is no centralized repository containing the set of paths from every source to every destination. A common approach to circumvent that issue has been using active measurements from nodes in several parts of the globe in order to traverse as many AS links (and cover as many distinct AS paths) as possible. However, the increasing popularity of IXPs (Internet eXchange Points) makes such task a non-trivial one (Ager et al., 2012; Gregori et al., 2011; Chatzis et al., 2013; Brito et al., 2016). Despite all the challenges, there are many recent efforts towards providing reliable sources of data to the community, by creating new data collection methods, or by aggregating existing sources in more convenient ways (Orsini et al., 2016; Nomikos and Dimitropoulos, 2016; Singh and Gill, 2016; Cunha et al., 2016).

Our work is subject to the same limitations as previous studies with regard to missing data, and could benefit from a more detailed topology and set of AS paths. However, the methods and sampling strategies developed in this dissertation are designed to reduce the effect of missing information. Moreover, we show evidence in Chapter 6 that our results do not appear to be strongly affected by missing AS links.

3.4 Routing State Distance

One of the main tools in this dissertation is a metric called RSD (Routing State Distance), which aims at comparing two prefixes with regard to the way they are reached in the Internet. RSD was originally used by Gürsun et al. (2012b,a) with the goals of inferring missing data and for identifying similarly-routed sets of ASes (“local atoms”).

In this dissertation, we build an additional set of tools (metrics and algorithms) on top of RSD to explore two different aspects of the interdomain Internet. In Chapter 4 we present a generalization of RSD, which extends the original metric to work with additional routing configurations that RSD could not handle (while remaining equivalent to RSD for any cases in which RSD can be used). Similarly, in Chapter 6 we extend RSD in order to detect ASes that are anomalous with respect to the set of routes used to reach them.

Chapter 4

Studying Interdomain Routing over Long Timescales

The dynamics of interdomain routing have traditionally been studied through the analysis of BGP update traffic. However, such studies tend to focus on the volume of BGP updates rather than their effects, and tend to be local rather than global in scope. Studying the global state of the Internet routing system over time requires the development of new methods, which we do in this chapter. We define a new metric, MRSD, that allows us to measure the similarity between two prefixes with respect to the state of the global routing system. Applying this metric over time yields a measure of how the set of total paths to each prefix varies at a given timescale. We implement this analysis method in a MapReduce framework and apply it to a dataset of more than 1TB, collected daily over 3 distinct years and monthly over 8 years. We show that this analysis method can uncover interesting aspects of how Internet routing has changed over time. Furthermore, we show that on any given day, approximately 1% of the next-hop decisions made in the Internet change, and this property has been remarkably constant over time; the corresponding amount of change in one month is 10% and in two years is 50%. Digging deeper, we can decompose next-hop decision changes into two classes: churn, and structural (persistent) change. We show that structural change shows a strong 7-day periodicity and that it represents approximately 2/3 of the total amount of changes.

The remaining of this chapter is organized as follows. Section 4.1 presents the

notation and definitions while Section 4.2 presents our data collection methodology. Measurement results are presented in Section 4.3 and finally, discussions, conclusions, and future work directions are stated in Section 4.4.

4.1 Notation and Definitions

In this section, we define the metrics we used to analyze changes in the global routing system.

4.1.1 Multiple next-hop RSD

Our starting point is the notion of Routing State Distance (RSD) as defined by Gürsun et al. (2012b). Briefly, RSD is a metric that defines the ‘distance’ between two destinations (e.g., prefixes) as the *number of nodes* which choose different next-hops for the two destinations.

While this general concept is a good starting point for studying routing changes, it has drawbacks. The main problem with RSD as defined by Gürsun et al. (2012b) is that it assumes that each node in the network has a unique next-hop towards any destination. In interdomain routing, where nodes are ASes and destinations are prefixes, this property does not hold. Gürsun et al. (2012b) decided to proceed as Mühlbauer et al. (2007), i.e., they partitioned Autonomous Systems in such a way that each of its parts (denoted “quasi-routers”) did not have more than one routing option to reach any prefix. Although that was a natural choice, this solution has a number of drawbacks. First, it can not be easily generalized to other routing systems with multiple next-hops, such as OSPF with Equal Cost Multi-Path (ECMP). Second, the problem of identifying quasi-routers optimally is NP-hard (Gürsun, 2013). Third, and most important for our study, there is no natural way to extend the quasi-router approach to routing systems that change over time, which introduces serious problems for longitudinal analyses.

Hence, we need to define a new metric that handles the case of multiple next-hops in a cleaner, more robust fashion: MRSD. MRSD avoids the problems of RSD by allowing next-hops decisions to be expressed as sets (instead of unique elements). MRSD then uses *Jaccard distance* to compare the next-hop sets.

Formally, let $G(V, E)$ be a directed graph where V is a set of vertices and E is the set of edges. For all source-destination pairs $(u, v) \in V \times V$ we define $N_{u,v}$ as the set of *next-hops* from u towards v . More formally, a vertex $w \in N_{u,v}$ if and only if, the edge (u, w) starts a path from u to v .

Definition 1. Let $G(V, E)$ be a graph and $S (\neq \emptyset)$ and P subsets of V denoting sets of sources and destinations respectively. For each pair $(d, d') \in P \times P$ we define the Multiple next-hop Routing State Distance (*MRSD*) over S by

$$MRSD(d, d') = \frac{\sum_{s \in S} \delta_s(d, d')}{|S|}, \quad (4.1)$$

where $\delta_s(d, d') = 1 - J(N_{s,d}, N_{s,d'})$ and $J(A, B)$ denotes the *Jaccard Index* of any pair of sets A and B .

The intuition behind MRSD is that when $MRSD(d, d')$ is close to 0 then d and d' are very similar in terms of the next-hop choices made by all nodes in the set S . On the other hand, when $MRSD$ approaches 1 we have that those destinations are routed very differently through the network. Note that MRSD is always a value between 0 and 1, and it can be interpreted as the *fraction* of next-hop decisions that differ, across all nodes in the network. It is also important to remark that when all destinations always have unique next-hops ($|N_{s,d}| = 1$ for all s and d), this definition is equivalent to the one presented by Gürsun et al. (2012b), and in that case, MRSD reduces to (normalized) RSD.

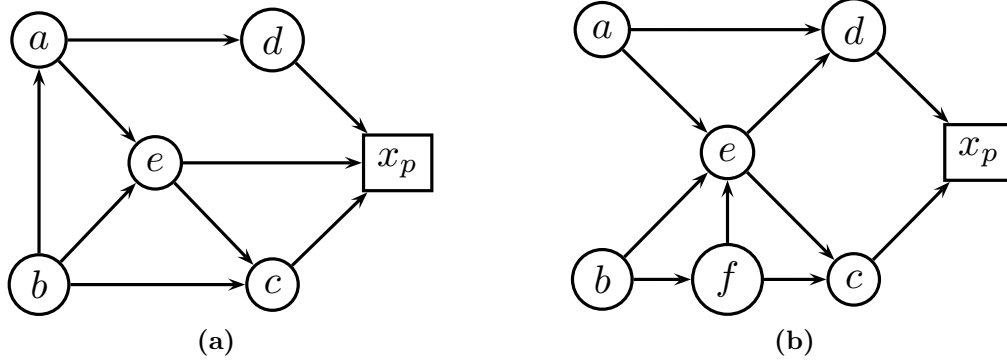


Figure 4-1: Differences between routing decisions towards prefix p , owned by AS x_p , at: (a) time t_i ; and (b) time t_{i+n} . ASes b and e change their next-hop choices, while f is a new AS in the network.

4.1.2 Temporal RSD

Next, we show how to apply the definition of the last section in the context of inter-domain routing on the Internet in order to conduct a longitudinal analysis.

Consider P and S being respectively a set of IP prefixes $\{p_1, \dots, p_{|P|}\}$ and sources (autonomous systems) $\{s_1, \dots, s_{|S|}\}$ on the Internet seen by a set of monitors at times t_1, \dots, t_T ($t_i - t_{i-1} > 0$ for $i = 2, \dots, T$). We define $N_{s,p}(t)$ as the set of next-hops (autonomous systems) that source s may use at time t in order to reach prefix p , where $p \in P$, $s \in S$ and $t = t_i$ for some $i \in \{1, \dots, T\}$.

Definition 2. For a prefix p we define the Temporal Routing State Distance (TRSD) over a set of sources S between time t_i and t_j ($1 \leq i < j \leq T$) as:

$$TRSD_p(t_i, t_j) = \frac{\sum_{s \in S} \delta_{s,p}(t_i, t_j)}{|D_p(t_i, t_j)|}, \quad (4.2)$$

where $\delta_{s,p}(t_i, t_j) = 1 - J(N_{s,p}(t_i), N_{s,p}(t_j))$ if both $N_{s,p}(t_i)$ and $N_{s,p}(t_j)$ are non-empty sets and zero otherwise, $D_p(t_i, t_j) = \{s \in S : N_{s,p}(t_i) \neq \emptyset \text{ and } N_{s,p}(t_j) \neq \emptyset\}$ and $J(A, B)$ denotes the Jaccard Index of any pair of sets A and B .

This definition is similar to the one presented in the last section, but has some key differences. The differences are: first, TRSD is specialized to the specific case

Table 4.1: TRSD computation for Figure 4-1.

s	$N_{s,p}(t_i)$	$N_{s,p}(t_{i+n})$	$\delta_{s,p}(t_i, t_{i+n})$
a	$\{d, e\}$	$\{d, e\}$	0
b	$\{a, c, e\}$	$\{e, f\}$	$\frac{3}{4}$
c	$\{x_p\}$	$\{x_p\}$	0
d	$\{x_p\}$	$\{x_p\}$	0
e	$\{c, x_p\}$	$\{c, d\}$	$\frac{2}{3}$
f	$\{\}$	$\{c, e\}$	0

of interdomain routing. Second, TRSD compares the next-hops of the same prefix in two different times instead of two different prefixes at the same time. And third, TRSD normalizes in a slightly different way, namely, the set S of sources becomes $D_p(t_i, t_j)$ which may vary with time. This was done by design, to address the fact that networks may grow over time.

To illustrate the definition, Figure 4-1 presents graphs related to the routing decisions towards a prefix p (hosted on AS x_p) at times t_i and t_{i+n} ($n > 0$). Table 4.1 shows the intermediate computations for TRSD over the set $S = \{a, b, c, d, e, f\}$.

From Table 4.1 we have that $|D_p(t_i, t_{i+n})| = 5$ and as consequence $TRSD_p(t_i, t_{i+n}) = 0.283$. This can be interpreted as saying that from time t_i to t_{i+n} , the network’s next-hop decisions changed 28.3% with regard to p . It is important to remark that, in this example, the last row (related to the source f) did not contribute anything to TRSD (because it was not considered in the composition of the set $D_p(t_i, t_{i+n})$ and $\delta_{f,p}(t_i, t_{i+n}) = 0$). This shows an important aspect of our definition, i.e., TRSD was designed with the intention of capturing routing *changes*, and hence does not increase simply due to the *growth* of the network.

4.2 Dataset Description

To explore the evolution of Internet routing over time, we collected four datasets comprising Routing Information Bases (RIBs) from RIPE Routing Information Services¹

¹<http://www.ripe.net/data-tools/stats/ris/ris-raw-data>

Table 4.2: Summary of the four datasets.

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Granularity	daily	daily	daily	monthly
First	01/01/05	01/01/08	01/01/11	01/01/05
Last	12/31/05	12/31/08	12/31/11	12/01/12
Size (GB)	200	500	680	160
$ S $	5,086	6,934	9,093	14,829
$ P $	316,519	517,773	616,714	1,157,670

and Route Views.² Datasets 1, 2 and 3 consist of all RIBs on a daily basis for the entire years of 2005, 2008 and 2011 respectively. The fourth data set consists of all RIBs for the first day of each month from 2005 to 2012. Since RIBs are made available at a coarser granularity than BGP updates (every 2 hours for Route Views and 8 hours for RIPE) we did not attempt to remove the effects of short term convergences, i.e., for each day, we kept all available distinct AS paths. It is important to remark that we collected data only for IPv4 prefixes.

Each RIB is a collection of records containing information about how to reach a prefix p from a specific autonomous system. From these records we extracted the following information: *route dumping date*, *autonomous system path* a_0, a_1, \dots, a_r ($r \geq 1$) and *destination prefix* p , hosted in the AS a_r . After that, we decomposed each record into r 4-tuples of the form $[route\ dumping\ date, a_i, a_{i+1}, p]$, for $i = 0, \dots, r - 1$. The semantics of each 4-tuple is: at the time of *route dumping date*, in order to reach the prefix p , AS a_i uses AS a_{i+1} as (one of its) next-hops. In the rest of the chapter, AS a_i will be referred to as a *source*, prefix p will be referred to as a *destination*, and AS a_{i+1} will be referred to as a *next-hop*.

To apply the definitions presented in Section 4.1 to this data, S is the set of all sources (a_i) and P is the set of all prefixes (p) found in a dataset. Table 4.2 presents a summary of each dataset collected after transformation into a collection of 4-tuples as described above.

²<http://www.routeviews.org>

To see the need for using Jaccard Index in Definitions 1 and 2, we note that, in our dataset on each day, approximately 7% of the sources have more than one next-hop choice towards a specific prefix.

4.3 Measurements

In this section, we show how TRSD can be used to extract useful knowledge about interdomain routing dynamics.

4.3.1 Analyzing TRSD

To analyze routing dynamics in the four datasets we compute time series of TRSD, averaged over all prefixes. Specifically, let $r_p(i, n) = TRSD_p(t_i, t_{i+n})$ for $1 \leq i \leq T-n$ and some $1 \leq n \leq T-1$. We then define $r(i, n)$ as the average of all $r_p(i, n)$ that can be computed, i.e., for all prefixes that have routing information at times t_i and t_{i+n} in our datasets. We use the average of $r_p(i, n)$ because it gives a measure of the total magnitude of change. Thus, $r(i, n)$ represents average proportion of change in next-hop decisions at timescale n .

Figure 4.2 presents the time series of $r(i, n)$ for our 4 datasets, where n is 1, 2, 7 and 30 days for datasets 1 to 3 (daily), and n is 1, 2, 6, 12 and 24 months for dataset 4 (monthly). From now on, to simplify the discussion, we assume that time indices always represent days for datasets 1, 2 and 3, and represent months for dataset 4.

The first striking aspect of these time series is that they are all approximately stationary. That is, they fluctuate around a mean value, but do not show any long-term trend. Furthermore, the mean values of TRSD do not show significant differences across years, from 2005 to 2012. This indicates that despite the considerable growth of the Internet in terms of ASes and Prefixes (which might suggest more next-hop options per AS) there is an approximately constant rate of routing decision changes over time. For example, for $n = 30$ days in the daily datasets, we have that $r(i, n)$

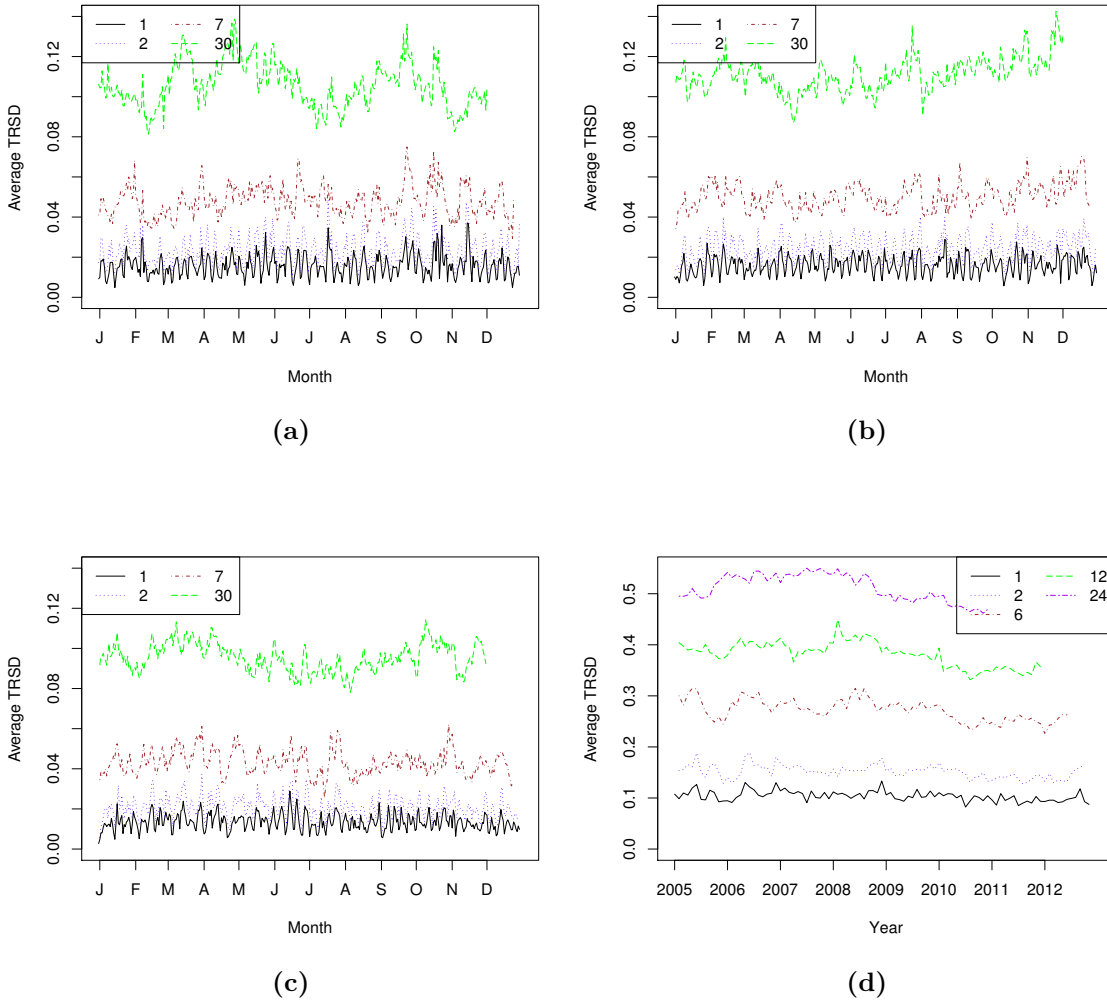


Figure 4.2: Average TRSD time series ($r(i, n)$ versus i) for different values of n and datasets: (a) 2005, daily; (b) 2008, daily; (c) 2011, daily; and 2005 – 2012, monthly. Time series are remarkably stable, and a considerable amount of changes are not undone quickly.

is approximately 0.1, meaning that from month to month, on average, 10% of the Internet changes in terms of next-hop routing decisions. The same value can be seen in the monthly dataset for the curve $n = 1$ month.

The next observation is that as n grows so does $r(i, n)$. This implies that at least a portion of the routing changes that happen over time are persistent, i.e., are not undone quickly to a previous routing state. On the other hand, we cannot say that it is a system governed by system-wide changes. To see that we can refer to the curve $r(i, 24)$ of the monthly dataset. From this curve, we can see that in a time window of 2 years, approximately 50% of the next-hop choices persisted (because 50% changed). This fact indicates many routes are stable over long periods.

Yet another property that can be seen in the daily datasets is related to seasonality. One can note that for $n = 1$ day there is a pattern of weekly variation in the data. This may be explainable as evidence of human input (network operators) in the system. We explore this conjecture in the following section.

4.3.2 Analyzing Routing Changes

After analyzing the results of the last section one question that arises is: of the total set of next-hop changes that happen over the time, what portion is related to *sustained* (changes that persist in the system for some time) and which fraction is related to *churn* (changes that change again in the near future)?

To answer this question we decompose the TRSD time series ($r(i, 1)$) into two components: $s(i, k)$ and $c(i, k)$, standing for *sustained* TRSD and *churn* TRSD respectively. Sustained TRSD ($s(i, k)$) captures the portion of $r(i, 1)$ that represents next-hop decisions that changed from t_i to t_{i+1} but did *not* subsequently change from t_{i+1} to t_{i+k} . In the same vein we define $c(i, k)$ as the portion of $r(i, 1)$ that represents next-hop decisions that changed from t_i to t_{i+1} and then changed *again* from t_{i+1} to t_{i+k} .

Formally, we define $s(i, k)$ as the average over all $s_p(i, k)$, where $s_p(i, k)$ is a measure for sustained changes for a specific prefix p given by:

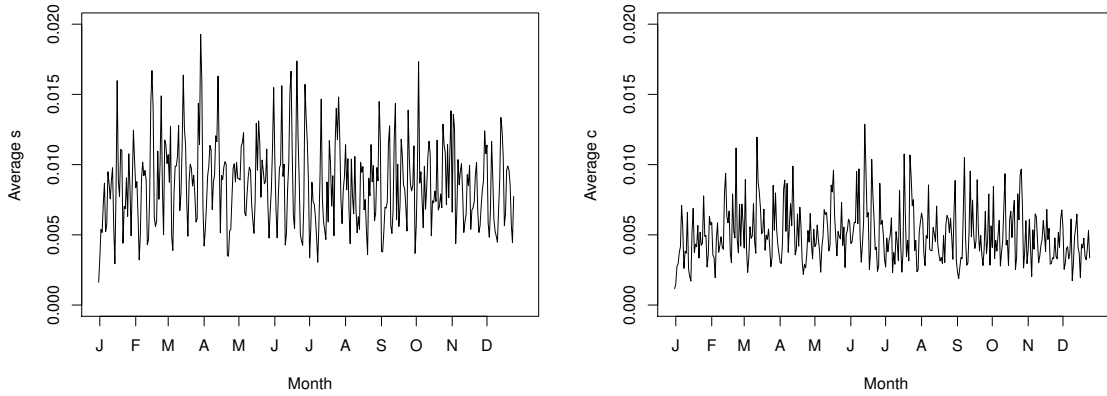
$$s_p(i, k) = \frac{\sum_{s \in S} \delta'_{s,p}(i, k)}{|D'_p(i, k)|}, \quad (4.3)$$

where $D'_p(i, k)$ is the set of all sources for which the sets $N_{s,p}(t_i)$, $N_{s,p}(t_{i+1})$ and $N_{s,p}(t_{i+k})$ are not empty and $\delta'_{s,p}(i, k)$ is the fraction of elements, which belong to the set $N_{s,p}(t_i) \cup N_{s,p}(t_{i+1})$, that satisfy any of the two following conditions: *i*) the element is in $N_{s,p}(t_i)$ but it is not in $N_{s,p}(t_{i+1})$ nor in $N_{s,p}(t_{i+k})$; or *ii*) it is in $N_{s,p}(t_{i+1})$ and $N_{s,p}(t_{i+k})$ but it is not in $N_{s,p}(t_i)$.

Then proceeding analogously to Equation (4.3) we can compute $c(i, k)$. One important observation is that $s_p(i, k)$ and $c_p(i, k)$, by definition, form a partition of $r_p(i, 1)$, i.e., $s_p(i, k) + c_p(i, k) = r_p(i, 1)$, for all p and k that have available routing information at times t_i , t_{i+1} and t_{i+k} .

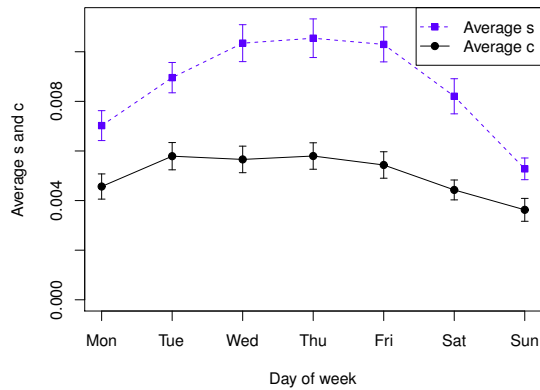
Figure 4-3 presents $s(i, 7)$ and $c(i, 7)$ for the daily dataset of 2011 (results for 2005 and 2008 are similar). The choice of $k = 7$ means that routing changes that persist for a week are considered to be sustained. The first observation, obtained by comparing Figures 4-3a and 4-3b, is that sustained routing changes ($s(i, 7)$) have more impact on TRSD than $c(i, 7)$. In fact, the average value of $s(i, 7)$ is around 0.007 (0.7%), while $c(i, 7)$ has an average value of approximately 0.004 (0.4%).

The next observation is that $s(i, k)$ inherits the weekly periodicity of $r(i, 1)$ while $c(i, k)$ is more similar to noise. This weekly periodicity suggests that there is some sort of human interaction with the system. Our conjecture is that those changes are triggered by BGP policy management and resulting changes to BGP configurations. In order to dig deeper, we computed the average of $c(i, k)$ and $s(i, k)$ over days of the week. Figure 4-3c presents the results, where we can see that both curves indicate more activity in the workweek and less on the weekends. However, the difference



(a)

(b)



(c)

Figure 4.3: Time series for sustained changes, $s(i, k)$, and churn, $c(i, k)$, for the 2011, daily dataset: (a) $s(i, k)$, $k = 7$ days; (b) $c(i, k)$, $k = 7$ days; and (c) weekly average of $s(i, 7)$ and $c(i, 7)$ – error bars are 95% confidence intervals. Sustained changes contribute more to TRSD, and most of the changes happen during the working days of the week.

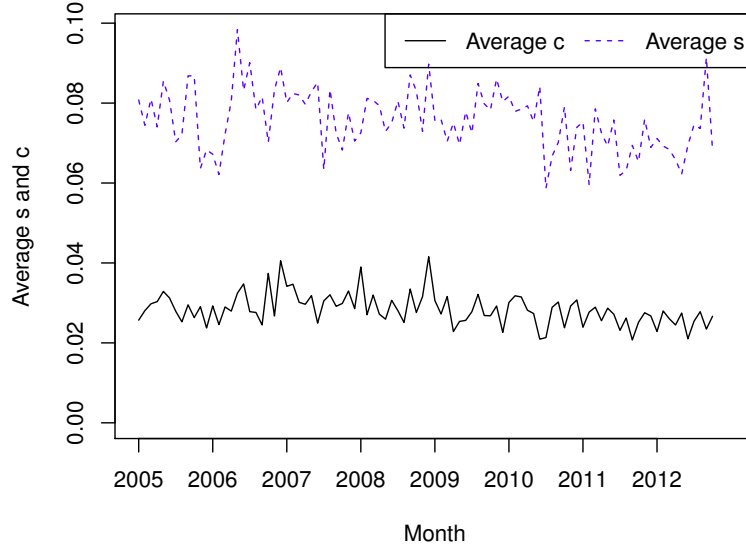


Figure 4.4: Time series for sustained changes, $s(i, k)$, and churn, $c(i, k)$, for the 2005 – 2012, monthly dataset, and $k = 2$. Even on a monthly timescale, sustained changes contribute more to TRSD.

between week and weekend behavior is much more significant in the case of the sustained changes, i.e., $s(i, k)$. These results corroborate with the idea of human interaction with the system, since we can understand them as the ASes avoiding intentional routing changes on weekends (probably due to reduced number of network operators working full time).

The decomposition into sustained TRSD and churn TRSD is also informative on long timescales. Figure 4.4 presents $s(i, 2)$ and $c(i, 2)$ for dataset 4 (2005 to 2012, monthly). Here we adopt the assumption that routing changes that persist for one month are considered to be sustained. This Figure shows that on average, about 3% of all routing decisions churn on a monthly basis, while about 8% of all routing decisions show a sustained change. Comparing Figures 4.4 and 4.3, one can see that the sustained and churn TRSD on a monthly basis are greater than the corresponding

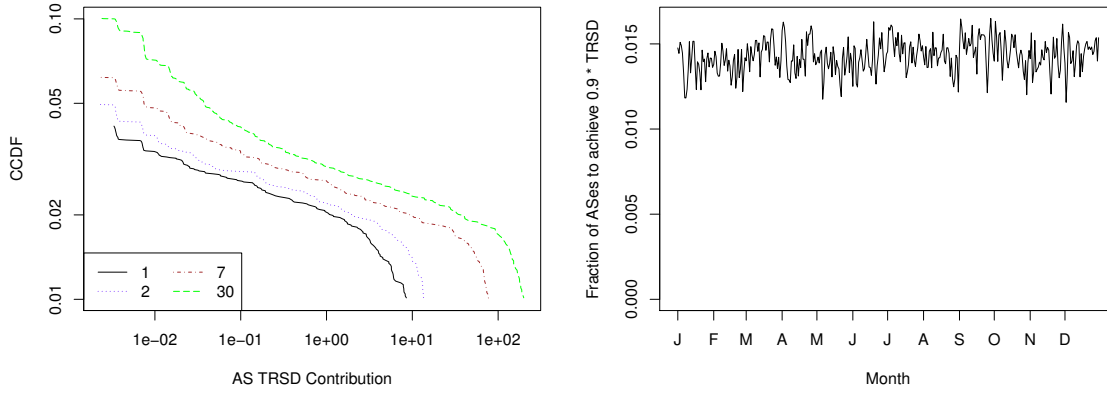
values on a weekly basis, which is to be expected. However, the difference between sustained and churn TRSD is much greater on the monthly timescale, showing that sustained TRSD captures the accumulation of intentional changes to the routing system over time, while churn TRSD reflects the continuous background noise in the system.

The relative stability of $s(i, 2)$ in Figure 4-4 shows that large-scale, system-wide changes to Internet routing are rare. There are only a couple of points in time in which there are noticeable peaks in the amount of sustained change in Internet routing – one peak in mid-2006 and one in late 2012. Initial investigation of these events shows evidence that system-wide routing changes took place: a large fraction of all prefixes was affected by next-hop changes during these events. A deeper investigation of these events is ongoing.

4.3.3 Contribution of Sources

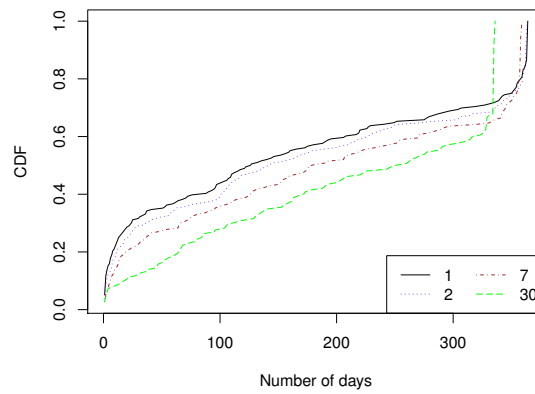
Our final set of results aims at answering the following question: *what is the contribution of each source AS to TRSD?* in other words, are the changes represented by TRSD uniformly spread over all ASes or are they concentrated in a small set? To start this analysis we sampled randomly chosen days in 2011 and computed the contribution of each AS over the total TRSD of that day. Figure 4-5a presents the Complementary Cumulative Distribution Function (CCDF) of this contribution for the first day of 2011 (results for other days are similar). The figure shows that the contribution of ASes is long-tailed, and that most ASes, approximately 90%, make little or no contribution to TRSD.

In order to inspect whether this is common behavior over time, for the entire year of 2011 we compute the fraction of ASes that are necessary to achieve 90% of TRSD. The result is presented in Figure 4-5b, where we can see that less than 2% of all ASes are in fact necessary to capture 90% of TRSD. This result holds for the other datasets



(a)

(b)



(c)

Figure 4.5: TRSD contribution per AS, 2011, daily dataset: (a) 01/01/11, for $n = 1, 2, 7, 30$; (b) Fraction of top contributors for $n = 1$; and (c) Number of days that each AS top contributor appears at Figure 4.5b, for $n = 1, 2, 7, 30$. Few ASes are responsible for most of TRSD.

with a threshold of 2.5%.

A question that may naturally arise then is: *are these heavy-hitter ASes the same over the time?* In order to answer it, we counted in how many days each AS appears during the computation of Figure 4-5b, in other words, in how many days each AS was among the set of sources responsible for 90% of the daily TRSD over 2011. The Cumulative Distribution Function (CDF) of this quantity is presented in Figure 4-5c. We can see that approximately 40% of ASes that appear on at least one day, appear in at least 300 days. Further, there is a group of approximately 20% of ASes that appear in all days. An initial investigation shows that many of them are in or are near to the network core.

4.4 Summary of the chapter

In this chapter, we presented results that uncover several aspects of the global Internet routing system. First, we showed that the rate of change in routing decisions has been stable over time, despite the growth in the network overall. Second, we showed how to decompose TRSD into components that reflect sustained change versus churn. We showed that the rate of sustained changes has a persistent weekly periodicity suggestive of a tendency of operators to make sustained (intentional) routing changes in the workweek. Finally, we showed that the locations of the routing changes in the Internet are generally concentrated among a small set of ASes, often those that are near or in the core. Throughout the entire study, we consider various timescales and show how the magnitude of routing change (both sustained and churn) varies with timescales from days to months.

Chapter 5

Identifying and Analyzing High-impact Routing Events

Understanding the dynamics of the interdomain routing system is challenging. One reason is that a single routing or policy change can have far-reaching and complex effects. Connecting observed behavior with its underlying causes is made even more difficult by the amount of noise in the BGP system. In this chapter, we address these challenges by presenting PathMiner, a system to extract large scale routing events from background noise and identify the AS or link responsible for the event.

PathMiner is distinguished from previous work in its ability to identify and analyze large-scale events that may re-occur many times over long timescales. The central idea behind PathMiner is that although a routing change at one AS may induce large-scale, complex responses in other ASes, the *correlation* among those responses (in space and time) helps to isolate the relevant set of responses from background noise, and makes the cause much easier to identify. Hence, PathMiner has two components: an algorithm for mining large scale coordinated changes from routing tables, and an algorithm for identifying the network element (AS or link) responsible for the set of coordinated changes.

We describe the implementation and validation of PathMiner. We show that it is scalable, being able to extract significant events from multiple years of routing data at a daily granularity. Finally, using PathMiner we study interdomain routing over past 9 years and use it to characterize the presence of large scale routing events and

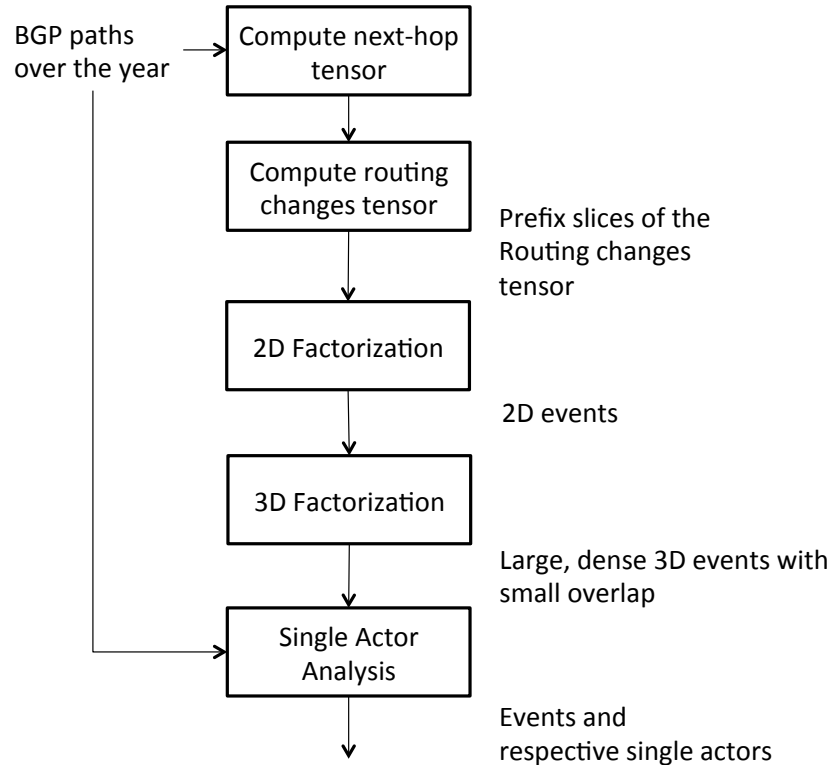


Figure 5.1: Stages of PathMiner.

to identify the responsible network elements.

A high-level view laying out the main stages in PathMiner is shown in Figure 5.1. In the rest of this chapter, we describe each of the stages shown in the figure. First, Section 5.1 uses an example to motivate the development of PathMiner. Section 5.2 presents a formal definition of our problem. In Section 5.3 we present and describe how to process raw BGP data (corresponding to the first two stages in Figure 5.1). In Sections 5.4 and 5.5 we describe and present results of our event detection methodology (the next two stages in Figure 5.1). Section 5.6 describes our single actor identification strategy (the last stage of the figure). Finally, we present concluding remarks in Section 5.7.

5.1 An Example

Before diving into the details of PathMiner, it is helpful to examine a typical example to provide intuition and motivate our approach.

Figure 5.2 shows a small subgraph representing the dynamics of a portion of the network with respect to routing towards two prefixes (hosted at AS42381 and AS44173, and shown in gray at the bottom). The figure captures routing dynamics over two consecutive days (April 30, 2013 and May 1, 2013).

This subgraph is a portion of an event extracted by PathMiner. The ASes along the top row of the figure and the prefixes at the bottom of the figure constitute the output of the first step of PathMiner. From the first to the second day, all the ASes at the top change their next-hops toward all the prefixes at the bottom. In fact, the full event is quite large, involving dozens of ASes that all change their next-hops; we have extracted these ASes which show behavior that is typical of all the others.

A directed edge in the graph denotes the fact that the first AS uses the second AS as its next-hop for (each of) the two prefixes. Black (solid) edges refer to edges seen in both days, red (dashed) edges refer to edges seen just on the first day and green (dotted) edges refer to those seen only in the second day. Inside each node, the negative (positive) number shows the number of paths passing through the node in the first (second) day of the event (Note that the path counts reflect the full event, which involves many ASes not shown.)

Figure 5.2 shows that on April 30, most of the paths towards prefixes 1 and 2 were passing through AS6939 (Hurricane Electric). In the next day, however, paths are more dispersed; some pass through AS3549 (Global Crossing), some through AS3257, some only through AS174 (Cogent), and some pass only through other ASes, not in the figure, which connect directly to AS29632. In other words, ASes are switching from AS6939 to other ways to reach AS29632. We can conclude that either (a) AS6939

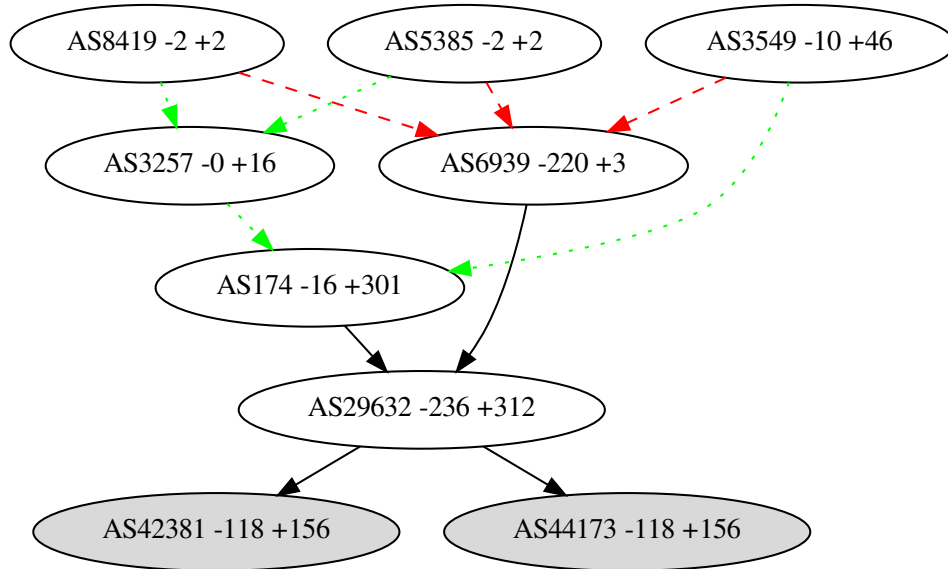


Figure 5·2: Summary of path changes in the network towards two prefixes (hosted at gray nodes) from Apr-30-2013 to May-01-2013.

made its routes to the subject prefixes unattractive or unavailable, or (b) AS29632, as well the others mentioned above, took actions to make their routes more attractive or available.

This is interesting as a single event, as it shows a large-scale reorganization of the network with regard to two prefixes hosted by different ASes. However, it becomes even more interesting when we note that the same event (or its reverse) happened 28 times during 2013. From these 28 days, PathMiner identified AS6969 and AS29632 as responsible by the event in 18 and 10 days respectively.

To illustrate how PathMiner finds such event, consider Figures 5·3a and 5·3b. These plots show all points in time during 2013 (on the x axis) where each AS (on the y axis) changes its next-hop towards one prefix. We treat each plot as a binary matrix, in which element (j, k) is 1 if the AS represented by row j changes it next-hop towards the subject prefix between the days k and $k + 1$.¹ It is clear that each matrix consists of noise plus a strong signal; that signal is extracted and shown in Figures 5·3c and 5·3d. Because Figures 5·3c and 5·3d are very similar, they together represent

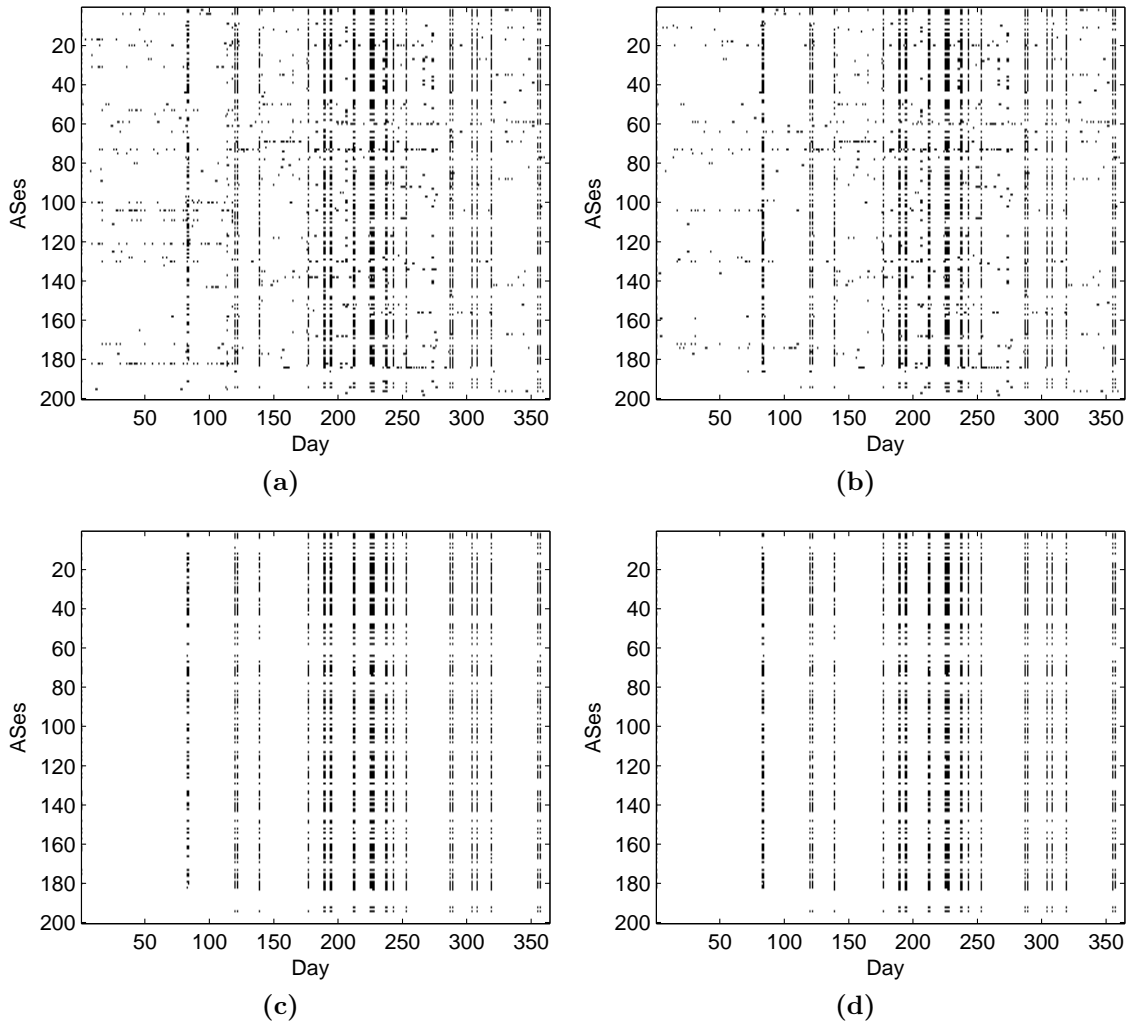


Figure 5.3: Next-hop changes towards the two prefixes of Figure 5.2 during 2013. An observed next-hop change, by AS j , towards the subject prefix at day k implies value 1, black dot, for the element (j, k) . Subfigures are: (a) prefix 1, all changes; (b) Prefix 2, all changes; (c) Prefix 1, signal; and (d) Prefix 2, signal.

the fact that a group of many ASes changed their next-hops towards both prefixes synchronously, multiple times in 2013.

This simple example shows the nature of the kinds of events captured by PathMiner. It also illustrates the key challenges that PathMiner must overcome: *i)* how can we extract signal from noise (e.g., going from Figure 5.3a to 5.3c)? *ii)* next, how can we find prefixes with similar signal matrices (e.g., matching Figure 5.3c with 5.3d)? And *iii)* once multi-AS/prefix/time events are extracted, how can we identify the AS or link most likely to have triggered all the routing changes captured in the event (as shown in Figure 5.2)? In the following sections, we present the solutions taken by PathMiner to these challenges.

5.2 Notation and definitions

In this section, we present the mathematical notation/definitions (Section 5.2.1) used throughout the chapter and a formal definition of the problem we aim at solving (Section 5.2.2).

5.2.1 Notation

In this chapter, scalars will be denoted by lower-case letters (a), sets by upper-case letters (A), vectors by lower-case bold-face letters (\mathbf{v}), and matrices by upper-case bold-face letters (\mathbf{M}). We will also work extensively with 3-dimensional arrays, or *tensors*,¹ which we denote by upper-case calligraphic letters (\mathcal{T}).

A tensor may be seen as a collection of slices (matrices), fibers (vectors) or elements. More specifically, for the n -by- m -by- l tensor \mathcal{X} , we use $\mathbf{X}_{i::}$, $\mathbf{X}_{:j}$ and $\mathbf{X}_{::k}$ respectively to denote horizontal, lateral and frontal slices. In the same way, $\mathbf{x}_{:jk}$, $\mathbf{x}_{i:k}$

¹A change is related to the observed next-hops at two instants of data collection, the first on day k , and the second on day $k + 1$. Transient changes, between the two data collection points, are not considered in this work. Details in Sections 5.2.2 and 5.3.

¹The word tensor is in general used to refer to N -dimensional arrays. In this chapter, we deal only with $N = 3$. More details about tensors can be found in the work of Cichocki et al. (2009).

and \mathbf{x}_{ij} : denote column, row and tube fibers respectively. Finally, x_{ijk} (and with same meaning $x_{i,j,k}$, \mathcal{X}_{ijk} or $\mathcal{X}_{i,j,k}$ depending on convenience) denotes the element (i, j, k) of \mathcal{X} . In all cases above, $i = 1, \dots, n$, $j = 1, \dots, m$ and $k = 1, \dots, l$.

Tensor \mathcal{Y} is an induced tensor from \mathcal{X} if there exist sets $A = \{a_1, \dots, a_{n'}\}$, $B = \{b_1, \dots, b_{m'}\}$ and $C = \{c_1, \dots, c_{l'}\}$ such that $y_{ijk} = x_{a_i, b_j, c_k}$. In this case, we say that sets A , B and C induce \mathcal{Y} in \mathcal{X} and we write $\mathcal{Y} = \mathcal{X}(A, B, C)$. The same meaning holds if we use vectors instead of sets (with the provision that when using vectors, the ordering of indices matters, which is not the case when using sets).

Unless otherwise stated, operations over tensors are defined analogously as operations over matrices. Specifically, for an n -by- m -by- l tensor \mathcal{X} , we denote its size as (n, m, l) , its volume as $vol(\mathcal{X}) = n \times m \times l$, and its Frobenius norm as $\|\mathcal{X}\| = \sqrt{\sum_{i,j,k} x_{ijk}^2}$. We will also frequently refer to the *density* of a tensor, which is the fraction of its entries that are nonzero. In the particular case of binary tensors, this is $den(\mathcal{X}) = \frac{\sum_{i,j,k} x_{ijk}}{vol(\mathcal{X})}$.

5.2.2 Problem definition

Our starting point is the path-based nature of BGP (Border Gateway Protocol), in which ASes keep information about the preferred paths to each reachable prefix. From a perspective of ASes and prefixes, at a given time t the global state of the system can be defined as: a set of ASes A , a set of prefixes P , and for each $a \in A$ a set of AS-paths, each of which allows a to reach a prefix in P . Another representation that we will also use is based on next-hops. In this representation, the state of the system consists of a set of tuples (a, b, p) where $a \in A$ uses $b \in A$ as the next-hop to reach $p \in P$. The next-hop representation of the system contains less information than the preferred-paths representation, so they are not equivalent, but each will be more convenient for certain parts of PathMiner.

To formalize the next-hop representation, at a given time t let \mathbf{N} be a multivalued

n -by- m matrix of next-hops in the network, where \mathbf{N}_{ij} denotes the set of next-hops used by AS j to reach prefix i . Observing \mathbf{N} over a discrete set of l points in time yields an n -by- m -by- l multivalued tensor \mathcal{N} , where \mathcal{N}_{ijk} is the set of next-hops used by AS j to reach prefix i at time k . Tensor \mathcal{N} represents the complete dynamics of the network over the set of measurements.

By comparing (frontal) slices of \mathcal{N} , we can identify next-hop changes in the network. This results in a binary tensor \mathcal{C} , which is the n -by- m -by- $(l - 1)$ tensor of *routing changes*, defined as:

$$\mathcal{C}_{ijk} = \begin{cases} 1, & \text{if } \mathcal{N}_{ijk} \neq \mathcal{N}_{i,j,k+1}, \\ 0, & \text{otherwise.} \end{cases} \quad (5.1)$$

Given these definitions, we can define a *high impact event* in the global routing system as: sets I (of prefixes), J (of ASes) and K (of points in time) such that the sub-tensor $\mathcal{C}(I, J, K)$ has large volume and high density. Because $\mathcal{C}(I, J, K)$ has large volume, it has potential for high impact – many routing changes might be involved. The fact that $\mathcal{C}(I, J, K)$ has high density means that it is likely to be a singular event – that is, most ASes are changing their next-hops toward most prefixes at most timepoints, and as argued above, such unusually coordinated activity is likely due to the actions of a single network element.

To make this definition concrete, we introduce the concept of a (λ, ν) -event:

Definition 3. ((λ, ν) -event) A binary tensor \mathcal{B} is a (λ, ν) -event with regard to binary tensor \mathcal{X} if there exist sets I , J and K such that $\mathcal{B} = \mathcal{X}(I, J, K)$; $\text{den}(\mathcal{B}) \geq \lambda$; and $\text{vol}(\mathcal{B}) \geq \nu$.

Definition 3 still is not enough to fully characterize the events we are seeking. For instance, two distinct (λ, ν) -events \mathcal{B} and \mathcal{B}' may be such that \mathcal{B} is a sub-tensor of \mathcal{B}' and hence, the former can be viewed just as redundant information when compared

with the latter. Therefore, it is also necessary to put constraints on the set of (λ, ν) -events we want to find. Thus, the final description of our problem, which we call Boolean Tensor (λ, ν) -Factorization (or (λ, ν) -BTF) is:

Problem 1. (*Boolean Tensor (λ, ν) -Factorization*) Given a binary tensor \mathcal{X} , integers r and ν and a real λ , the Boolean Tensor (λ, ν) -Factorization problem consists of finding r triples of sets (I_h, J_h, K_h) , $h = 1, \dots, r$ such that:

- i)* $\mathcal{X}(I_h, J_h, K_h)$ is a (λ, ν) -event in \mathcal{X} , for $h = 1, \dots, r$; and
- ii)* $\left\| \mathcal{X} - \bigvee_{h=1}^r \mathcal{X}^{(h)} \right\|$ is minimized, where: $\mathcal{X}^{(h)}$ is a binary tensor with same size as \mathcal{X} , and $x_{ijk}^{(h)} = 1$ iff $(i, j, k) \in I_h \times J_h \times K_h$; and $\bigvee_{h=1}^r \mathcal{X}^{(h)}$ is defined as the elementwise logical or of $\mathcal{X}^{(1)}, \dots, \mathcal{X}^{(r)}$.

The intuition behind the definition of Problem 1 is to find a set of r binary tensors that best approximate \mathcal{X} as blocks of 1's. Moreover, we are only interested in tensors related to (λ, ν) -events. More formally, $\bigvee_{h=1}^r \mathcal{X}^{(h)}$ is the best approximation for \mathcal{X} that can be obtained with a rank- r binary tensor, when each $\mathcal{X}(I_h, J_h, K_h)$ is a (λ, ν) -event.

Unfortunately, easy ways to solve Problem 1 exactly are not known. Setting $\lambda = 0$ and $\nu = 0$, the problem is equivalent to the Boolean Tensor Factorization problem studied by Erdős and Miettinen (2013), which is known to be NP-hard (Miettinen, 2011). Therefore, the more general problem of (λ, ν) -BTF is also NP-hard.

In summary, this section sets out a definition that translates the general notion of *identifying high impact events in the global routing* into a specific problem. Unfortunately, this definition highlights two difficulties: *i)* complete topologies of the global interdomain routing system over time are not available; and *ii)* computing an exact solution to Problem 1 is hard in general. In the next section we discuss the available data we used and how we address its incompleteness, and then in Section 5.4, we describe our heuristic algorithms for finding solutions to Problem 1.

It is important to emphasize that in this work we did not aim at a general approach for solving instances of the BTF problem coming from an arbitrary application. In fact, among recent works trying to solve the general BTF problem are the ones by Cerf et al. (2009) and Erdős and Miettinen (2013). The former is designed to find blocks of closed relations (i.e., equivalent to blocks with density 1), which is too strong a requirement for our application. The latter would suit our needs, but although it does not impose any formal requirement for volume and density, unfortunately, the available implementation did not scale for our datasets due to the amount of data analyzed by PathMiner.

5.3 Dataset description

In this section, we present the dataset we used, how we processed it, and its limitations. Our source of data consisted of BGP RIBs (Routing Information Bases) made available by RIPE Routing Information Services¹ and Route Views.²

We obtained data from 9 years, from the beginning of 2005 until the end of 2013, at a daily timescale. For each day and repository, we obtained the RIB made available at 8am (or, if not available, the closest one). We made the arbitrary choice of 8am in order to have approximately 24 hours between routing information collected for each day. From the RIBs, we extracted all records of the form *route dumping date, prefix*, and *AS-path* (only data related to IPv4 prefixes). We stored this data on a 12-node cluster in a Hadoop Distributed File System (HDFS) and, for the most data intensive computations, we used Hadoop³ and Spark.⁴ Table 5.1 presents a summary of our dataset.

As described in Section 5.2, our first step is to obtain the next-hop tensor \mathcal{N} .

¹<http://www.ripe.net/data-tools/stats/ris/ris-raw-data>

²<http://www.routeviews.org>

³<https://hadoop.apache.org>

⁴<https://spark.apache.org>

Table 5.1: Summary of dataset.

Year	Prefixes	ASes	Size (GB)
2005	286723	23157	340
2006	331421	26188	470
2007	400784	29382	640
2008	438730	32929	806
2009	506978	36336	845
2010	543868	39339	939
2011	626312	43151	1154
2012	879730	46262	1397
2013	850997	49502	1745

To this end, for each entry of the form [date, prefix p , AS-path], where AS-path is given by $[AS_1, \dots, AS_q]$, we computed the $q - 1$ 4-tuples [date, p , AS_i , AS_{i+1}], for $i = 1, \dots, q - 1$. Each 4-tuple means that at time *date*, in order to reach prefix p , the source AS_i uses as next-hop AS_{i+1} .

Referring to Table 5.1, it is important to remark that we do not have routing information from every source to every destination at every point in time. In this context, missing data can arise mainly for two reasons: (a) data collection issues and (b) visibility problems (the RIPE and Route Views RIBs do not capture the complete AS-topology of the Internet). Note as well that the datasets considered are large – for recent years, over 1TB each in size.

Fortunately, both of these problems (missing data, and dataset size) can be significantly lessened by carefully selecting a representative subset of the data. Accordingly, we selected our data subset by greedily choosing ASes and prefixes with most of the next-hop changes in the network. To that end, using big data tools, we computed the tensor \mathcal{C} ($\mathcal{C}_{ijk} = 1$ iff AS j changed next-hop towards prefix i from day k to $k + 1$) for each of the nine datasets in full. Next, we computed the total number of changes over the year for each prefix i (as a destination), each AS j (as a source) and for each pair (i, j) . Figures 5.4a and 5.4b present the log-log complementary CDFs of the total number of changes for prefixes and ASes, respectively. It can be seen that for ASes there is a distinct subset of heavy-hitters that account for the majority of

routing changes.

Going further, Figure 5.4c shows the cumulative number of changes for a pair (i, j) for the year of 2011. That is, value at row i and column j of the heat map represents the fraction of changes corresponding to the i prefixes and j ASes with most changes in the dataset. This figure provides information about how to greedily, with respect to the number of routing changes, obtain a sample of ASes and prefixes. With regard to ASes, it is possible to see that including more than the top 200 ASes in the sample does not increase significantly the fraction of changes captured.

On the other hand, there are hundreds of thousands of prefixes that experience significant levels of routing changes. At the same time, it is also important to recognize that there exist many sets of prefixes for which routing information is essentially redundant. In particular, in many cases prefixes originated by the same AS are routed similarly (Broido and claffy, 2001). For our problem, such sets of prefixes add no additional information about coordinated routing changes.

Hence, in our data subset we chose the top 20000 prefixes in terms of volume of changes, but only allowing at most one prefix hosted in each AS.¹ Since the 20000 prefixes come from 20000 distinct ASes, and there are only about 50000 active ASes in total, we expect that this subset captures a large fraction of the distinct and observable routing changes in our data. That is, the intuition behind our sampling strategy was to obtain a set of prefixes responsible for many changes, and at the same time avoiding the discovery of blocks with many prefixes belonging to the same AS and hence having identical routing changes.

Table 5.2 presents statistics about tensor \mathcal{C} projected only over the samples; each sample tensor is of size 20000-by-200-by-364. Note that because the set of active ASes and prefixes has changed over the 9-year timespan, the sampling procedure was done independently for each year. Therefore, the sets of prefixes and ASes considered vary

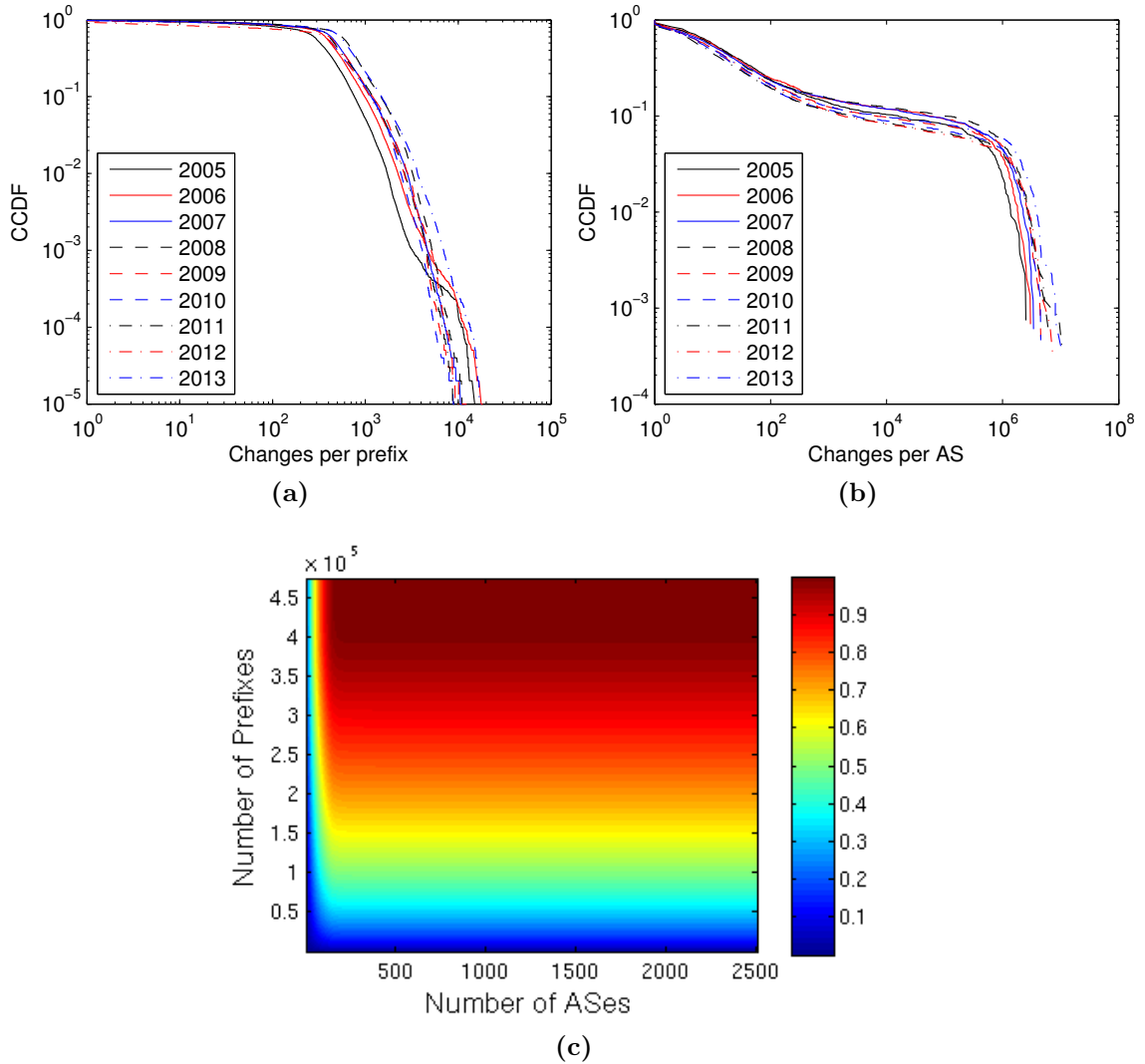


Figure 5-4: Distributions of the total number of next-hop changes: (a) per prefix; (b) per AS; and (c) per pair (AS, prefix), for 2011 only. In (c) value at (i, j) indicates the total number of next-hop changes from the first j ASes towards first i prefixes (ASes and prefixes are sorted, in decreasing order, by their number of changes).

Table 5.2: Summary of the sampled tensors of routing changes.

Year	Density (%)	Missing (%)
2005	0.9	7.4
2006	1.2	8.7
2007	1.5	6.7
2008	1.9	6.3
2009	1.7	8.1
2010	1.6	7.0
2011	1.7	5.9
2012	1.8	7.4
2013	2.3	5.5

from year to year, in order to capture the maximum amount of routing changes, and hence results in the remaining of this work were computed independently for each sample.

Table 5.2 also shows the percent of missing data, i.e., missing next-hop entries in the tensor \mathcal{N} (projected over the samples). Since these percentages were reasonably small, we handled missing data as follows. Suppose that at some point in time k we have $\mathcal{N}_{ijk} \neq \emptyset$ for some i, j and $\mathcal{N}_{i,j,k+1} = \emptyset$. Because data at time $k + 1$ is missing, we cannot know for certain whether: *i*) AS j cannot reach prefix i at time $k + 1$ (which would imply that $\mathcal{C}_{ijk} = 1$), or *ii*) AS j can reach i at time $k + 1$ but our data set does not contain that information (which implies that \mathcal{C}_{ijk} can be either 0 or 1). In light of this limitation, we took a conservative approach and defined \mathcal{C}_{ijk} to be 0. Therefore, a 1 in the tensor \mathcal{C} means that we definitely observe a next-hop change in the data, but a 0 can either imply that a change did not happen, or that we simply do not know what happened. Choosing to set these entries to 0 also reflects that fact that zeros comprise the vast majority of known values in our data.

Note that the amount of uncertainty in the routing changes tensors (\mathcal{C}) is thus bounded by the 5 to 10% of missing data. This has implications in the next section for our event extraction. In particular, we should expect that even when a set of ASes all change their next-hops for a set of prefixes, our data may only show a portion (e.g.,

¹We considered the *host* of a prefix to be the AS advertising the prefix for the first time in each year.

90%) of those changes – i.e., the observed density of a valid event may be less than 100%.

Another aspect worth mentioning about the dataset is the presence of more than one next-hop from some ASes towards some prefixes. In fact, that was the motivation for our definition of next-hop change, given in Equation 5.1, where one can see that we compare sets, instead of elements. Other possible approach would be to use a finer granularity of the network, where the issue of more than one next-hop never arises. For instance, such granularity could be obtained by considering quasi-routers. Unfortunately, the use of quasi-routers would introduce problems hard to be addressed in our analyses. For example, which quasi-router observed at a day k corresponds to the quasi-routers observed at day $k + 1$?

In order to provide means to understand how much the issue of more than one next-hop may affect our results, we computed the fraction of cases in which $|\mathcal{N}_{ijk}| > 1$ or $|\mathcal{N}_{i,j,k+1}| > 1$, during the computation of \mathcal{C}_{ijk} , in our nine datasets (before sampling). In all cases this, fraction was below 8%. Furthermore, from the cases where $\mathcal{C}_{ijk} = 1$ we have that the same fraction is below 30% (in all datasets).

5.4 Extracting Events

As previously discussed, at a high-level PathMiner consists of two main steps: (1) finding high-impact events, and (2) identifying the most likely network element causing the event. In this section, we describe our solution to step (1), which is an algorithm for finding good solutions to the (λ, ν) -BTF problem.

An attractive solution to the (λ, ν) -BTF problem would be to start from an existing algorithm for standard Binary Tensor Factorization. Unfortunately, the large size (combined with the density) of our binary tensors was too much for existing algorithms to handle. As a result, we developed a scalable heuristic to find (λ, ν) -events

in the routing changes tensor \mathcal{C} .

Our approach has two steps. First, we look at individual slices of \mathcal{C} and extract (λ, ν) -events within slices (Section 5.4.1). Then, we aggregate similar slice events in order to form events that span multiple slices (Section 5.4.2).

The algorithmic approach we adopt is well suited to the case where significant large, dense events exist in the data (as those presented in Figure 5-3). Thus, it is well suited to the current problem, while other algorithms (including the work of Erdős and Miettinen (2013)) may be more appropriate when events are smaller and rarer. For the sake of brevity and due to space limitations we present only a high-level description of our algorithms.

5.4.1 2D Factorization

Let \mathbf{X} be a slice of \mathcal{C} . It can be a prefix (horizontal), AS (lateral) or a time (frontal) slice. For example, \mathbf{X} may be the prefix slice shown in Figure 5-3a. The goal of 2D Factorization is to extract from \mathbf{X} a set of large volume, high-density patterns such as shown in Figure 5-3c. For simplicity of notation, we assume from now on that \mathbf{X} is a prefix slice of \mathcal{C} for prefix i . Description for AS and time slices proceeds analogously.

To start, we define a rank-1 binary matrix: given binary vectors \mathbf{a} and \mathbf{b} , a rank-1 binary matrix \mathbf{X} is given by $\mathbf{X}_{ij} = \mathbf{a}_i \times \mathbf{b}_j$. That is, a rank-1 binary matrix is the outer product of two binary vectors and can be thought of as a ‘block’ of 1s (after reordering of rows and columns).

The 2D Factorization step repeats the following as long as a significant fraction of 1s can be obtained from \mathbf{X} : Find a good rank-1 binary approximation for \mathbf{X} , denoted \mathbf{M} . If the set of rows J (representing ASes) and columns K (representing days) with non-zero elements of \mathbf{M} induces a submatrix \mathbf{B} with volume at least ν and density at least λ , then label the triple $(\{i\}, J, K)$ as a (λ, ν) -event. Next, independently of volume and density of \mathbf{B} , remove all ones captured by the set of rows J and the set

Algorithm 1: 2D-Factorization

Data: n by m binary matrix \mathbf{X} related to prefix i , λ , ν and convergence parameter ϵ

```

1  $F \leftarrow \{\}$ 
2  $\mathbf{Z} \leftarrow \mathbf{X}$ 
3 repeat
4    $\mathbf{Z}' \leftarrow \mathbf{Z}$ 
5    $\mathbf{M} \leftarrow \text{Rank-1-Approximation}(\mathbf{X}, \mathbf{Z}, \lambda, \nu)$ 
6    $J \leftarrow \{j : m_{jk} = 1\}$ 
7    $K \leftarrow \{k : m_{jk} = 1\}$ 
8    $\mathbf{B} \leftarrow \mathbf{X}(J, K)$ 
9   if  $\text{den}(\mathbf{B}) \geq \lambda$  and  $\text{vol}(\mathbf{B}) \geq \nu$  then
10     $F \leftarrow F \cup \{(\{i\}, J, K)\}$ 
11    for  $(j, k) \in J \times K$  do
12       $z_{jk} \leftarrow 0$ 
13 until  $\|\mathbf{Z}\| = 0$  or  $\frac{\|\mathbf{Z}-\mathbf{Z}'\|^2}{\|\mathbf{Z}\|^2} < \epsilon$ ;
14 return  $F$ 

```

of columns K . This strategy is described more precisely in Algorithm 1.

The key challenge is to obtain the rank-1 approximation \mathbf{M} (Line 5). In fact, this is equivalent to the Frequent Itemset Mining problem, which currently has not a known easy way to be exactly solved. Furthermore, the tensor \mathcal{C} may have many thousands of slices, so the algorithm we use must run quite quickly. Hence, we use the strategy of *relaxing* the discrete problem into a continuous one, solving the continuous problem, and thresholding the result to find an approximate discrete solution.

Our relaxation seeks a real-valued rank-1 approximation to \mathbf{Z} (which initially is a copy of \mathbf{X}). For this, we use Non-Negative Matrix Factorization (NNMF) (Berry et al., 2007). Using NNMF we find real nonnegative vectors \mathbf{w} (n -by-1) and \mathbf{h} (1-by- m) such that \mathbf{wh} is a real nonnegative rank-1 matrix approximating \mathbf{Z} . We then threshold \mathbf{w} and \mathbf{h} independently, obtaining the binary vectors \mathbf{w}' and \mathbf{h}' , such that $\mathbf{M} = \mathbf{w}'\mathbf{h}'$ minimizes $\|\mathbf{X} - \mathbf{M}\|$ (see that the error is computed considering the original matrix). We note that once \mathbf{w} and \mathbf{h} are computed, computing \mathbf{M} (by finding the

optimal thresholding) can be performed in time $O(mn)$.

For the results in this chapter, we ran Algorithm 1 for each of the 9 datasets with $\lambda = 0.7$, $\nu = 100$ and $\epsilon = 1\%$. We considered as input three different sets of slices: prefix slices, AS slices and time slices. Figure 5-5 summarize the results by presenting a scatterplot of density *versus* volume for the year of 2013 (same general comments apply for other datasets). While prefix slices give us events with a wide range of density and volume, AS and time slices behave differently. Basically, most of the AS and time events have low density or volume close to 10^4 . The explanation is twofold: first, AS and time slices are harder to mine, since their size is significantly larger than the size of prefix slices; and second, many of the events on AS and time slices consist of one AS changing its next-hop towards all (or almost all) prefixes in the network. Hence, because prefix slices offer the best quality results (in terms of density) and reveal the most interesting structure in the data, we use only prefix slices to generate the (λ, ν) -events used in the next stage of the algorithm.

5.4.2 3D Factorization

The next step is to find 3 dimensional (λ, ν) -events in the tensor \mathcal{C} . An input S , obtained using the method in the last section, is a set of triples of the form $(\{i\}, J, K)$, where i represents a prefix, J a set of ASes and K a set of points in time. The basic idea is to find triples having similar blocks (given by sets J and K) and to group them to create 3D events.

We consider that there may be two ways of combining a pair of blocks. First, they may be nearly identical – that is, their intersection may be nearly as large as their union. In that case, we merge them by constructing the block that contains them both. Second, their intersection may be much smaller than their union, but still sufficiently large in terms of absolute volume. In that case, we merge them by constructing the block that is their intersection.

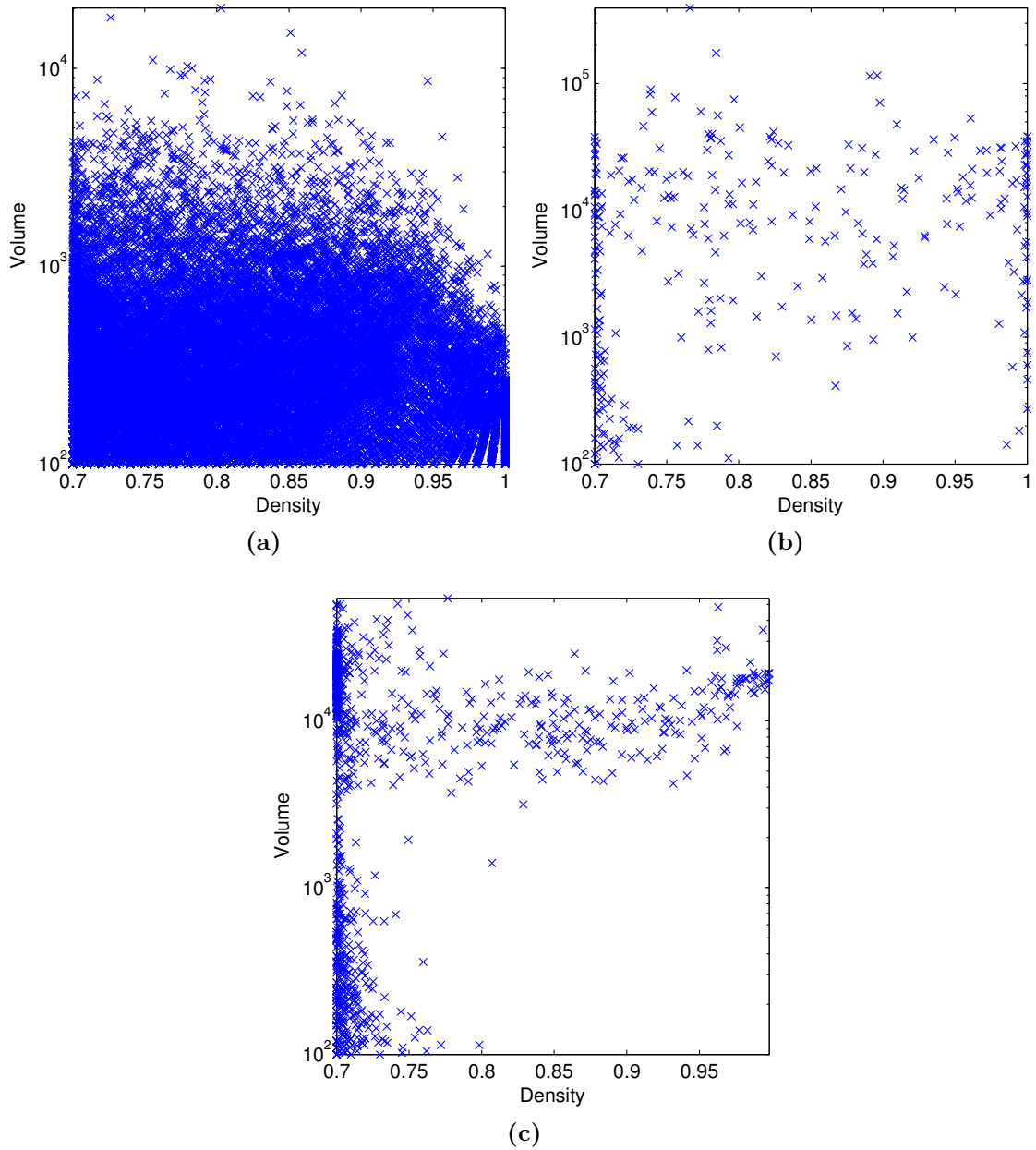


Figure 5.5: Density *versus* Volume for 2D (λ, ν) -events obtained from slices of tensor \mathcal{C} , 2013 only. Prefix (horizontal) slices in (a), AS (lateral) slices in (b), and time (frontal) slices in (c). Experiments performed using Algorithm 1 with $\lambda = 0.7$, $\nu = 100$ and $\epsilon = 1\%$.

To evaluate these two cases, we define two functions over pairs of triples $x = (I, J, K)$ and $x' = (I', J', K')$ as:

$$d_B(x, x') = 1 - \frac{|J \cap J'| \times |K \cap K'|}{|J \cup J'| \times |K \cup K'|} \quad (5.2)$$

and

$$s_B(x, x') = |J \cap J'| \times |K \cap K'|. \quad (5.3)$$

The distance function d_B measures the volume of the intersection of two blocks divided by the volume of their union. If close to zero, then J is similar to J' and K is similar to K' . Hence, it is natural to assume that $(I \cup I', J \cup J', K \cup K')$ is a larger event. We extend this strategy to merge multiple triples in a single step as follows: given a triple x , look for a set S' (which contains x) such that $\max_{y, z \in S'} d_B(y, z) \leq \gamma$, for some $0 \leq \gamma \leq 1$. Once S' is found we combine all of its elements at once using the following operator:

$$\text{COMBINE-UNION}(S') = \left(\bigcup_{y \in S'} y_1, \bigcup_{y \in S'} y_2, \bigcup_{y \in S'} y_3 \right),$$

where y_1 , y_2 and y_3 represent, for triple y , the sets of prefixes, ASes and points in time respectively.

The similarity function s_B captures the case when the intersection of two blocks is large enough by itself to merit merging the blocks. That is, it may be the case that triple x is not nearly the same as any other triple in S , but there still exists x' such that $s_B(x, x')$ is significantly large (larger than some threshold β). In this case, we may conclude that $(I \cup I', J \cap J', K \cap K')$ is an event. In fact, the second part of the algorithm looks for elements $x' \in S$ such that $s_B(x, x') \geq \beta$ and then combines them using the following operator:

$$\text{COMBINE-INTER}(x, x') = (I \cup I', J \cap J', K \cap K').$$

Algorithm 2: Event-Selection

Data: Tensor \mathcal{C} , S , set of triples of sets of the form (I, J, K) and thresholds λ and ν

- 1 $L \leftarrow$ sort triples $(I, J, K) \in S$ in decreasing order of volume ($|I| \times |J| \times |K|$)
- 2 $S' \leftarrow \emptyset$
- 3 **for** $i = 1$ **to** $|L|$ **do**
- 4 $(I, J, K) \leftarrow L_i$
- 5 $\mathcal{B} \leftarrow \mathcal{C}(I, J, K)$
- 6 **if** $vol(\mathcal{B}) \geq \nu$ **and** $den(\mathcal{B}) \geq \lambda$ **then**
- 7 **foreach** $(i, j, k) \in I \times J \times K$ **do**
- 8 $\mathcal{C}_{ijk} \leftarrow 0$
- 9 $S' \leftarrow S' \cup \{(I, J, K)\}$
- 10 **return** S'

Alternating the test for maximum distance and minimum similarity and using COMBINE-UNION and COMBINE-INTER operators iteratively yields the discovery of new triples (possibly representing new (λ, ν) -events) and is the core of our strategy to find 3-dimensional blocks in \mathcal{C} . The procedure is summarized in Algorithm 4 (presented in Appendix A).

One may note that Algorithm 4 does not check the density of new formed 3D blocks. This is because extracting the relevant portions of \mathcal{C} for this check is so time-consuming as to be prohibitively expensive for our datasets. Of course, this absence of verification may lead to triples that induce blocks with low density. A second problem that may arise is that the extracted 3D blocks may still show significant overlap. In order to address these two issues, we added a final stage that discards blocks with low density and blocks that do not add new information to the results because it overlaps with many others. This process is described in Algorithm 2, which greedily (by decreasing order of volume) selects only triples with a minimum density and that captures significant information not contained in blocks previously selected. Results and parameters set up related to the methodology presented in this section will be

Table 5.3: Summary of (λ, ν) -events found by PathMiner.

Dataset	#Events	#1's Retrieved	Percent
2005	5255	1107109	8.2
2006	6823	1689299	9.5
2007	8252	2504558	11.1
2008	7996	2411041	8.3
2009	8602	2466807	9.7
2010	9646	2952688	12.6
2011	12042	3991264	16.3
2012	13910	4611049	17.8
2013	13992	5880885	17.7

discussed in Section 5.5.

5.5 Characterizing Events

After the execution of the set of algorithms presented in the previous section, we have a collection of 3D events extracted from the routing changes tensor \mathcal{C} , each one with large volume and high density. In this section, we briefly pause to characterize the events found by PathMiner in the 9 years of routing data.

After experimentation, we settled on the following parameters for the algorithms presented in Section 5.4.2: $\lambda = 0.8$, $\nu = 100$, $\gamma = 0.1$ and $\beta = 100$.¹ Table 5.3 summarizes the overall performance of PathMiner when using these parameter settings on our data. The column ‘# 1’s Retrieved’ is the total number of routing changes that were contained in events, and the ‘Percent’ column is the fraction of all routing changes in our data that were contained in events.

The table shows that PathMiner is able to find many blocks, comprising a significant fraction of the routing changes contained in the datasets, ranging from 8.2% in 2005 up to 17.8% in 2012. Figures 5-6a, 5-6b and 5-6c present the same statistics on a daily basis as a time series over the 9 years of data. Note that the samples for each year are distinct so trends should not be inferred across years. From Figures 5-6b and

¹For instance, using $\beta = 200$ and $\gamma \in \{0.2, 0.05\}$ did not change the results significantly in terms of percentage of retrieved next-hop changes, average volume or average density. Tests performed with 2013 dataset.

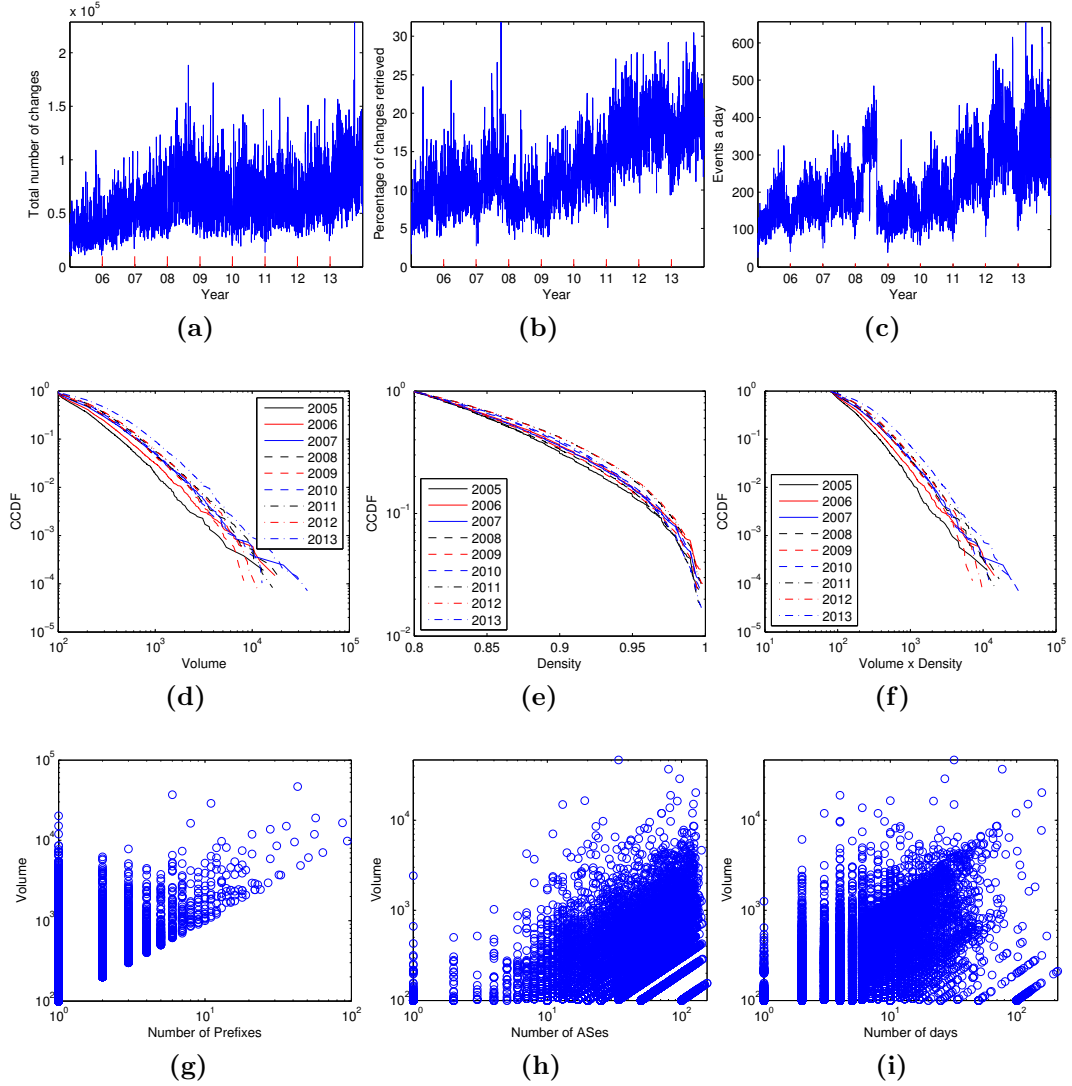


Figure 5.6: Basic statistics about (λ, ν) -events found by PathMiner ($\lambda = 0.8$ and $\nu = 100$). Time series of: (a) total number of next-hop changes in each yearly sample; (b) percentage of these changes captured in events by PathMiner; (c) number of events found in each day. Complementary CDFs of: volume of events – volume is defined as the product of the number of prefixes, ASes and days in an event; (e) density of events – density is the number of changes captured by an event over its volume; (f) product volume \times density of events. Event scatterplots (2013 only): (g) number of prefixes *versus* volume; (h) number of ASes *versus* volume; (i) number of days *versus* volume.

5-6c we can see that PathMiner performs better in tensors with higher density and that overall, the shape of the time series related to the fraction of changes and the number of events a day is similar to the shape of the time series related to the total number of changes (higher values in 2008 and in the last three years, 2011, 2012 and 2013).

Next, we move to analyzing density and volume of events. As an algorithmic constraint, we have that each block has density higher than 0.8 and volume greater than 100. But, in general, how large and dense are the events? Figures 5-6d, 5-6e and 5-6f present the log-log complementary CDF for volume, density and their product, respectively, for all 9 datasets. Generally, the distributions are long-tailed, with many small events and a few very large events. In terms of volume, we can see that for 2005 more than 10% of all blocks have volume greater than 500 and that this number goes up to 1000 in 2013. In the tail of the curves, we can see some huge events with volume greater than 10^4 . Similar comments can be made for Figure 5-6f. In terms of density, although 0.8 can be considered a high threshold, there are many blocks that are much denser. For instance, approximately 30% of all events (in all datasets) have density higher than 0.9, and 10% are higher than 0.95 in density.

We also explore the relationship between the volume of events and the number of prefixes, ASes and days that the event contains. Figures 5-6g, 5-6h and 5-6i present the results for the year of 2013 (results for other years follow the same trend). It can be seen that the extracted events include large numbers of days and ASes (obtained via the 2-dimensional factorization algorithm) and that events with many prefixes are formed as a result of the 2D event aggregation step.

Finally, Table 5.4 presents the 5 largest events, in terms of volume for each year. Columns 2 to 4 represent the number of prefixes, ASes and days in each event (thus the volume of the event can be obtained by their product). The table shows that

Table 5.4: Description of top-5 events of each year.

Dataset	#Prefixes	#ASes	#Days	Density
2005	2	54	200	0.90
	2	60	109	0.88
	1	59	138	0.83
	1	57	95	0.87
	2	52	48	0.91
2006	3	68	168	0.86
	82	11	19	0.83
	1	66	183	0.86
	3	58	61	0.91
	1	65	156	0.85
2007	140	49	20	0.80
	13	52	44	0.81
	75	8	35	0.89
	77	5	28	0.80
	45	37	6	0.91
2008	156	28	25	0.88
	79	41	6	0.85
	15	44	23	0.83
	22	37	13	0.83
	10	103	10	0.88
2009	45	102	2	0.80
	1	90	91	0.87
	13	51	11	0.81
	6	89	13	0.80
	14	48	10	0.83
2010	49	38	16	0.90
	28	41	11	0.97
	60	101	2	0.88
	98	15	8	0.94
	17	43	13	0.80
2011	81	23	16	0.80
	37	23	19	0.90
	21	43	14	0.88
	46	53	5	0.90
	47	20	11	0.95
2012	1	87	152	0.82
	4	65	44	0.86
	1	98	101	0.90
	1	94	96	0.90
	31	36	8	0.87
2013	43	34	32	0.82
	6	80	77	0.82
	11	92	27	0.81
	1	128	158	0.80
	57	83	4	0.86

PathMiner was able to find events of remarkable scope, some of which involve dozens of ASes, prefixes, and days. The largest events (in 2007 and 2008) involve over 100,000 individual routing changes. In this respect, it is important to recall as well that each prefix in our datasets is originated in a distinct AS, so the actual number of routing changes per event in the full data is much larger because of the similar routing behavior of co-originated prefixes.

5.6 Single Actor Analysis

In this section, we present the second component of PathMiner, an algorithm to identify the network element that is most likely to have caused a large event, given that the ASes, prefixes and days involved in the event are known. We start by explaining our identification methodology, and then we present results.

5.6.1 Algorithm

We start with some basic observations. Consider an action taken by some network element a (say, an AS) that causes a path from AS b towards prefix p to change. We observe that a is, in general, either on the path from b to p before the change, or after (Feldmann et al., 2004). For example, if a link fails or comes up, or an AS announces a new route and/or withdraws an existing route, these events can cause changes to many paths, but the paths involved will all pass through the link or AS either before or after the change.¹

Thus, if we are interested in some event that happened on day k , then we may want to compare the set of paths seen in the network on day k with those seen on day $k+1$. Counting paths in order to identify (or narrow down) a root cause for a routing change has been explored before (Lad et al., 2006). However, PathMiner differs from

¹The assumption that the cause of the event will be in the new or old path is not always true (Javed et al., 2013). However, PathMiner is built under such assumption because it deals with large-scale events, where the likelihood of observing paths passing through the cause increases.

such previous work in terms of goals (since it particularly focuses on large events that may re-occur over time) and in methods (since it carefully chooses the set of paths to analyze).

PathMiner takes advantage of the fact that starting from a collection of (λ, ν) -events, found by its first component, is an effective way of isolating the set of paths to study, thus avoiding interference of paths that are changing for unrelated reasons. Specifically, when analyzing an event that on day k has set of ASes J and set of prefixes I , we consider only paths seen on days k and $k + 1$ passing through ASes in J towards prefixes in I . If one of such path start at an AS which is not in J and passes through $a \in J$, then PathMiner takes such path in account, but it ignores the portion before a . Accordingly, we define P_k to be the multiset of all paths that are found starting at an element of J and ending at an element of I on day k .

Considering the above observations we have that a good candidate for the single network element responsible for the entire set of observed path changes has the following properties: on either day k or day $k + 1$, (a) most of the changed paths pass through the network element and (b) most the paths going through the network element change. We call these rules *single actor rules* and the network element so identified the *single actor*.

For concreteness, assume that network element e is the single actor, and the changed paths go through e on day k but not on day $k + 1$. Among all the paths being considered, define $D = P_k \setminus P_{k+1}$ as the paths that disappeared; D_e as the paths in D passing through e ; and $P_{k,e}$ as the paths in P_k that pass through e . If e is the single actor, by the single actor rules one concludes that $r_e^D = \frac{|D_e|}{|D|}$ should be close to 1, and $p_e^D = \frac{|D_e|}{|P_{k,e}|}$ should be close to 1. In other words, r_e^D captures the fraction of disappeared paths passing through e (rule (a)) and p_e^D captures the fraction of paths passing through e that disappeared (rule (b)).

For intuition, we note that p_e^D and r_e^D correspond respectively to the definitions of *precision* and *recall* for a particular classifier. This classifier is the one that declares that a path will disappear if the path passes through e on day k . In other words, maximizing p_e^D and r_e^D with regard to e yields the network element that best classifies the paths as changing versus unchanging. Following common practice in machine learning, we combine precision and recall into the F-score. In fact, in our case, we use the F_2 score defined by $F_2^D(e) = \frac{5p_e^D r_e^D}{4p_e^D + r_e^D}$ to place higher emphasis on the role of recall.

We can make analogous definitions covering the set of paths that appear on day $k+1$. In that case, we define $A = P_{k+1} \setminus P_k$, and A_e as the paths in A that pass through e . Hence, $p_e^A = \frac{|A_e|}{|P_{k+1,e}|}$, $r_e^A = \frac{|A_e|}{|A|}$, $F_2^A(e) = \frac{5p_e^A r_e^A}{4p_e^A + r_e^A}$ and the same interpretation given above is valid here as well.

In practice, one does not know a priori whether e will be found using the single actor rules applied to disappeared paths on day k , or appearing paths on day $k+1$. To overcome this problem we define the candidate single actor to be the network element (or set of elements) that maximizes $\Delta F(e) = \max\{F_2^D(e), F_2^A(e)\}$. From this analysis, we exclude AS-links that are seen on days k and $k+1$ and ASes which all incoming and outgoing AS-links are also seen on days k and $k+1$. This decision is justified by the fact that an AS-link that has not been disrupted and an AS that, explicitly, has not changed its local preference and/or export policies are unlikely the element that triggers a large-scale event.

This algorithm is effective, but can some times return multiple elements as candidate single actors for any given day k in an event. However, one can bring one more observation to bear: the single actor should be the same over all days k in the event. This offers an additional opportunity to winnow the set of candidate single actors. So the final step of the single actor analysis is: for events that re-occur over a set of days

K , repeat the single-day strategy for every $k \in K$. Then, define as the final single actor of the event the element (or elements) that is observed in the candidate single actors set in at least the majority of the days in K . If such element is not found (no element is a candidate single actor in the majority of days), the algorithm declares that it was not able to identify the cause of the event.

5.6.2 Performance

PathMiner is able to identify the actors responsible for most of the events presented in Table 5.4. Table 5.5 shows the ASes identified as the single actor for each event, and the number of days that the element was classified as single actor (note that in the table we present only the most frequent one and that other elements, present in the majority of days, also have to be considered as possible actors). The boldface numbers indicate the cases where PathMiner identified the same element(s) as cause(s) for a majority of days in the event. It can be seen that PathMiner has more difficulty identifying single actors for events spanning many days and/or involving few ASes.

In order to validate the actors identified by PathMiner for events in Table 5.4 we performed a visual inspection of each event as follows⁴: for each day of each event we looked at the network element identified as actor, the graphical representation of the event, and asked two questions. First, “would an action of that network element (e.g. an AS changing its local preferences and/or its export policies) explain the occurrence of the event?” Second, “Do the actions of this network element provide the simplest explanation among all elements involved in the event?” If the answers for those question were affirmative in the majority of days of the event we considered such event validated.

We also conducted analyses to understand how often PathMiner is able to identify

⁴Graphical representations for these events and actor identification summary are available at <http://cs-people.bu.edu/gcom/bgp/imc2014>. We also make available the same representation and summary for another 500 randomly selected events for the year of 2013.

Table 5.5: Single actor analysis of Top-5 events of each year.

Dataset	Most Frequent Actor	Days
2005	31050	76
	23918	38
	1299	38
	8342	57
	20485	30*
2006	23918	50
	3257	13
	33697	103
	20485	58
	33697	93
2007	174	14
	1273	18
	4637	35
	3257	16*
	7575	6
2008	9121	16*
	25462	3
	174	12
	25462	5
	3303	8
2009	3216	2*
	15412	62
	3216	7
	8359	10
	29049	5
2010	30890	14
	3491	11
	21219	2
	5588	8*
	13249	8
2011	5588	16
	3549	17
	3491	13
	12989	4*
	174	10
2012	38312	142
	29632	37
	4755	96
	56209	91
	4651	8
2013	7713	28
	12880	52
	8529	19
	10029	116
	21219	4

* The actor found by PathMiner provides a simple explanation for the event. However, other elements providing a simple explanation could be identified by visual inspection.

a single actor as cause of an event. Our first analysis is related to the maximum value of $\Delta F(e)$ obtained for each day in an event and for all events that PathMiner found in the 9 datasets. The results, presented in Figure 5.7a by a CDF, indicate that large values are the predominant case. For instance, the figure shows that the maximum $\Delta F(e)$ is equal to 1 in 40% of the cases and is greater than 0.8 in more than 80%. These numbers suggest that in most of the days within events PathMiner is capable of finding a network element (or a set of) that, in fact, can explain the massive amount of changes related to the event.

Although promising, these results only refer to individual days within events. It is important to also ask how consistent the identification is over the days of an event, and how often PathMiner is able to find the same actor over the set of days of the event. To answer those questions we identified for each event the element that appears as cause of the event most frequently over the days of the event and the fraction of days this single actor has been identified. Figure 5.7b presents the results. It can be seen (with exception of 2005) that in more than 40% of the events a single network element has been observed by PathMiner as an event cause over all days of the event. Furthermore, we can see that in more than 90% of the events there is a single actor that is present over the majority of days. For the cases where the majority has not been found we refer to Figure 5.7c to show that in some of them the event contains many days, which naturally increases the complexity of the problem. However, there are many small (in terms of days) events for which PathMiner was not able to identify a single actor. Initial investigation reveals that those events are related to few ASes, and as consequence to few paths, which is not an ideal situation to work with measures such as precision and recall. We emphasize that further investigation is necessary in that direction as future work.

As a final analysis, we list the ASes which, as actors had the greatest impact on

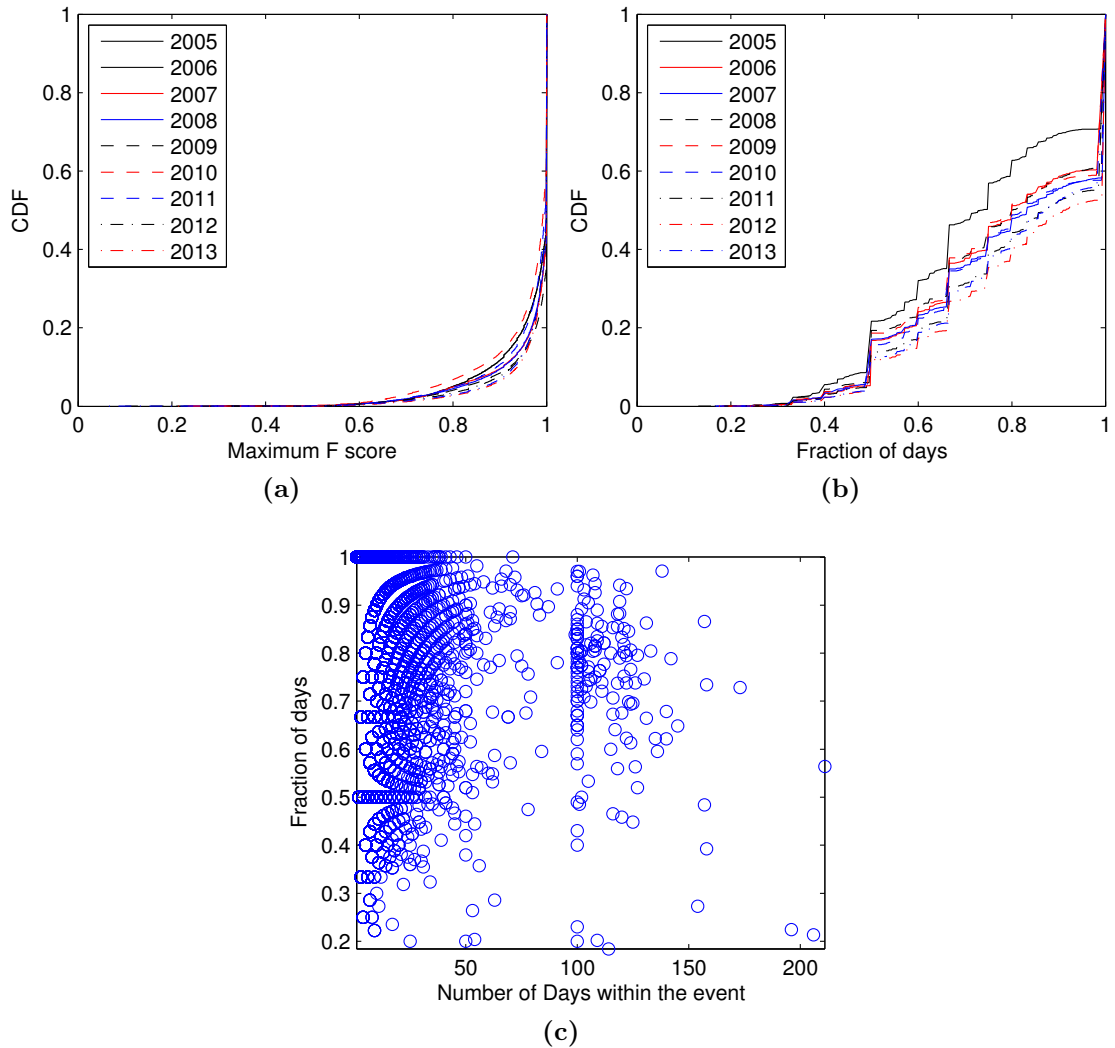


Figure 5.7: Summary of our methodology for Single Actor identification: (a) CDF of ΔF , computed over each day of each event; (b) CDF of the fraction of days that the same AS (or AS-link) is reported as actor (has maximum ΔF score) over all days of the event; (c) scatterplot of the number of days within an event *versus* the fraction of days that the same AS (AS-link) is reported as actor – same variable as in (b).

Table 5.6: Three different rankings for the Top-20 Single Actors

#Events	Total volume	Total days
AS174	AS9498	AS9498
AS9498	AS4755	AS4755
AS9002	AS6453	AS6453
AS3356	AS12880	AS174
AS6939	AS6939	AS3549
AS3549	AS9002	AS3216
AS12389	AS174	AS6939
AS3216	AS3549	AS3356
AS20485	AS3216	AS20485
AS6453	AS12389	AS15412
AS31133	AS15412	AS12389
AS7018	AS20485	AS9002
AS701	AS3356	AS701
AS4755	AS3491	AS3491
AS12880	AS55410	AS18101
AS8167	AS10029	AS8167
AS6461	AS12989	AS12880
AS209	AS4651	AS7018
AS3491	AS197556	AS18881
AS8359	AS8167	AS55410

the network. We considered the year of 2013, for all events that PathMiner was able to identify a single actor and that involved at least 50 ASes (to avoid the problem of few paths mentioned previously). Table 5.6 shows the top-20 ASes ranked in three different ways: first, by the number of events that the AS is an actor; second, by the aggregate volume of all events that the AS is an actor; and third, by the aggregate number of days of all events that the AS is an actor. In all three cases, if more than one AS was identified as actor, or if there was an AS-link, we counted each AS involved individually. Therefore, events with more than one AS as actor are counted more than once in Table 5.6.

First, it can be seen that the way we ranked the ASes change the order in the three columns of the table, but there is a significant overlap between the top-20 of each rank, showing that, in general, there are heavy-hitter actors that trigger many large, reoccurring events. Second, in all three ranks, it is possible to identify some large ASes, which peer other large ASes and are important parts of the Internet core.

For example, AS174 (Cogent), AS6939 (Hurricane Electric), AS3549 (GBLX) and AS3356 (Level3) are in the top-20 of each list. This suggests that some ASes in the core are responsible for a large amount of the reorganization that happens in the network. However, it is important to remember that our strategy to sample ASes may have influence in this result, and ultimately that our original dataset does not cover the complete AS-level topology of the Internet (which in fact motivated our sampling strategy).

5.6.3 Case studies

In this section, we present three case studies in order to show the ability of PathMiner to identify actors of high impact events. Case studies I and II were chosen to illustrate typical scenarios where PathMiner is able to identify a network element (or set of elements) whose actions would explain, in a simple way, the occurrence of the event. We remark that many events share the same structure (this can be seen in the online supplementary material). Case study III, on the other hand, shows that PathMiner is not guaranteed to always work. In that case, we discuss the reason and show that the assumptions under which PathMiner was built are not satisfied.

In the representation of each case study, Figures 5·8, 5·9 and 5·10, ASes in gray hosts a destination prefix. Red (dashed) edges are those present only at day k , green (dotted) edges are present only at day $k + 1$ and black (solid) edges are present in both days. Inside each node we have: the AS identifier; the number of paths passing through the AS at day k , with a negative sign; and the number of paths passing through the AS at day $k + 1$, with a positive sign. Only paths passing through an AS in the event towards a prefix in the event are counted.

Case study I

Our first case study is related to the second event for the year of 2009 presented in Table 5.4. In this event, we have 90 ASes changing their next-hops towards one prefix over 91 days during the year. Among those, we picked 2 pairs of consecutive days in order to illustrate the nature of the event. Following the same convention used in Section 5.1, Figures 5-8a and 5-8b picture the most important part of the network for our needs. From Feb-08 to Feb-09 many ASes stopped using AS15412 as next-hop and started using AS4637 instead. Just from that observation, it is possible to conjecture that one of these two ASes was responsible for the changes, and in fact, considering Feb-08 and Feb-09, both ASes have the same ΔF value. On the next day, almost the reverse event happens, with ASes leaving AS4637 and starting to use AS15412 as next-hop. But it can be seen that AS4637 still has a path towards the prefix and it still is used by some ASes as next-hop. This fact indicates that AS4637 is probably not the actor responsible for the event.

It is also necessary to observe that AS15412 and AS18101 are not seen in our dataset at Feb-09 and Feb-10 (no paths passing through them). Hence, we cannot say exactly if the cause is one or the other or even a link leaving one of them, narrowing down the set of candidates to AS15412, AS18101 and AS10029 (and links). However, on other days of the event, the same structure of the figures was observed, but there were paths passing through AS15412 and AS18101. This indicates that in fact many paths left (preferred) AS15412 as next-hop due to its actions, not AS18101 or AS10029.

Following the majority rule, PathMiner selected AS15412 as the single actor of the event and therefore, it was able to capture the discussion presented above.

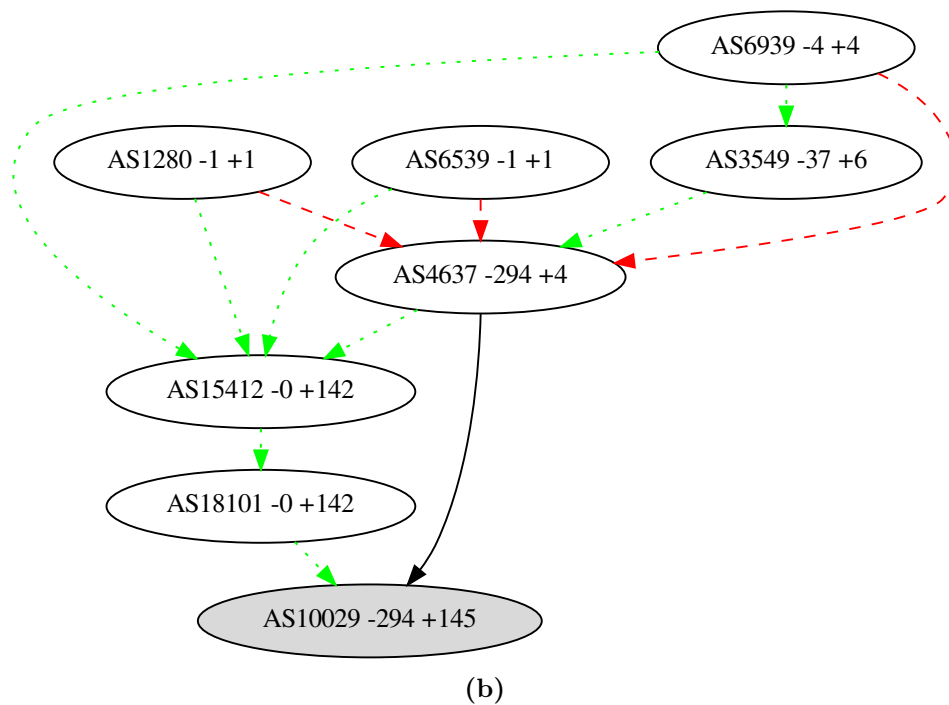
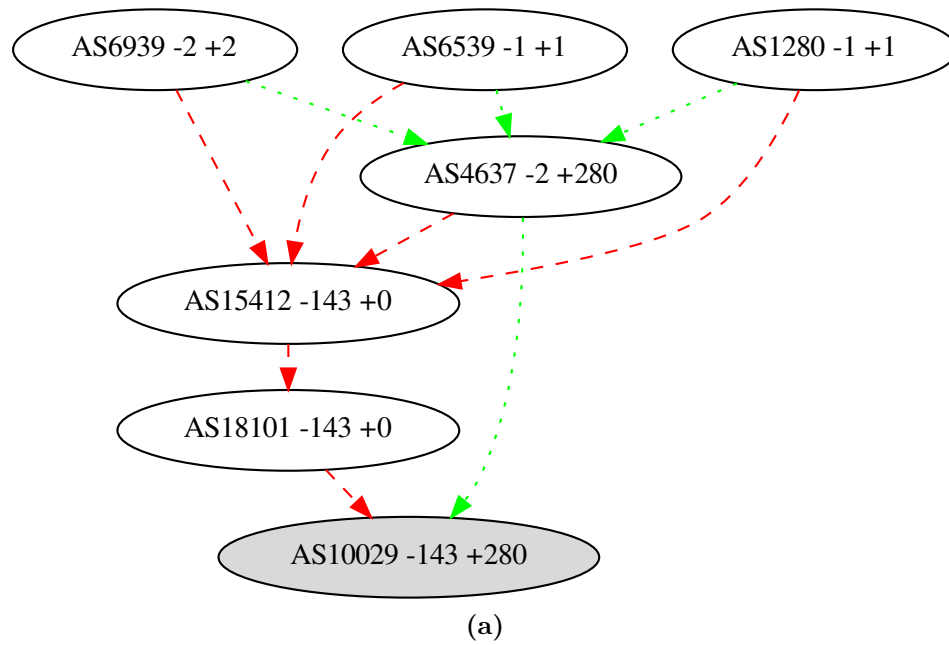


Figure 5-8: Representation of case study I: (a) path changes between Feb-08-2009 and Feb-09-2009; and (b) path changes between Feb-10-2009 and Feb-11-2009

Case study II

In this case study, we explore the second event for the year of 2012 shown in Table 5.4. The event involves 4 prefixes, 65 ASes, and occurs on 44 days during the year. Figures 5.9a and 5.9b present the simplified dynamics of the changes for 2 days, Jan-10-2012 and Jan-11-2012. This case study seems similar to the previous one, since we can observe that over the time ASes are alternating their next-hops (towards the four prefixes) between AS35320 and AS29632. Simply looking at those figures it is possible to narrow down the cause to one of these two ASes (or the links between them and AS42418). The question that arises is: why did PathMiner choose AS29632? It turns out that for many of the 44 days of the event, some of ASes leaving next-hop AS29632 started reaching two of the four destinations by paths not passing through AS35320. Therefore, either AS29632 or two other ASes (at the same time) are the causes. PathMiner identifies AS29632 as actor, capturing the idea that the simplest explanation is the most likely one.

Case study III

Our last case study discusses an event for which PathMiner was not able to identify a single network element responsible for the whole event. The event is the one on the first row of Table 5.4 for the year of 2005. The event involves 2 prefixes, 54 ASes and it happens on 200 days of the year. However, PathMiner was not able to find a single network element responsible for changes in at least 100 days. Why did that happen? A closer look at the event shows us that the assumptions over which PathMiner is built are not valid. In fact, this is a strange case. Figures 5.10a and 5.10b present two typical subgraphs describing the changes over the days of the event. First, we note that the structure of the graph is completely different from our two previous case studies. The second, and key, fact is that the event contains only two prefixes, but

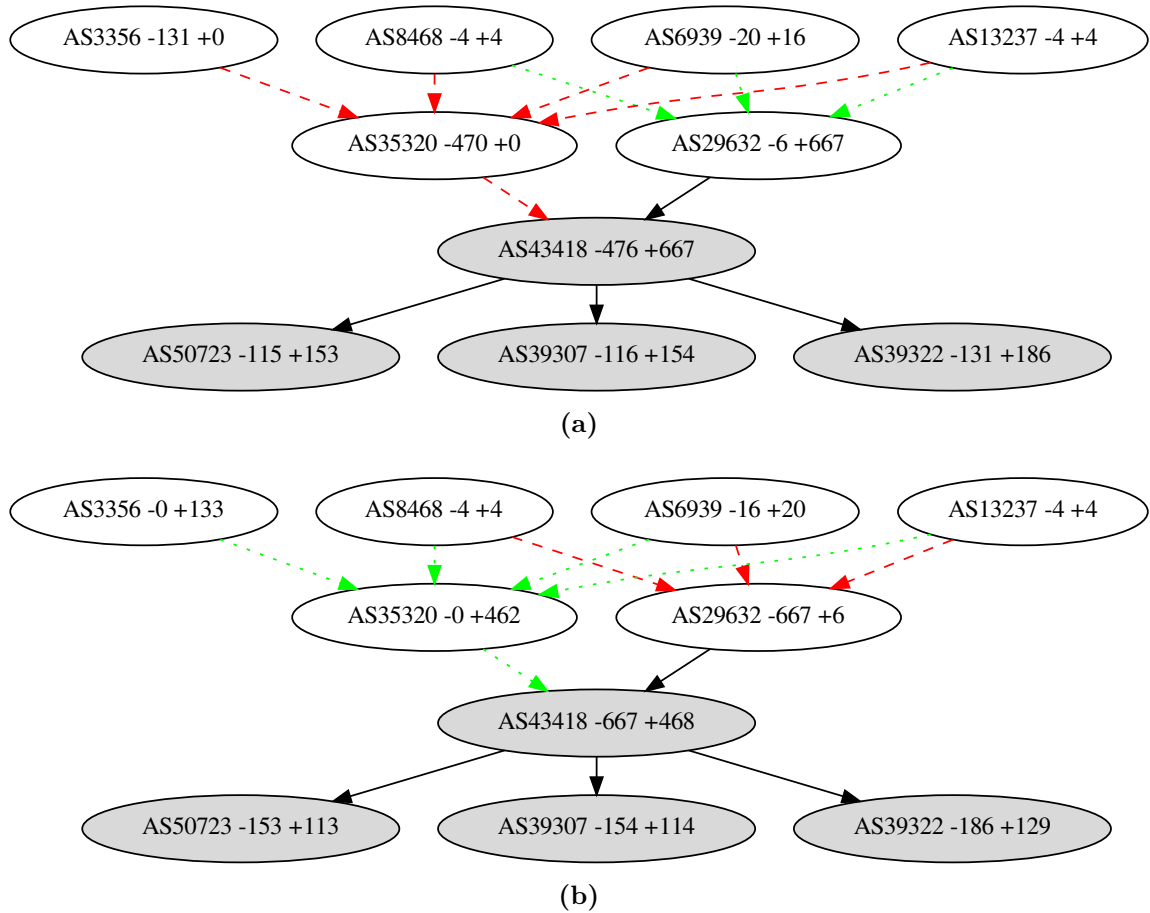
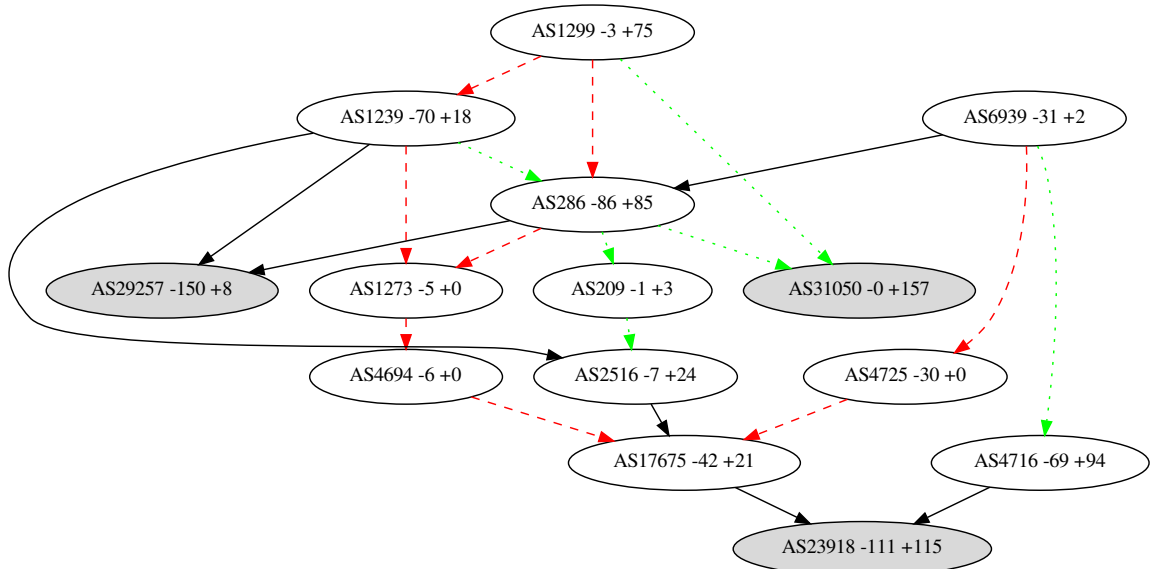
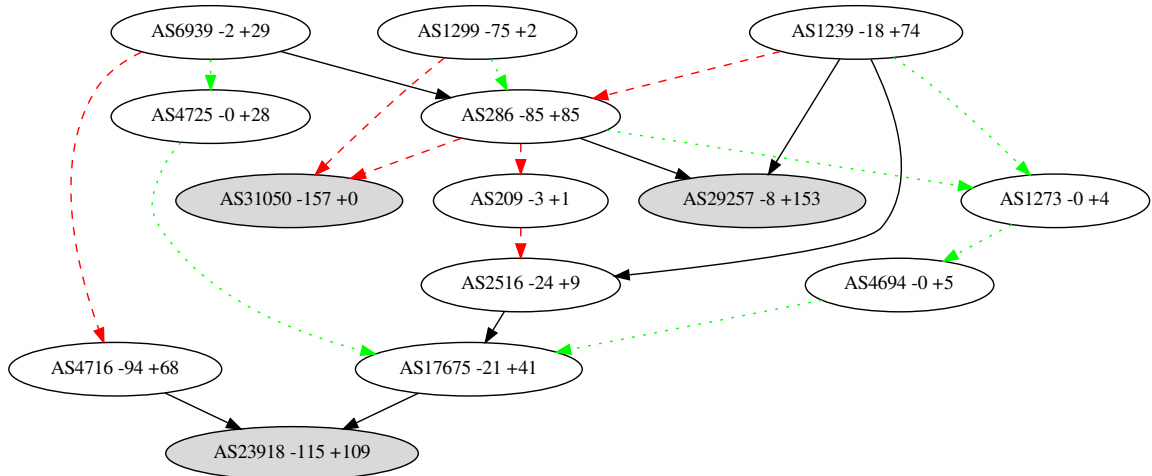


Figure 5-9: Representation of case study II: (a) path changes between Jan-10-2012 and Jan-11-2012; and (b) Path changes between Jan-11-2012 and Jan-12-2012.



(a)



(b)

Figure 5·10: Representation of case study III: (a) Path changes between Jan-11-2005 and Jan-12-2005; and (b) Path changes between Jan-12-2005 and Jan-13-2005.

we can see three gray nodes in the figures (gray nodes are those hosting the prefixes of the event). Denoting those two prefixes by prefix 1 and prefix 2, through data examination it was possible to see that: *i*) at Jan-11 prefixes 1 and 2 were being hosted at AS23918 and AS29257 respectively; *ii*) at Jan-12 prefix 1 was being hosted at AS29257 and at AS31050 (which does not seem to be a normal situation) and prefix 2 was being hosted at AS23918; and *iii*) at Jan-13 prefixes are hosted as at Jan-11.

Over the 200 days of the event, this alternating state was repeated. In summary, it can be seen that is hard to find one network element that can be responsible for these changes. More specifically, our assumption of a single actor causing the event does not seem to be valid here – it appears that a set of coordinated changes is being implemented through actions of multiple ASes.

5.7 Summary of the chapter

In this chapter we presented PathMiner, a system capable of identifying and analyzing high impact events in the interdomain routing system of the Internet. We started by giving a formal definition of the problem and discussing how to prepare the data and how to deal with missing data. In its first phase, PathMiner has the advantage of working with a next-hop representation of the AS-level Internet, which allows PathMiner to naturally combine paths obtained from multiple sources.

In order to identify events, we proposed a new heuristic to solve the Boolean Tensor Factorization problem, since algorithms currently available in the literature were not scalable to our datasets. Using datasets spanning 9 years of routing in the Internet we showed that PathMiner is able to find many large and dense events.

Next, we addressed the challenge of discovering network elements (ASes or AS-links) that were likely the actors responsible for the event occurrence. Our method-

ology, based on concepts borrowed from classification theory, was able to identify possible event causes in most of the events we found. One key aspect of PathMiner is its ability to look at the same event re-occurring many times, which helps the process eliminating event cause candidates. Finally, we presented 3 case studies, for large events, exemplifying all these aspects of PathMiner.

Chapter 6

Detecting Unusually-Routed ASes

The routes used in the Internet’s interdomain routing system are a rich information source that could be exploited to answer a wide range of questions. However, analyzing routes is difficult, because the fundamental object of study is a set of paths. In this chapter, we present new analysis tools – metrics and methods – for analyzing AS paths, and apply them to study interdomain routing in the Internet over a recent 13-year period. Our goal is to develop a quantitative understanding of changes in Internet routing at the micro level (of individual ASes) as well as at the macro level (of the set of all ASes). To that end, we equip an existing metric (Routing State Distance) with a new set of tools for identifying and characterizing unusually-routed ASes. At the micro level, we use our tools to identify clusters of ASes that have the most unusual routing at each time (interestingly, such clusters often correspond to sets of jointly-owned ASes). We also show that analysis of individual ASes can expose business and engineering strategies of the organizations owning the ASes. These strategies are often related to content delivery or service replication. At the macro level, we show that ASes with the most unusual routing define discernible and interpretable phases of the Internet’s evolution. Furthermore, we show that our tools can be used to provide a quantitative measure of the ‘flattening’ of the Internet.

The remaining of this chapter is organized as follows: Section 6.1 shows notation and definitions. Section 6.2 presents the main dataset we used. Section 6.3 formalizes the concept of unusually routed ASes, and describe how to find them. Section 6.4

presents tools for explaining unusual routing strategies. Section 6.5 describes how the set of unusually-routed ASes changes over time, and how such changes correlate with the evolution of the Internet. Section 6.6 shows how the framework described in this chapter can be used to quantify the flattening of the Internet. In Section 6.7 we discuss the robustness of our results with regard to missing data. Finally, Section 6.8 contains the summary of the chapter.

6.1 Definitions

In this section, we present the basic definitions that constitute the basis for the tools and methods developed in the next sections. To that end, we borrow RSD (Routing State Distance) from the work of Gürsun et al. (2012b), and we equip such concept for cases where next-hops from a source to a destination are not unique. Such modification enables the study of a wider class of path-based networks, at different granularities and over time. Therefore, a contribution of the present work is to show how RSD can be used as part of a larger tool set to take on a new class of problems.

6.1.1 Path-based networks

The set of paths used in interdomain routing can be abstracted as a *path-based network*. We define a path-based network G as a pair (V, P) , where V is a set of nodes and P is a function that, for $a, b \in V$, maps (a, b) to a set of paths $P(a, b)$. Each element $p \in P(a, b)$ is a sequence of nodes from V , forming a directed path starting at a and finishing at b . If $p = a \rightarrow v_1 \rightarrow \dots \rightarrow v_p \rightarrow b$, we say that the *next-hop* of a towards b is v_1 ; and if $p = a$ we say that a is the next-hop to itself. Hence, the set $P(a, b)$ entails a multiset of next-hops $N(a, b)$, representing the diversity of local decisions made at a in order to reach b . Figure 6.1 presents an example of path-based network.

Structurally, a path-based network is more complex than a network. A network

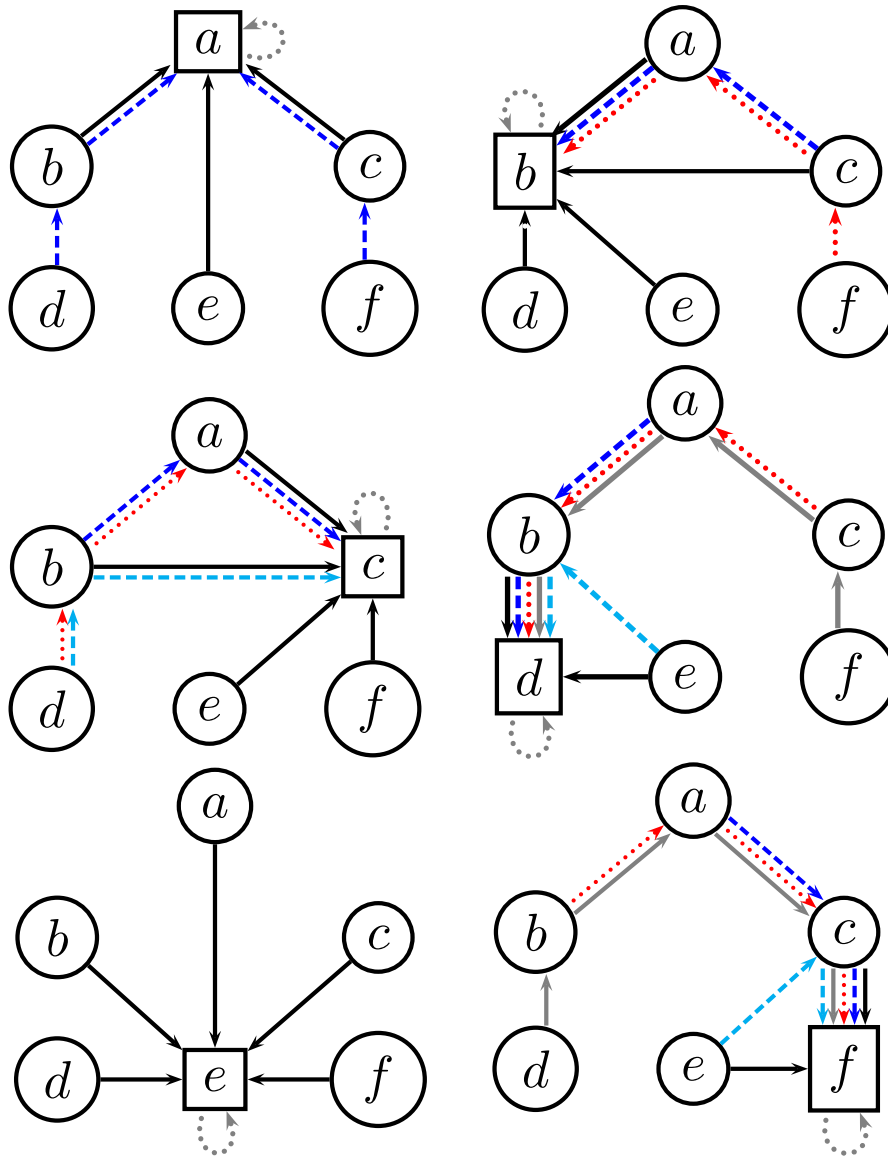


Figure 6-1: A path-based network over nodes $V = \{a, b, c, d, e, f\}$, broken out to separately show the paths towards each square node. A path is a sequence of directed edges with same line style. Examples of path sets are: $P(f, d) = \{fcbad\}$, $P(b, b) = \{b\}$, and $P(d, c) = \{dbac, dbc\}$. Those path sets entail respectively the following next-hop multisets: $N(f, d) = \{c\}$, $N(b, b) = \{b\}$, and $N(d, c) = \{b, b\}$.

is a two-way relationship among nodes, and may be encoded in a set of pairs (v_1, v_2) , denoting that vertices v_1 and v_2 are connected by an edge. In contrast, a path-based network is a three-way relationship among nodes; a set of paths may be encoded in a set of triples (v_1, v_2, v_3) stating that v_3 is a next-hop taken from v_1 on the path to v_2 . For instance, in Figure 6-1, b has a direct connection to reach node c , but b cannot use this connection to reach f .

A path-based network is a particularly suitable tool for modeling the interdomain routing system, because the set of paths used is hard to describe concisely, and because data is constrained to flow only over specific paths. In this context, we consider that the set of nodes is formed by ASes and IP prefixes (as destinations), and the path sets are formed by the AS-paths leading to the AS originating the prefixes. In this case, the next-hop representation takes into account the fact that the same AS can choose different next-hops toward the same prefix. At the AS granularity, we model ASes as nodes, and the next-hops as the multiset union over the next-hops toward each prefix originated at the AS. Thus, if a and b are ASes, $N(a, b)$ characterizes the diversity of local decisions of a in order to reach b . In other words, it naturally considers that ASes are not atomic structures, and that by a variety of reasons, there can be different paths from a source to a destination (Mühlbauer et al., 2007, 2006).

A limitation of our modeling approach (driven by characteristics of currently available datasets) is that it labels prefixes associated with caches deployed at an access network as originated by the access network, even though the addresses may be related to services offered by other companies. This limitation and its impact on our results are subjects of future work.

6.1.2 RSD

Additional background to the methods we use in this chapter is a metric called *routing state distance* (RSD), which is defined for a path-based network. RSD conceptually

measures the dissimilarity of two nodes in terms of *how they are reached* in a path-based network.

Definition 4. Given $Z \subseteq V$, the Routing State Distance between two network nodes x and y is defined as

$$\text{RSD}(x, y) = \frac{1}{|Z|} \sum_{z \in Z} d(N(z, x), N(z, y)), \quad (6.1)$$

where d is a dissimilarity measure over multisets that assumes values between 0 and 1.

RSD values range between 0 and 1, and if d satisfies the triangle inequality, so does RSD. For d we use the Generalized Jaccard Distance (GJD) between the vectors of frequencies of each next-hop in $N(z, x)$ and $N(z, y)$ normalized by the number of prefixes originated at x and y respectively. GJD has been studied and used in many contexts and is a metric (Chierichetti et al., 2010). For two real and non-negative n -dimensional vectors u and w (given that u and w are not simultaneously null vectors), GJD is defined as follows:

$$\text{GJD}(u, w) = 1 - \frac{\sum_{i=1}^n \min(u_i, w_i)}{\sum_{i=1}^n \max(u_i, w_i)}. \quad (6.2)$$

Taking Figure 6.1 as example, for destinations a and b , the sources d and f completely agree on their next-hops choices toward both destinations; sources a , b and e completely disagree; and source c partially agrees (with $d(N(c, a), N(c, b)) = 0.5$). Therefore, $\text{RSD}(a, b) \approx 0.58$.

Unfortunately, in practice, it is not always the case that we can have access to full visibility of the network. Hence, it is not possible to obtain $Z = V$ in order to compute RSD. In our work, we choose Z such that $N(z, x) \neq \emptyset$ for all $z \in Z$ and $x \in V$; we call Z the set of *sources*. One way of describing $\text{RSD}(x, y)$ is that it approximates the *probability that an arbitrary node in Z uses different next-hops on*

the paths to x and y .

Finally, although in this work we use GJD as the distance function d in Equation (4), it is possible to use different distances for different applications. For instance, one can incorporate weights related to other sources of information (e.g., traffic volume and prefix popularity), if it is available.

6.2 Data Preparation

In this section, we describe the data used in our analysis and how we adapt it to measure RSD consistently. We obtain Internet AS paths from two sources: the RIPE Routing Information Service (RIS)¹ and the Route Views² projects. We assess the potential impact of missing data (especially AS peering links) in Section 6.7.

Data acquisition: From each project, we obtain all available RIBs (Routing Information Base) for the first three days of each month from January 2003 to December 2015. Each entry in each RIB provides one AS-path record of the form $[t; AS_1, \dots, AS_s; p]$, for day t , AS-path AS_1, \dots, AS_s , and IP prefix p (IPv4 only). The choice of the three first days is arbitrary, and we expect results to hold if one consistently uses any three consecutive days of the month. Considering much longer periods of time can lead to complications when trying to obtain a good approximation of the state of the system (for a given time), thus introducing noise in the results.

Next-hop determination: Each AS-path record provides $(s - 1)$ next-hop records of the form $[t, AS_i, AS_{i+1}, p, AS_s]$, $i = 1, \dots, s - 1$. Such record means that at day t , AS_i uses AS_{i+1} as next-hop towards prefix p , which is originated at AS_s . All tuples with $AS_i = AS_{i+1}$ are filtered out. This processing allows us to cover cases where a prefix is originated at more than one AS.

Instability filtering: For each month, we remove next-hop records $[t, x, y, p, o]$ for

¹www.ripe.net/data-tools/stats/ris/

²www.routeviews.org

which (x, y, p, o) appears in only one of the three snapshots of the month. This lowers the influence of short-term routing changes on our data. Each day t is converted to its corresponding month, and resulting duplicate records are removed. We refer to the portion of the resulting dataset for any given day as a single *snapshot*. There are 156 snapshots covering the 13 years of our study.

Prefix assignment: Each prefix is assigned to its originating AS. Thus, all next-hop records are converted from $[t, x, y, p, o]$ to $[t, x, y, o]$, (i.e., we remove the destination prefix, but keep the destination AS that originates it). Next, all next-hops having the same time, source and destination are aggregated. This yields records of the form $[t, x, N(x, o), o]$, where $N(x, o)$ represents the multiset of next-hops from x towards o . These records define the next-hop function $N(\cdot, \cdot)$ used in Equation (6.1). Observe that next-hops are still constrained to whole ASes. It is possible to consider finer granularities, as in the work of Gürsun et al. (2012b), but as Comarela et al. (2013) argue, such approach introduces several complications for temporal analyses.

Selection of sources and destinations: Consistent computation of RSD requires that we know the value of $N(x, o)$ for all x and o over some domain. Accordingly, we heuristically looked for large AS sets Z and V , denoting sources and destinations respectively, such that every $x \in Z$ had at least one next-hop towards all $o \in V$. As a result, each of the 156 snapshots yield sets Z that range in size from 37 (Jan 2003) to 183 (Dec 2015) and sets V that range in size from 14051 (Jan 2003) to 51202 (Dec 2015). For each snapshot, the corresponding Z is used as in Equation (6.1). The details of this phase of the data processing are in Appendix B.

Stable source selection: For the static analyses in Section 6.3 the above data is sufficient. However, for the dynamic analyses in Sections 6.4-6.6, variation in the membership of sets Z introduces noise into the results. To reduce the effects of ASes that appear only infrequently in source sets Z , we keep only sources appearing in at

least 78 (half) of the 156 snapshots. After this stability filtering, the size of Z ranges from 21 (Jan 2003) to roughly 70 in the end of 2015.

6.3 Unusually-Routed ASes

We start our investigation by exploring the nature of the most unusually-routed Autonomous Systems at any given time. In this section, we develop methods for identifying unusual nodes in a fixed snapshot of the network, and illustrate the value of the methods.

The key idea behind our approach is as follows: we fix the concept of an ‘unusually-routed AS’ as meaning one for which the set of paths leading to the AS is very different from the set of paths leading to any other AS. A natural similarity measure over paths is defined in terms of the set of next-hops comprising a path. Keeping in mind the view of RSD as ‘the probability that an arbitrary node in Z uses different next hops to reach the two destinations’ we can formalize an ‘unusually-routed AS’ as one that *has high RSD to all other ASes*. As an example, one can observe that paths leading to node e in Figure 6.1 are very different from paths leading to any other node; such fact is captured by RSD, which is 1 (maximum value) between e and any other node. Hence, we say the e is unusually-routed in that path-based network.

Although such reasoning leads to an operational definition of unusually-routed ASes, an important complication arises because an organization may operate multiple ASes. Hence, there may be multiple ASes with similarly-unusual patterns of reachability. This necessitates a search for *clusters* of unusually-reached ASes.

6.3.1 Problem definition

To identify unusually-routed clusters, we seek to find a group of k non-overlapping subsets of V , say C_1, \dots, C_k , meeting three requirements:

1. each C_i is a small set;
2. the elements of a C_i are close to each other, as measured by RSD; and
3. the elements of a C_i are all far from all the elements not in C_i , as measured by RSD.

We formalize these requirements by considering the complete weighted undirected graph H , whose nodes are V and whose edges connect each pair of nodes (x, y) with weight given by $\text{RSD}(x, y)$. Using H , we define the notions of *join* and *disconnect* of $C \subseteq V$ as follows:

Definition 5. Given $C \subsetneq V$, the *join* of C , denoted by $J(C)$, is:

$$J(C) = \min_{x \in C, y \notin C} \text{RSD}(x, y).$$

Intuitively, $J(C)$ is the smallest RSD threshold at which another node joins the subset of nodes in C .

Definition 6. Given $C \subseteq V$, the *disconnect* of C , denoted by $D(C)$, is:

$$D(C) = \max_{C' \subsetneq C} \min_{\substack{x \in C' \\ y \in C \setminus C'}} \text{RSD}(x, y).$$

Intuitively, $D(C)$ is the largest RSD threshold that internally disconnects the subgraph defined by C in H .

Note that one should expect that for a cluster C of interest $J(C) > D(C)$; this means that each node in C has a closer connection to another node in C than to any node outside C . Using the above definitions we formalize our search for unusual node clusters as Problem 2.

Problem 2. Given $G(V, P)$ and integers δ and k , find k disjoint sets C_1, \dots, C_k that

$$\begin{aligned} & \text{maximize} && \min_{1 \leq i \leq k} J(C_i) \\ & \text{subject to} && 0 < |C_i| \leq \delta, \quad i = 1, \dots, k, \\ & && J(C_i) > D(C_i), \quad i = 1, \dots, k. \end{aligned}$$

Algorithm 3: FindUnusual($G(V, P), \delta, k$)

```

1  $H \leftarrow$  weighted and undirected graph  $(V, V \times V, \omega)$ , with  $\omega((x, y)) = \text{RSD}(x, y)$ 
2  $T_H \leftarrow$  a Minimum Spanning Tree of  $H$ 
3  $L \leftarrow$  edges of  $T_H$  sorted in non-increasing order
4 foreach  $e$  in  $L$  do
5      $T_H \leftarrow T_H$  minus edges with weight  $\omega(e)$ 
6      $C \leftarrow$  connected components of  $T_H$ 
7      $\mathcal{C} \leftarrow \emptyset$ 
8     for  $c \in C$  do
9         if  $|c| \leq \delta$  then
10              $\mathcal{C} \leftarrow \mathcal{C} \cup \{c\}$ 
11             if  $|\mathcal{C}| = k$  then
12                 return  $\mathcal{C}$ 
13 return  $NULL$ 

```

6.3.2 Algorithm

A solution to Problem 2 yields k clusters C_1, \dots, C_k that together satisfy the three requirements. The constraint that $|C_i| \leq \delta$ ensures that the clusters are small, and the constraint that $J(C_i) > D(C_i)$ means that elements in a cluster are generally close to each other. The third requirement is met by maximizing the objective function $\min_{1 \leq i \leq k} J(C_i)$. Observe that the C_i 's do not necessarily form a partition of V , only the unusual nodes are clustered.

Problem 2 can be solved by a polynomial time algorithm with complexity $O(|V|^2 \log |V|)$ using the algorithm with pseudo-code shown in Algorithm 3. Note that the running time computation assumes that all pairwise RSD distances $\text{RSD}(x, y)$ are provided as part of the input.

In a high level, the algorithm first builds the complete, weighted graph H . Then, it computes a minimum spanning tree of H and removes edges from the tree in non-increasing order, until it obtains k connected components smaller than δ . When we solve Problem 2 for a particular value of k we refer to the solution as the *top- k*

clusters.

Although Problem 2 is different from single-linkage clustering, Algorithm 3 has similarity to the solution strategy for the single-linkage clustering problem (Kleinberg and Tardos, 2005). We have the following result about the output of Algorithm 3.

Proposition 1. *For given $G(V, P)$, k and δ , Algorithm 3 finds the optimal solution of Problem 2.*

Proof. See Appendix C □

6.3.3 Results

Figure 6-2 is a dendrogram of the unusual AS clusters for the snapshot of December 2014, obtained by solving Problem 2 for $k = 40$ and $\delta = 50$. (Varying δ up to 200 did not change the results; large values of k generate many unusual clusters). Groups of ASes that were placed in the same cluster in the solution have been given the same color in the figure (all singletons are shown in blue). The figure illustrates a number of points:

Unusual clusters are important. The figure shows that unusual clusters often correspond to important Internet businesses or organizations. Among other significant organizations, Google, Verisign, Motorola, Microsoft, Facebook, and GoDaddy are all represented in the top-40 clusters (note that there are over 47,000 ASes in the dataset). The ASes in the top-40 are quite unusual in terms of how they are reached in the Internet: the x axis shows that the minimum RSD between these clusters and any other nodes is in the range 0.6 - 0.74. Comparing to the density plot of split values across the entire dataset, we see that these values are quite rare.

Clusters often reflect organizational boundaries. Figure 6-2 also shows that clusters of ASes are often owned by the same company; only two clusters consists of ASes that are not completely co-owned. The remaining clusters (Google, Verisign, OVH, Leaseweb, Yandex and Microsoft) all consist of separate ASes that are owned

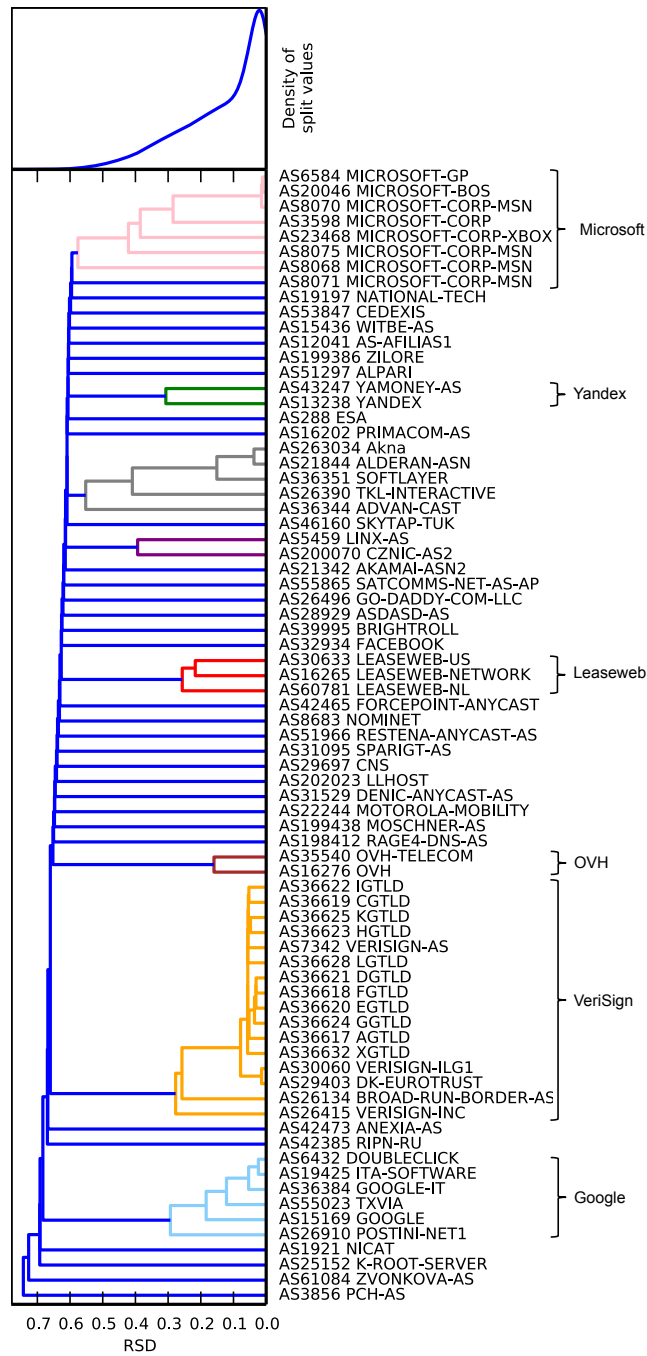


Figure 6-2: December 2014 dataset. (upper) Density of dendrogram split values across the dataset; (bottom) Dendrogram of the top-40 clusters of unusual ASes. AS descriptions are shortened from <http://bgp.potaroo.net/cidr/autnums.html>.

and operated by the same company. For example, the cluster marked ‘Google’ includes ASes that are nominally registered to Postini, TxVia, and DoubleClick – each of which reflects a prior acquisition by Google. In fact, identifying ASes owned by the same organization is important in analyzing the Internet for business and political reasons (Cai et al., 2012). These results suggest that clustering ASes using routing information may be another strategy for inferring co-ownership. Intuitively, when unusual routing strategies arise in groups, it is unlikely to be a coincidence – more likely, the participating ASes are seeking to achieve common business or engineering goals because, for instance, they are owned by the same organization.

Why unusual clusters are unusual. Each AS cluster in Figure 6·2 has a distance greater than 0.6 from *any other* AS in the dataset. Under our intuitive interpretation of RSD, this means that given an AS x in one of these clusters, at least 60% of the time an arbitrary AS $z \in Z$ chooses a different next-hop as the next step on the path towards x as compared to *any* other AS in the Internet.

To understand how this can occur, one relatively simple case is when the next-hop from z towards x is x itself – i.e., x is ‘its own next-hop.’ In fact, this can be a good approximation for certain unusual ASes (but not all, as we show later). For instance, in the group of Google ASes, AS15169 (the main Google AS) represents more than 50% of the next-hops towards any AS in the group. In other words, a large fraction of the source ASes in our study, when exchanging data with Google, do so by *directly connecting* to Google. We return to the analysis of Google in the next section; here we just note that these results are consistent with a number of other studies (Shavitt and Weinsberg, 2012; Chiu et al., 2015; Calder et al., 2013). We observed path properties similar to Google’s, if not as extreme, for a number of other unusual clusters.

Reviewing the set of organizations in Figure 6·2, a number of different business goals are evident. Furthermore, we have performed cluster analysis using Algorithm 3

for each month of our 13-year study, and the results show that various organizations have come to adopt unusual routing strategies at different times in the past. This motivates developing tools to analyze how an individual AS’s connection and routing strategy evolves over time, which we present in the next section.

6.4 Individual ASes

When analyzing an individual node, we are concerned with understanding its *path set* – the set of all paths leading to the node. An AS’s path set is a useful reflection of its business and engineering strategies. We analyze the strategies employed by individual ASes over time by asking:

1. How does the ‘unusualness’ of a node vary over time?
2. When are a node’s path sets stable, and when are they in flux?
3. What are the key path characteristics that characterize a node’s path set?

To answer these questions we make use of three tools, and introduce time-indexing on quantities of interest. First, in order to quantify how unusual a node x is when compared to the network as a whole, we define *average RSD*, denoted $\Delta_t(x)$.

Definition 7. (Average RSD) Given a destination x and a time t , define

$$\Delta_t(x) = \frac{1}{|V_t| - 1} \sum_{y \in V_t, x \neq y} \text{RSD}_t(x, y) - \frac{1}{|V_t|(|V_t| - 1)} \sum_{y, z \in V_t, y \neq z} \text{RSD}_t(z, y), \quad (6.3)$$

which measures how far the average RSD between x and the other nodes is from the global average RSD of the network. Next, to identify periods of time when path sets are stable or in flux, we use *temporal RSD* (as defined by Comarella et al. (2013)) which we denote by $\tau_{t,t'}(x)$.

Definition 8. (Temporal RSD) Given a destination x and two points in time t and t' , define

$$\tau_{t,t'}(x) = \frac{1}{|Z_{t,t'}|} \sum_{z \in Z_{t,t'}} d(N_t(z, x), N_{t'}(z, x)), \quad (6.4)$$

where the distance function d is the same as used in Equation (6.1) and $Z_{t,t'} = Z_t \cap Z_{t'}$.

Finally, in order to examine a node's path characteristics in detail, we define *next-hop distribution*, denoted $\eta_t(x, y)$.

Definition 9. (Next-hop Distribution) Given network nodes x and y , and time t , we define $\eta_t(x, y)$ as the fraction of times that x appears as a next-hop towards y at time t :

$$\eta_t(x, y) = \frac{\sum_{z \in Z_t} \text{frequency of } x \text{ in } N_t(z, y)}{\sum_{z \in Z_t} |N_t(z, y)|}. \quad (6.5)$$

We use four ASes as case studies, operated by Google (AS15169), GoDaddy (AS26496), Ams-IX-Amsterdam (Ams-IX, AS1200) and the K-Root Server (AS25152). We choose these four ASes to illustrate key features seen across the entire dataset. Together the organizations operating these four ASes conduct a very wide range of activities, but each AS is in the top-40 unusual set at certain times. Results for these ASes are presented in Figure 6-3, and are discussed individually below.

Google. Founded in 1998 as a search engine, Google became a major content provider on the acquisition of YouTube in October 2006. Figure 6-3A shows that this acquisition coincided with the beginning of a dramatic increase in its average RSD. This increase is understood via Figure 6-3I, which shows that the fraction of ASes whose next-hop on the path to Google is Google itself began to increase starting in late 2006, and has increased steadily to the present. In the most recent dataset, over 80% of the sources we studied connect directly to Google and use it as the next-hop towards Google prefixes.

The analysis shows that during the post-2006 period Google built out a world-wide network of locations connecting directly with many ASes, bypassing traditional

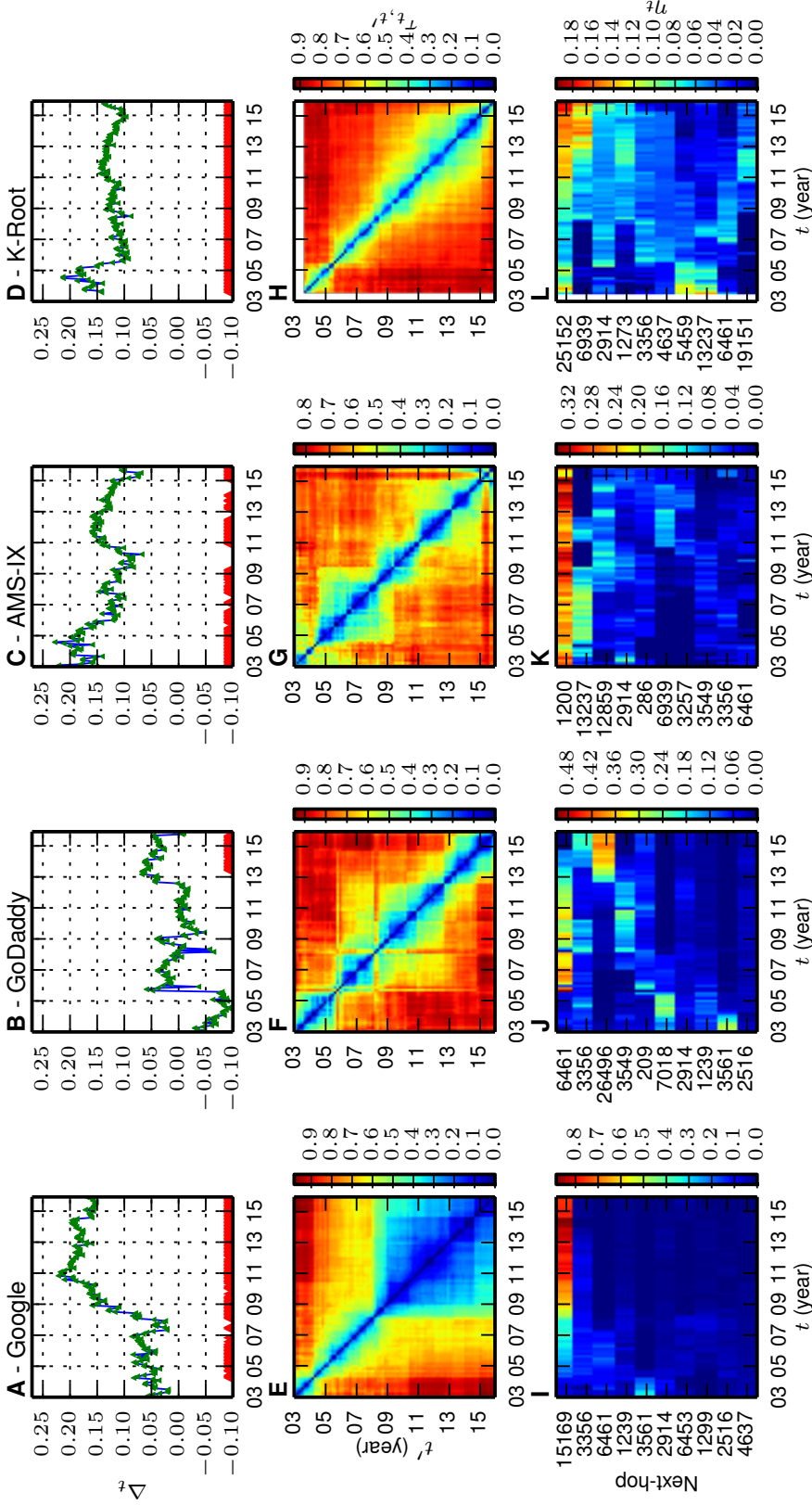


Figure 6-3: Columns correspond to ASes operated by Google (AS15169), GoDaddy (AS26496), Ams-IX (AS1200) and K-Root Server (AS25152). The top row shows Average RSD (Δ_t) time series; the red triangles at the bottom indicate when the AS was in one of the top-40 groups of unusual ASes. The middle row shows Temporal RSD ($\tau_{t,t'}$) comparing all possible pairs t and t' over the 13 years of study. The bottom row shows next-hop distribution (η_t) of the top 10 next hops used to reach the corresponding AS.

hierarchical Internet routing. Since content delivery (e.g., YouTube video) involves the transfer of much higher volumes of traffic than does search engine service, this build-out was presumably motivated by a desire to avoid the costs of paying transit providers, and to avoid network delay and congestion by being closer to customers. From mid-2008 onward Google appears continuously in the top-40 unusual ASes. Corroborating this view, the Temporal RSD plot in Figure 6-3E shows two distinct phases, before and after 2008, corresponding to Google’s transformation into a business that has a major content delivery component.

Besides Google, other companies that are known to be expanding their network infrastructure are also present in Figure 2. These include ASes owned by Microsoft and SoftLayer, agreeing with results of Chiu et al. (2015). Amazon, which is also discussed by Chiu et al. (2015), is occasionally identified as unusual in our results. Using our tools to analyze AS16509 (the main Amazon AS), we observe that the popularity of AS16509 as a next-hop to itself is consistently increasing over time, but that at present Amazon still heavily relies on large transit providers.

GoDaddy. GoDaddy is the world’s largest domain registrar, managing the assignment of many of the names held in the DNS. GoDaddy started in 1997 and began a period of rapid growth in early 2005, which included adding content-hosting services for small businesses.

Figures 6-3B, F, and J illuminate GoDaddy’s expansion strategy. On commencing growth in 2005, GoDaddy began expanding its set of connections, leading to a jump in Δ_t and a diversification of the next-hops used to reach AS26496 (visible also in Figure 6-3F). However, in early 2013, GoDaddy abruptly shifted to an unusual connection pattern, connecting directly to many networks and cutting out the intermediaries that had been dominant in 2005-2013. This is visible in the sharp rise in Δ_t in early 2013, and the sudden dominance of AS26496 as its own next-hop in

Figure 6-3J. This shift suggests an intensive effort by GoDaddy to build new network infrastructure in order to reach its customers over shorter paths. The pre-2013 period is also useful to illustrate how the output of our tools for ASes that do not behave unusually. In such case, one can observe lower RSD and higher dependency on transit providers, when compared to the post-2013 period.

Ams-IX. Ams-IX operates Internet eXchange Points (IXPs), locations established to facilitate connections between ASes. Over 700 ASes currently make use of the Amsterdam IXP as a location to connect to other ASes. Figures 6-3C, G, and K show that, unlike the previous two examples, routing to the Ams-IX AS (AS1200) has been unusual over almost the entire period of study. Figure 6-3K shows that this is because, as with the previous two cases, AS1200 is the most common next-hop on paths to itself. The difference in the temporal patterns of Δ_t compared to Google and GoDaddy arise because direct-connection occurs not due to a large infrastructure build-out, but rather to the hundreds of ASes that connect directly to Ams-IX at their IXP. These connections enable Ams-IX to provide coordination and support for the participating ASes.³

K-Root Server. As a final example, we discuss AS25152, which operates the K-Root Server. This server is an important element of the Domain Name System (DNS), so ensuring the constant availability of the services provided by K-Root is important for smooth operation of the Internet and the Web.

To provide highly available service K-Root uses anycast, which consists of creating multiple hosts with the same IP address and connecting those hosts to the Internet in different locations. The normal action of the routing system then serves to direct any request for service sent to the K-Root IP address to a nearby K-Root host; if a K-Root host fails, the routing system naturally adapts to direct data to other K-Root

³More details at <https://ams-ix.net/technical/specifications-descriptions/as1200-peering>.

hosts instead.

K-Root has been operated using anycast since near the beginning of our study period, which explains why it has consistently been an unusual AS. In contrast to the three previous cases, the unusual routing towards K-Root is not because its AS is its own most common next-hop (as can be seen in Figure 6-3L); K-Root is only connected in less than 20 locations at present. However, the connection of the K-Root AS at multiple locations means that it is not “near” any other *single* AS from the perspective of a majority of other ASes.

This shows that anycast generates routing patterns that meet our definition of unusual. Hence, it suggests that these methods might be useful for detecting anycasted ASes in general, with the advantage of being lightweight and requiring only passively collected information (routing tables). As future work we intend to evaluate such possibility and compare the results with other works, e.g., the works of Cicalese et al. (2015b) and Cicalese et al. (2015a). For instance, initial investigation showed a considerable overlap between ASes listed on Figure 6-2 and the results of Cicalese et al. (2015a).

Taken together, these case studies paint a varied picture of why and how an organization decides to make use of an unusual routing strategy for its ASes. In some cases, such as Google and GoDaddy, the goal is to build out a worldwide infrastructure to provide high performance, high reliability, or both. In other cases, the need to provide coordination services for many other ASes (at an IXP) leads to unique paths to the coordinating AS. And yet another reason for adopting unusual routing is when using anycast, which does not lead to the announcing AS using paths similar to any other particular AS in the view of most other ASes.

6.5 Evolution of Unusual ASes

Section 6.3 showed that unusually-routed ASes are often significant Internet organizations, and Section 6.4 showed that organizations make distinct decisions to adopt unusual routing strategies at particular times. This suggests that macro-level insights into the overall business and engineering goals driving Internet routing can be obtained by examining the sets of unusually-routed ASes over time.

We characterize the evolution of unusually-routed ASes as follows. First, we consider the set of unusual nodes at a given time to be the union of the top- k clusters returned as the solution to Problem 2. Then, we collect all such sets over time into an $m \times T$ binary matrix \mathbf{X} , where m is the total number of unusual nodes over all timesteps, $T = 156$ is the number of timesteps, and $x_{it} = 1$ if and only if node i is unusual at time t .

Our initial results suggest that the unusually-routed ASes in the Internet have formed a set of distinct phases over time. This evidence is shown in Figure 6-4A (See details about additional data cleaning steps for this figure in Appendix F). For each snapshot we construct its set of unusual ASes, based on the top- k clusters for $k = 45$, leading to the dataset \mathbf{X} (details on how k is selected are described later in this Section). Figure 6-4A then plots for each pair of unusual sets A and B , the overlap distance ($O_d(A, B)$), representing the fraction of the smaller set that is not contained in the larger set. It is possible to identify a noisy diagonal block structure in the heat map, suggesting that the sets of unusual ASes have gone through periods of stability that have alternated with short periods of more rapid change.

This leads to the following goals, which drive the analyses in this section: we seek to find a good segmentation of the sets of unusual ASes over time, and we seek to use that segmentation to understand the evolution of routing strategies among important Internet organizations.

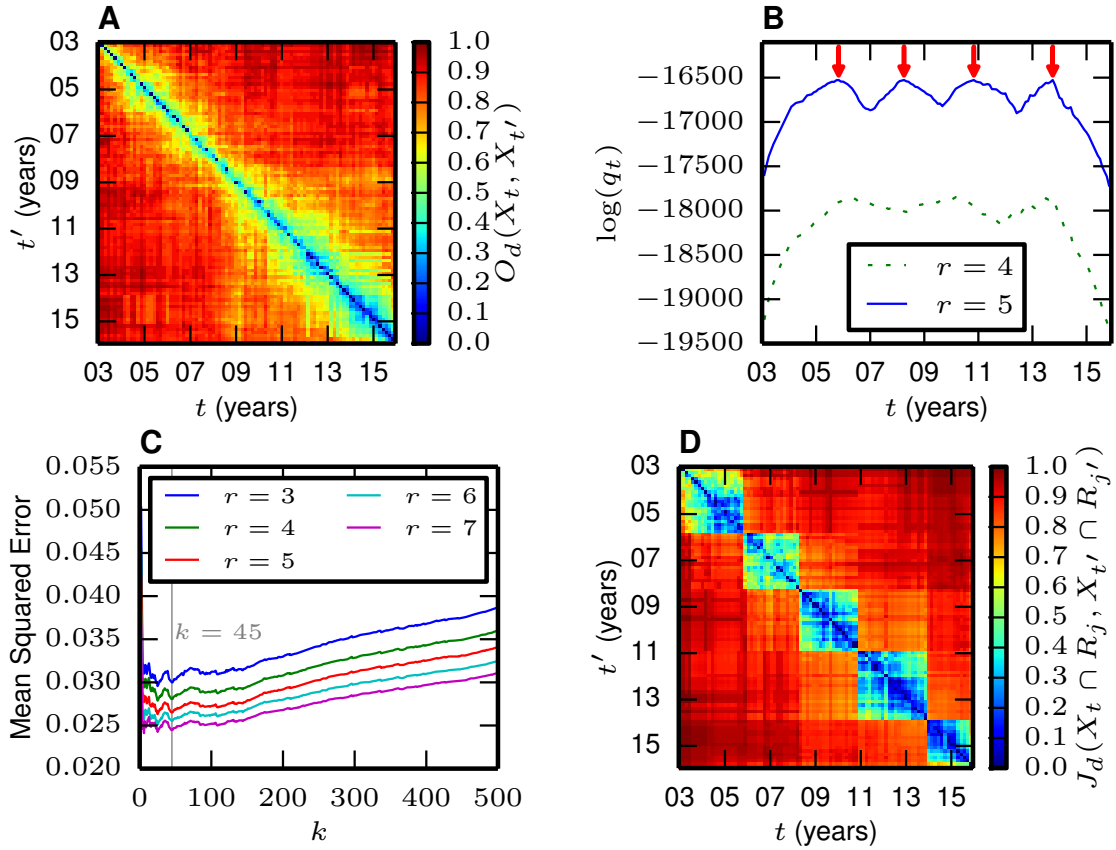


Figure 6-4: Phase-based analysis of unusual ASes for the past 13 years. Each month t is characterized by its set of top-45 unusual AS clusters, denoted by X_t . (A) O_d , overlap distance between sets of all unusual ASes at different times. (B) Segmentation scores $\log(q_{r,t})$, showing the relative quality of t as a segment boundary. Red arrows show actual segment boundaries determined by Problem 3. (C) Mean Squared Error of the segmentation model, given the data for different r and k values. (D) J_d , Jaccard distance between sets of unusual ASes after using segment representatives to filter noise.

6.5.1 Problem definition

We say that \mathbf{X} has a good r -segmentation if we can partition the columns of \mathbf{X} into r segments, in which any two columns of the same segment of \mathbf{X} show strong similarity.

To formalize this problem, let S be a set of $r + 1$ segment boundaries $1 = s_0 < s_1 < \dots < s_r = T + 1$, and $\mathbf{P} \in [0, 1]^{m \times r}$, where p_{ij} represents the probability that $x_{it} = 1$ when $s_{j-1} \leq t < s_j$. If \mathbf{X} has a good r -segmentation, much of the information in \mathbf{X} can be captured by a model $\mathcal{M} = (S, \mathbf{P})$. Hence we segment \mathbf{X} by finding a model \mathcal{M} that has high likelihood. Under the assumption of independence of the x_{it} 's the task of finding a good \mathcal{M} is formalized in the following problem definition:

Problem 3. *Given an integer $r > 0$ and an $m \times T$ binary matrix \mathbf{X} , find a model $\mathcal{M} = (S, \mathbf{P})$ that maximizes the likelihood function of \mathcal{M} given by:*

$$\mathcal{L}(\mathcal{M}|\mathbf{X}) = \prod_{j=1}^r \prod_{t=s_{j-1}}^{s_j-1} \prod_{i=1}^m (p_{ij})^{x_{it}} (1 - p_{ij})^{1-x_{it}}. \quad (6.6)$$

Hence, we cast the problem of finding a good r -segmentation to the problem of maximizing $\mathcal{L}(\mathcal{M}|\mathbf{X})$, in which the latent variables are the $r - 1$ segment boundaries and the probability matrix \mathbf{P} .

6.5.2 Algorithm

Our algorithm for solving Problem 3 is adapting the standard dynamic-programming recursion (Bellman, 1952) for segmenting the sequence of observations stored in \mathbf{X} to the likelihood function we describe in Equation (6.6). The algorithm has running time $O(T^2(r + m))$, and since T is small (156), execution takes only a few minutes. For details, see Appendix D.

An important step in identifying a good r -segmentation for \mathbf{X} is the choice of r . To assess the quality of a solution to Problem 3 for a given r , we ask whether the computed segment boundaries are sharp. We compute a score $q_{r,t}$ that indicates the

relative quality of all r -segmentations that contain t as a segment boundary. More specifically,

$$q_{r,t} = \frac{\sum_{S \in \mathcal{S}_{r,t}} \max_{\mathbf{P}} \mathcal{L}((S, \mathbf{P}) | \mathbf{X})}{\sum_{S \in \mathcal{S}_r} \max_{\mathbf{P}} \mathcal{L}((S, \mathbf{P}) | \mathbf{X})}, \quad (6.7)$$

where \mathcal{S}_r is the set of all possible segmentation boundaries of $1, \dots, T$ in r segments, and $\mathcal{S}_{r,t}$ is the set of all possible segmentation boundaries in which t is present as a boundary. Intuitively, the presence of distinct peaks in $q_{r,t}$ that correspond to the boundaries obtained by solving Problem 3 suggests that the obtained segment boundaries are well-localized and distinct.

Although computing $q_{r,t}$ requires going over *all* possible segmentations of the data into r segments, we adopt techniques from the works of Koivisto et al. (2003) and Terzi (2006) to compute the values of the above scores in polynomial time using dynamic programming. In fact, the running time of the algorithm for this task is the same as the time required to compute the optimal segmentation ($O(T^2(r+m))$). The details of $q_{r,t}$ computation are given in Appendix E.

6.5.3 Results

Computing $q_{r,t}$ for different values of r suggests that $r = 5$ is the smallest r with sharp segment boundaries. Figure 6.4B shows the $q_{r,t}$ for $r = 4$ and 5, showing as an example that segment boundaries for $r = 4$ are not as distinct as they are for $r = 5$. To choose k we examined the MSE (Mean Squared Error) of the model \mathcal{M} given the data \mathbf{X} . As shown in Figure 6.4C, increasing r always decreases the MSE, but across all curves, there is a clear pattern change after $k = 45$, with the MSE being minimum (or close to) at that point.

For $k = 45$ and $r = 5$ the segment boundaries obtained by solving Problem 3 are December 2005, May 2008, December 2010 and November 2013. Inspection of the resulting segments shows that the dominant unusual ASes differ across segments. To

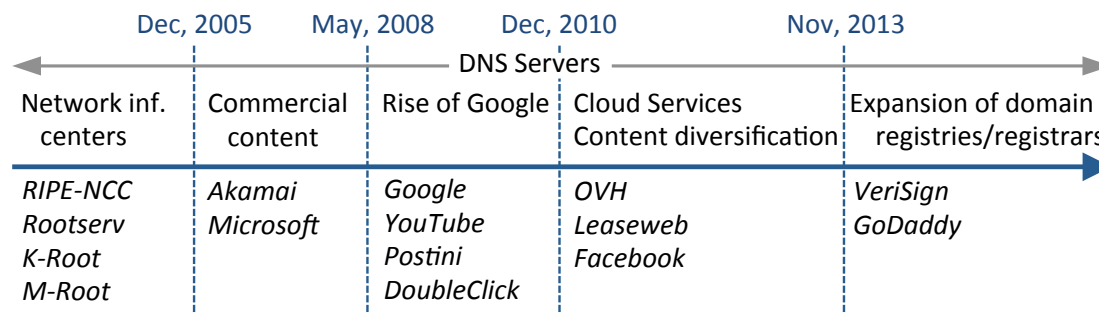


Figure 6-5: Segment representatives reflecting the evolution of unusually-routed ASes.

demonstrate this, we remove all ASes that do not appear in more than half of the snapshots of at least one segment. This has the effect of filtering noise due to ASes that only infrequently appear as unusual. While $m = 1192$ distinct ASes appear as unusual at some point over the 13 years of the study period, only 104 ASes appeared in at least half of the snapshots of a segment. We define the *representative* of segment j , denoted by R_j , as the set of ASes that appear in the unusual sets of more than half of the snapshots in segment j .

Figure 6-4D shows Jaccard distance among the sets of unusual ASes over time, after denoising by intersecting the unusual ASes in each snapshot with the respective segment representative R_j . The figure shows that representative nodes are present consistently throughout each segment, and that there is little overlap in the representative nodes across segments.

To interpret the nature of the phases shown in Figure 6-4, we manually select some well-known ASes that are characteristic of a segment's representative set and appear in that set for the first time. These are shown in Figure 6-5, and reflect a number of aspects of Internet organization over the past 13 years.

First, spanning all segments are DNS root servers that use anycast to ensure high availability and performance. This demonstrates the crucial role that DNS plays in

the Internet and the importance of reliable DNS service.

In the period before December 2005, dominant unusual nodes tend to be associated with network operations, as exemplified by DNS servers and network information centers. The period from December 2005 to May 2008 exhibits a shift towards the building of infrastructure for delivery of commercial content. Microsoft and Akamai are notable for their emergence as unusual ASes in this period. The third segment marks a turning point in which Google and its associated businesses – YouTube, Postini, and Doubleclick, among others – launch a very significant buildout of infrastructure. As shown in the previous section, the goal of this buildout was to minimize the use of the Internet as a hierarchy, instead constructing short, direct paths from many of its customers to its network. Although Google has appeared as unusual in earlier segments, one can note that the emergence of Google as the representative in this segment is in accordance with the transformations observed through Figure 6-3E. By December 2010, a number of other large content and cloud providers followed Google’s lead, and expanding their own infrastructures, including Facebook and OVH. Finally, after November 2013 a new set of organizations emerge with unusual routing structures, mostly related to the commercial exploration of DNS services, exemplified by Verisign and GoDaddy.

6.6 Global Path Characteristics

Our last set of macro-level investigations concerns the time evolution of global changes to the entire path set of the network. At the highest level, our goal is to measure the extent to which the set of all AS paths in the Internet reflects a shift away from hierarchical and toward mesh-like (flat) routing.

One of the results of Section 6.4 is that when an AS deviates from hierarchical routing, the RSD between that AS and other ASes increases. In fact, we can formalize

this effect using the following proposition:

Proposition 2. *Let $G_1(V, P_1)$ and $G_2(V, P_2)$ be path-based networks, where P_1 and P_2 are composed of shortest paths overlaid on a complete graph and on a tree respectively. Then*

$$\text{RSD}(x, y) = 1, \text{ for } G_1, \text{ and} \quad (6.8)$$

$$\text{RSD}(x, y) = \frac{1 + \text{treeDist}(x, y)}{|V|}, \text{ for } G_2, \quad (6.9)$$

where $\text{treeDist}(x, y)$ is the length of the shortest path (counting edges) between x and y in a tree.

Proof. Proof in Appendix G □

Proposition 2 shows that RSD is capable of distinguishing the extremes of hierarchical and flat routing schemes. In particular, the RSD between two nodes assumes its highest value for flat routing: for G_1 the pairwise distances assume value 1. In contrast, RSD values in hierarchical routing are relatively low – roughly $O(\log(|V|)/|V|)$ for G_2 – assuming that the underlying tree of G_2 is balanced.

These observations suggest an approach for characterizing the entire path set of the network in order to assess Internet flattening over time. To that end, we introduce the following metric.

Definition 10. (Global RSD) *Given a path-based network $G(V, P)$, define*

$$\Delta(G) = \frac{1}{|V|(|V| - 1)} \sum_{x, y \in V, x \neq y} \text{RSD}(x, y). \quad (6.10)$$

The motivation for Global RSD is that if Internet flattening is taking place, then average RSD values across the entire network should be increasing.

Figure 6.6A shows Global RSD for each of the snapshots in our dataset. The figure shows a fairly steady growth of Global RSD over the 13 years of our study. According to Proposition 2, such growth is consistent with a shift from tree-like to mesh-like routing. Interestingly, we observe that Global RSD growth had been taking place well before Internet flattening was first reported by Gill et al. (2008).

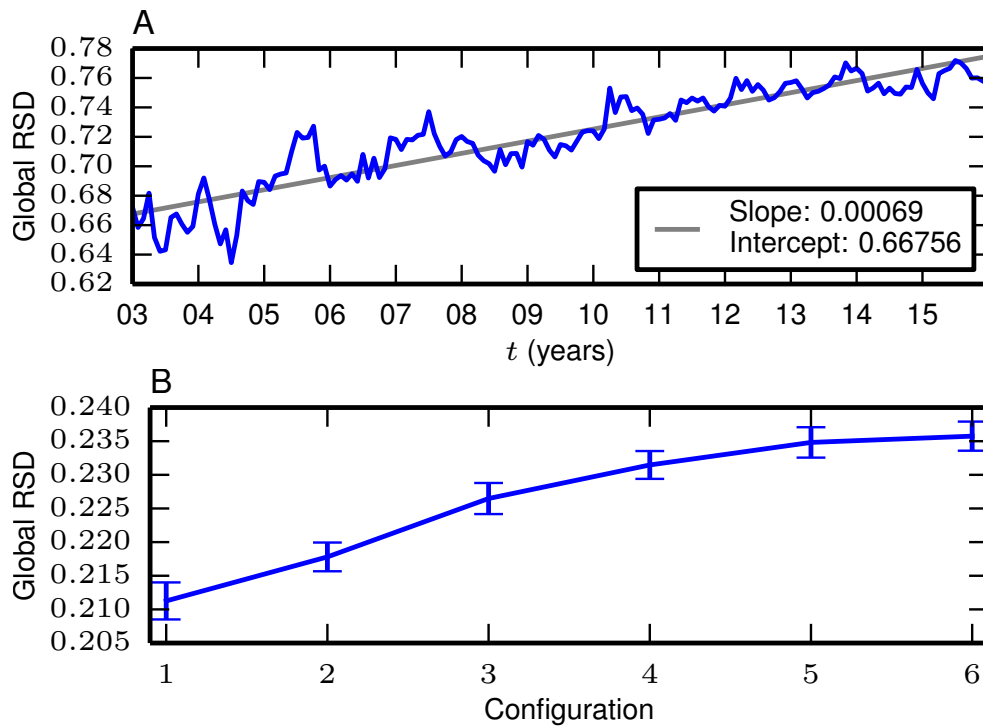


Figure 6-6: (A) Global RSD over the 13-year period of study. The line is fit using weighted least squares (using the variance of pairwise RSD at each time). (B) Global RSD for the six ITER configurations of our simulation study. Configurations 1 and 6 represent hierarchical and mesh-like structures respectively. Error bars are 95% confidence intervals.

One concern in interpreting Figure 6.6A is that the idealized routes considered in Proposition 2 differ considerably from the more complex set of routes used in the Internet. To explore whether RSD can reflect flattening in routing patterns more representative of the Internet, we turn to simulation. For this purpose we use ITER (Dhamdhere and Dovrolis, 2010), an agent-based simulator specifically developed to shed light on the transition of the Internet from a hierarchical scheme to one closer to a peering mesh. ITER uses an agent-based formulation in which ASes individually adjust their connections to other ASes to meet business strategies such as profitability.

Dhamdhere and Dovrolis (2010) vary three parameters in ITER to study the transition from hierarchical to mesh-like routing. These are: the number of regions that a content provider can span (R), the fraction of traffic generated by content providers (C), and the traffic threshold for peering (α ; we refer the reader to the original work of Dhamdhere and Dovrolis (2010) for details).⁴ Dhamdhere and Dovrolis (2010) use settings of (R, C, α) equal to $(1, 0.1, 1)$ and $(6, 0.6, 10)$ to study hierarchical and flat routing structures respectively. To study varieties of networks ranging from hierarchical routing to flat routing, we use six parameter settings that interpolate between these two extremes. For each of the six configurations, we execute 100 runs of the simulator. Each run results in an AS topology with annotated link relationships (peer or customer-provider). Using these, we employ standard algorithms to infer non-valley-prefer-customer paths (Gao and Wang, 2002), and then compute global RSD as in Definition 10.

Figure 6.6B presents Global RSD averaged over each of the 100 runs. The figure shows that RSD responds smoothly to the transformation of the network from a hierarchical to a flat structure. Thus, we conclude that the effect predicted by Proposition 2 extends to the situation in which the set of paths is more representative of

⁴We use here the notation from the original work of Dhamdhere and Dovrolis (2010) for consistency. ITER parameters in this section should not be confused with the same symbols in other sections.

actual Internet routing.

Taken as a whole, the results in this section suggest that Global RSD has been growing over the 13-year period of our study. Furthermore, we conclude that Global RSD changes predictably as the set of paths in a network transitions from hierarchical to mesh-like, and that the changes in Global RSD we observe are consistent with reports of such changes taking place in the Internet.

6.7 Data Considerations

The data sources for our study (Route Views and RIPE RIS) determine the visibility we have on the Internet’s routing structure. A natural question, therefore, concerns the impact of that visibility on our results. In that regard, we address two questions:

1. The monitoring points for Route Views and RIPE RIS yield full visibility mainly for Internet transit providers. Are such points good locations for observing global Internet evolution?
2. The nature of the Route Views and RIPE RIS monitoring points is such that many unobserved links are of the peering type, particularly those formed in IXPs (Internet eXchange Points) (He et al., 2009; Augustin et al., 2009; Ager et al., 2012; Oliveira et al., 2010). How does this affect our results?

6.7.1 Monitor locations

To understand whether the transit providers fully visible through Route Views and RIPE RIS are good locations for observing global routing changes, we first observe that the degrees of freedom in making routing decisions are much greater for highly-connected transit providers than for the majority of ASes that are at the edge of the network. For example, consider a stub AS that is single-homed to its provider, through which it reaches every destination. In that case, the next-hop from that AS

to *every other AS* is constant. In other words, its routing table is constrained to carry the same entry for every destination.

Another way of stating this is that a subset of paths carries most of the information used in RSD analysis. For example, the next-hops from a stub AS provides no *information* to measurements of RSD, since $N(x, \cdot)$ is a constant function for a single-homed stub AS x .

To demonstrate this effect in realistic Internet routing, we provide further analysis of our ITER simulation runs. ITER places ASes into four different categories: Enterprise Customers (EC), Small Transit Providers (STP), Large Transit Providers (LTP) and Content Providers (CP). Depending on their category, ASes have different functions to optimize and interconnection strategies. We use the default configuration of 500 ASes, consisting of 430 enterprise customers, 22 content providers, 38 small transit providers, and 10 large transit providers. In the default configuration, about half of the enterprise customers will be singly-homed, and half will be multi-homed.

To investigate the contribution that an AS as a source makes in measuring Global RSD, we have Definition 11.

Definition 11. (*RSD contribution*) *Given an AS s , its contribution as a source to global RSD is defined as*

$$\nu(s) = \frac{1}{|V|(|V| - 1)} \sum_{x, y \in V, x \neq y} d(N(s, x), N(s, y)), \quad (6.11)$$

where d and N are defined as for Equation (6.1).

For each of the 100 runs within each simulator configuration, we ranked ASes according to their ν values within categories, and then averaged the 100 values for each rank.

Figure 6.7 presents $\nu(s)$ for each of the four categories of ASes, considered as potential monitoring points. The figure shows the results for each simulator config-

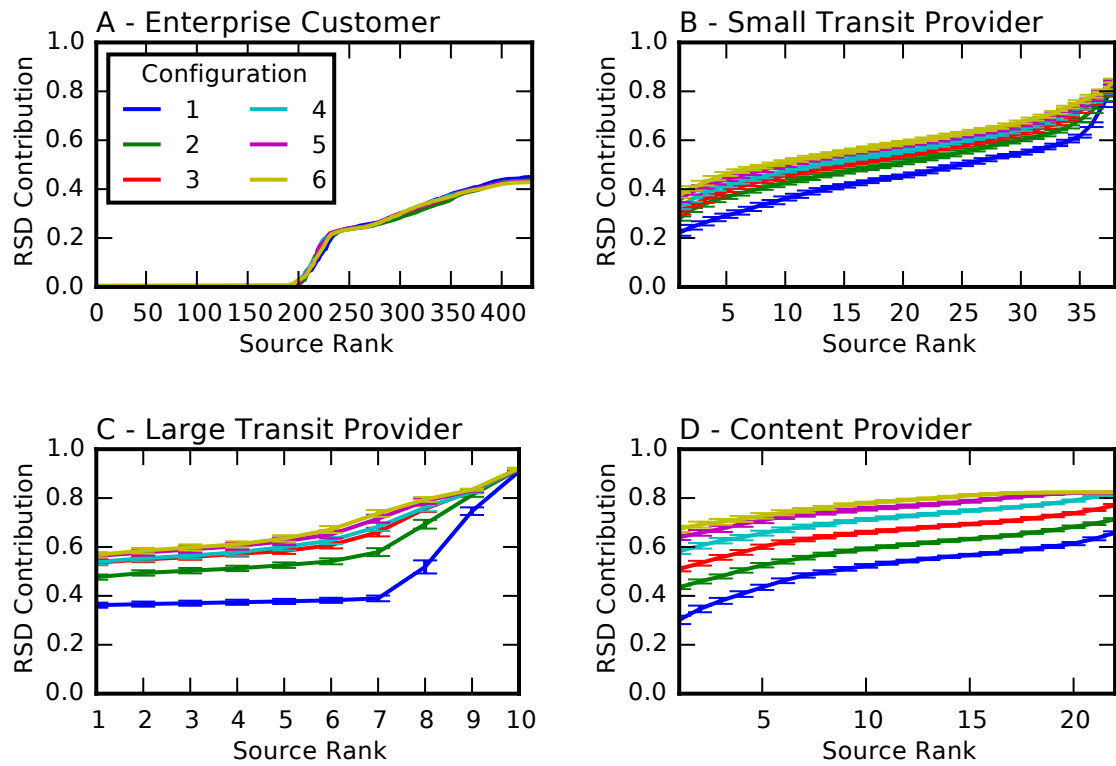


Figure 6-7: RSD contribution for different AS categories and parameter configurations. Error bars are 95% confidence intervals – Omitted for the leftmost figure to improve visibility.

uration, ranging from hierarchical routing (Configuration 1) to flat routing (Configuration 6). The figure shows clearly how the “role” of an AS within the Internet affects the amount of RSD information contributed by the AS. Figure 6.7A shows that, across all configurations, the first 200 or so EC ASes are single-homed enterprise customers. These ASes have *no effect* on any RSD values. The next 230 or so ASes are multi-homed enterprise customers, which make modest contributions to RSD measurements. However, there is no impact of these measurements on Global RSD across routing configurations. That is, the significant differences in RSD contribution across the ITER configurations occur essentially *only* among the transit providers and content providers. Hence, all the information necessary to construct Figure 6.6 as a measure of Internet flattening can be obtained solely by measuring a *small subset* of the most informative AS paths.

Figures 6.7B and C show that transit providers make much larger contributions to overall RSD measures. Furthermore, these monitoring locations allow the observation of distinct differences across the various routing configurations. Finally, we note that the greatest variation in RSD contribution across routing configurations is made by content providers, which makes sense since content providers are mainly driving the evolution of routing from hierarchical to mesh-like structure.

We conclude that transit providers and content providers that have many neighbors represent particularly informative monitoring points for measuring RSD. However, we acknowledge that this conclusion is subject to limitations of ITER, which in some cases do not capture real-world complexities. For example, ITER does not include peering of ASes in the EC category.

6.7.2 Peering links

Fully understanding the impact of missing peering links on our results is difficult in the absence of historical data that includes peering links. However, we are able to

assess the impact of missing link data at two specific time points, ten years apart. We make use of two datasets from different sources and times; each contains many AS links from IXPs not present in our original data. The first is from May 2005 He et al. (2009), and the second from September 2015.⁵ Each of these datasets is the result of a measurement campaign designed to capture missing AS links, typically within IXPs.

When comparing the new and old datasets we identified 10504 and 27412 new undirected links for 2005 and 2015 respectively. The main challenge in integrating these datasets is that they contain links rather than paths. We could simply add the corresponding link as a single next-hop, corresponding to a one-hop path, but that is too conservative; the presence of the link can be used to infer additional next-hops as well. Accordingly, for each new AS link (x, y) (we take same actions from y to x), we chose to add y as the next-hop from x for all the destinations in the *customer cone* of y , as defined by Luckie et al. (2013).⁶ From the customer cone of y we did not consider ASes that were in the customer cone of x and peers of x . These resulting inferred next-hops are consistent with traditional Internet routing models, which despite their limitations (Anwar et al., 2015), we use as an approximation of what one would expect to be visible if we had access to the routing table from x . In many cases, such a strategy may not yield the exact set of next-hops that actually involve the link (x, y) – for example, when a customer of y is multi-homed the path to it from x may not pass through y . However, we emphasize that the goal of this exercise is not to obtain precise AS topologies.

By following the steps presented in Section 6.2 (except the last one) we obtained new next-hop matrices containing entries related to data from RIPE, Route Views and the new AS-links. When comparing these new matrices with the original ones

⁵<http://sg-pub.ripe.net/emile/ixp-country-jedi/>

⁶ <http://www.caida.org/data/active/as-relationships/>

(with only data from RIPE and Route Views), we observed only minor changes in our results. More specifically, global RSD changed by less than 1.1% and the sets of unusual ASes (for $k = 45$ and 100 , and for $\delta = 50$) changed by less than 15% (for both the 2005 and 2015 datasets). Therefore, the impact of missing links in the next-hop matrices analyzed in this work is not significant, and minor variation should be expected.

However, these results do not prove that unknown AS-links are unimportant. The methods that we use rely on vantage points (sources) from which paths to most Internet locations are visible (such as are provided by Route Views and RIPE). Adding the new data, unfortunately, does not provide additional vantage points of this type. The reason is that many of the new AS-links are related to IXPs, and paths crossing IXPs typically reach certain customer cones, but not the majority of the Internet. As a result, it is difficult to practically assess the impact of the missing paths that cross IXPs. We note that these initial studies did reveal small areas with high RSD values in the new next-hop matrices. We conclude that a full assessment of the impact of missing IXP links on RSD values is an open question and a valuable direction for future work.

6.8 Summary of the chapter

In this chapter, we showed that, at the level of individual ASes, characterizing paths can expose relationships among ASes in terms of managing organizations, and can reflect how an organization's business goals and the associated engineering strategies shift over time. In particular, we show that organizations deviate from hierarchically-routed connections for a number of reasons, from content delivery (exemplified by Google), to service reliability (exemplified by DNS servers), to inter-node coordination (exemplified by Ams-IX).

At the level of the whole network, characterizing paths can expose phases in the sets of ASes that are most unusually routed over time. Applying this analysis to the Internet details how content delivery began to drive the Internet flattening in late 2005, expanding in 2008 with Google's move into video delivery, and maturing by mid-2010.

Chapter 7

Conclusions

The set of paths in a path-based network has rich information content; analyzing them can expose nodes with unusual path sets, identify how a node’s path set changes, and uncover network-wide transitions over time.

In this dissertation, we proposed new ways to study the Internet at the AS level, as an instance of path-based network. We started by using a different representation of the system (compared to prior work), proposed several efficient methods for analyzing large volumes of data, and applied them to sets of paths collected over the last decade.

In the first part of this dissertation, we introduced a new metric, TRSD, for quantifying how the set of paths towards a prefix changes over time. By applying TRSD over the set of all prefixes in a given timescale we were able to measure the rate of path changes in the Internet. Despite our global focus, it is important to note that the results presented in Chapter 4 offer opportunities for TRSD. For instance, it may be usefully applied to individual prefixes, i.e., in a microscopic analysis, supporting the detection of malicious attacks (e.g., hijacking and rerouting). Another opportunity is applying TRSD in order to analyze router-level topologies. Although a router-level study is a challenge due to the lack of publicly available data, an analysis at this granularity would be interesting because the same AS path may have traffic flowing through different physical paths.

Next, we presented PathMiner, a system for identifying and analyzing high-impact events in the interdomain Internet. We showed that PathMiner is able to identify

many large events from real BGP snapshots. Despite showing successful event detection, we acknowledge the need to better understand cases where PathMiner is not capable of finding a single actor responsible for the event (as exemplified in one of our case studies). However, to the best of our knowledge, PathMiner is the first system capable of exploring the amount of information we analyzed, finding daily large routing events, and recognizing that such events re-occur many times during the period of one year, in some cases months apart.

Finally, we described a framework for identifying and understanding unusually-routed ASes, and we showed that those ASes are important for the evolution of the Internet. Moreover, our results also suggest some directions for future study, that were out of the scope of this dissertation. First, we note the need of further investigation with regard to missing data, since from the RIPE RIS and Route Views projects we cannot obtain all Internet paths. Second, by considering only ASes and the prefixes they originate, we are not able to distinguish addresses associated with caches deployed inside access networks (by content providers, content delivery networks, etc.). One possible approach to circumvent this limitation is to consider destinations as services, instead of ASes (e.g., `google.com` instead of AS15169). To that end, it is necessary to obtain paths towards a large variety of services, requiring active measurements – for instance, from looking glass servers and RIPE Atlas.¹

Third, we showed that groups of unusual ASes often correlate with ASes that are owned by an organization. Hence, routing information can be used to support other techniques of co-owned ASes detection (Cai et al., 2012). In this context, it may be worth investigating an AS that joins/leaves an unusual group in order to decide whether it was a simple routing strategy change or reflecting acquisition or sale of another company. Similarly, unusual ASes may be a good starting point to detect anycast adoption. There are other approaches to that end (Cicalese et al., 2015b,a),

¹<https://atlas.ripe.net>

but addresses announced by these unusual ASes can be obtained without the need of active measurements.

Fourth, the RSD framework can also benefit organizations. At the operational level, the framework can also provide valuable information to an organization about its competition. More specifically, observing the movement of a group of ASes in RSD space can expose how their business and engineering strategies behave and change over time. Naturally, these questions demand the ability to produce results in real (or near to real) time, which could be achieved by implementing our tools on top of BGPStream.²

As a final remark, we note that the tools presented in this dissertation can be applied whenever a node set is equipped with a set of paths; for example, they can be applied to any path-selection strategy used in a network, such as shortest-path or maximum-flow. Based on the results of this study, we believe that these tools show promise for the analysis of other path-based networks, including the movement of goods in transportation networks and the movement of information in social networks.

²<https://bgpstream.caida.org>

Appendix A

Algorithm for clustering 2D-events

Algorithm 4: 3D-Factorization

Data: S , set of triples of the form (I, J, K) , and thresholds γ and β

- 1 $F \leftarrow \emptyset$
- 2 **while** $S \neq \emptyset$ **do**
- 3 $s \leftarrow$ pick and **do not** remove an element of S
- 4 $S' \leftarrow \{x : x \in S \text{ and } d_B(x, s) \leq \gamma\}$
- 5 find maximal $S'' \subseteq S'$ such that $\max_{x, y \in S''} d_B(x, y) \leq \gamma$
- 6 **if** $|S''| > 1$ **then**
- 7 $y \leftarrow \text{COMBINE-UNION}(S'')$
- 8 $S \leftarrow S \setminus S''$
- 9 $S \leftarrow S \cup \{y\}$
- 10 **else**
- 11 $F \leftarrow F \cup \{s\}$
- 12 $S \leftarrow S \setminus \{s\}$
- 13 $S' \leftarrow \{x : x \in S, s_B(x, s) \geq \beta \text{ and } \text{COMBINE-INTER}(s, x) \neq s\}$
- 14 **if** $|S'| > 0$ **then**
- 15 $s' \leftarrow \arg \max_{x \in S'} s_B(s, x)$
- 16 $y \leftarrow \text{COMBINE-INTER}(s, s')$
- 17 $S \leftarrow S \cup \{y\}$
- 18 **return** F

Appendix B

Algorithm for selecting sources and destinations

Algorithm 5 describes the procedure we used to obtain sets of ASes, as sources and destinations, for which full next-hop visibility was available. In the algorithm, $I(x)$ maps AS x to the set of destinations towards which x knows at least one next-hop.

Algorithm 5: SourcesDestinationsSelection()

```

1  $A \leftarrow$  all ASes in the dataset
2  $Z \leftarrow \{\}$ 
3  $V \leftarrow A$ 
4 while  $Z \neq A$  do
5    $s \leftarrow \arg \max_{x \in A \setminus Z} |V \cap I(x)|$ 
6   if  $|V \cap I(s)| \approx |V|$  then
7      $Z \leftarrow Z \cup \{s\}$ 
8      $V \leftarrow V \cap I(s)$ 
9   else
10    return  $Z, V$ 
11 return  $Z, V$ 

```

The algorithm builds sets Z and V with the invariant that in every iteration of the **while** loop (line 4) every $x \in Z$ has at least one next-hop towards every $y \in V$. Z starts empty and V with all ASes in the dataset. In each iteration of the algorithm, we look for the source s such that $I(s)$, number of destinations towards which s has a next-hop, has maximum $|V \cap I(s)|$. Next, if the difference between V and $V \cap I(s)$ is significant (line 6), then we stop the procedure, because it is not possible to add

more sources without a significant impact on the set of destinations. We say that the difference is significant if $|V|$ and $|V \cap I(s)|$ differ by more than ϵ . In our experiments, we conservatively fixed $\epsilon = 0.5\%$ in order to keep most of the available destinations.

Appendix C

Proof of Proposition 1

Let H be a complete graph over the set of nodes V , where the weight of edges are represented by function ω , that maps each edge $e = (x, y)$ to $\text{RSD}(x, y)$.

Lemma 1. *Let $\emptyset \neq W \subseteq V$. If $\{W_1, \dots, W_\ell\}$ is an arbitrary partition of W , with $W_i \neq \emptyset$ and $\ell \geq 2$, then for every $1 \leq i \leq \ell$ there exists $j \neq i$, $1 \leq j \leq \ell$, such that there is an edge e connecting W_i and W_j with $\omega(e) \leq D(W)$.*

Proof. Denote by $H_W(W, E_W, \omega)$ the subgraph induced by W in H . Suppose there exists i such that for all $j \neq i$ and for any edge e connecting W_i to W_j it is the case that $\omega(e) > D(W)$. Let $e' = (x, y)$ be the edge that minimizes ω with $x \in W_i$ and $y \in W_j$ ($j \neq i$). Hence, $\omega(e') > D(W)$. Then we have,

$$\begin{aligned} D(W) &= \max_{W' \subsetneq W} \min_{\substack{u \in W' \\ v \in W \setminus W'}} \omega((u, v)) \geq \min_{\substack{u \in W_i \\ v \in W \setminus W_i}} \omega((u, v)) \\ &= \omega(e') > D(W), \end{aligned}$$

which is a contradiction. □

Lemma 2. *Let $\emptyset \neq W \subsetneq V$. In any Minimum Spanning Tree T_H of H all the edges $e = (u, v)$, belonging to the tree, with $u \in W$ and $v \in V \setminus W$ are such that $w(e) \geq J(W)$.*

Proof. Suppose there exists a minimum spanning tree T_H of H that contains an edge $e = (u, v)$ such that, $u \in W$, $v \in V \setminus W$ but $\omega(e) < J(W)$. Since e is in T_H , it is also in H , hence

$$\omega(e) < J(W) = \min_{\substack{x \in W \\ y \notin W}} \omega((x, y)),$$

which is a contradiction. □

Lemma 3. *Let $\emptyset \neq W \subsetneq V$. In any Minimum Spanning Tree T_H of H , if $e = (u, v)$, $u \in W$ and $v \in V \setminus W$, and e has minimum weight among edges connecting W to $V \setminus W$ in T_H , then $\omega(e) = J(W)$.*

Proof. Suppose there exists a minimum spanning tree T_H of H with an edge $e = (u, v) \in T_H$ such that, $u \in W$, $v \in V \setminus W$, $\omega(e)$ has minimum weight among edges connecting W to $V \setminus W$ and $\omega(e) > J(W)$ (from Lemma 2 it cannot be smaller). Pick $e' = (u', v')$ in H , with $u' \in W$ and $v' \in V \setminus W$, such that $\omega(e') = J(W)$ (see that by definition of $J(W)$ such edge exists).

Add e' to T_H . By doing that a cycle is formed in the tree, and in this cycle we have another edge e'' that connects W to $V \setminus W$ in T_H . Remove e'' from T_H obtaining a new spanning tree T'_H . See that $\omega(e'') \geq \omega(e) > \omega(e')$. We have

$$\sum_{e \in T'_H} \omega(e) = \sum_{e \in T_H} \omega(e) - \omega(e'') + \omega(e') < \sum_{e \in T_H} \omega(e),$$

which is a contradiction because T_H is a MST, and hence it has minimum cost. The last inequality comes from the fact that $\omega(e'') > \omega(e')$. \square

Lemma 4. *Let $\emptyset \neq W \subsetneq V$. If $J(W) > D(W)$, then W induces a connected subgraph in any MST of H .*

Proof. Suppose $\emptyset \neq W \subsetneq V$ with $J(W) > D(W)$, and that there exists a MST T_H of H in which W does not induce a connected subgraph. Say W induces l connected components in T_H denoted by $\{W_1, \dots, W_l\}$. Pick W_1 , by Lemma 2 we have that W_1 connects to $V \setminus W_1$ in T_H through an edge e_1 such that $\omega(e_1) \geq J(W)$.

By Lemma 1 we have that there is an edge, in H , e_2 that connects W_1 to W_j for some $2 \leq j \leq l$ and $\omega(e_2) \leq D(W)$.

Now, remove e_1 and add e_2 to T_H , obtaining T'_H . Then, since $J(W) > D(W)$ we have

$$\sum_{e \in T'_H} \omega(e) = \sum_{e \in T_H} \omega(e) - \omega(e_1) + \omega(e_2) < \sum_{e \in T_H} \omega(e),$$

which is a contradiction, since T_H is a MST of H it has minimum cost. The last inequality comes from the fact that $\omega(e_1) > \omega(e_2)$, once $J(W) > D(W)$. \square

With the above lemmas, we move to prove Proposition 1. Let T_H be the MST computed in Step 2 of Algorithm 1. Also, let $\mathcal{C} = \{C_1, \dots, C_k\}$ be the output of

Algorithm 1, and assume $J(C_1) \geq \dots \geq J(C_k)$. Similarly, consider $\mathcal{B} = \{B_1, \dots, B_k\}$ be the optimal solution for Problem 1, and assume $J(B_1) \geq \dots \geq J(B_k)$.

Suppose now that \mathcal{B} is a better solution than \mathcal{C} , i.e., $J(B_k) > J(C_k)$.

From Lemma 4, the B_i 's and C_i 's induce connected subgraphs in T_H . As a consequence, all the B_i 's (C_i 's) can be obtained by removing edges from T_H . Moreover, from Lemma 2 those edges have the weight greater than $J(B_k)$ ($J(C_k)$).

Now, let e^{b_i} and e^{c_i} be edges in T_H such that $\omega(e^{b_i}) = J(B_i)$ and $\omega(e^{c_i}) = J(C_i)$ respectively (From Lemma 3 we know that such edges exist).

Since $J(B_k) > J(C_k)$, we have $\omega(e^{b_k}) > \omega(e^{c_k})$. Now see that by the construction of Algorithm 1 this is a contradiction. Because the algorithm would have inspected all edges heavier than e^{b_k} (including edges with weight $\omega(e^{b_k})$) before reaching edge e^{c_k} , yielding a valid solution, and its output would not be \mathcal{C} .

Appendix D

Algorithm for segmentation

The goal of this section is to show how to find $\mathcal{M} = (S, \mathbf{P})$ that maximizes $\mathcal{L}(\mathcal{M}|\mathbf{X})$. Recall that the input is an $m \times T$ binary matrix \mathbf{X} , where $x_{it} = 1$ if and only if the node represented by the integer i is unusual at time t . We have,

$$\mathcal{L}(\mathcal{M}|\mathbf{X}) = \prod_{j=1}^r \prod_{t=s_{j-1}}^{s_j-1} \prod_{i=1}^m (p_{ij})^{x_{it}} (1 - p_{ij})^{1-x_{it}}, \quad (\text{D.1})$$

where r is a given parameter, i.e., the number of segments.

Consider the following definitions:

- $\mathbf{X}[a, b]$ as what remains of \mathbf{X} when removing columns before a and after b (columns a and b are not removed);
- $\mathcal{L}^*(l, m)$ as the maximum value that \mathcal{L} can assume when segmenting $\mathbf{X}[1, l]$ in m segments; and
- $U(a, b)$, as the maximum value that \mathcal{L} can assume when segmenting $\mathbf{X}[a, b]$ in a unique segment.

Hence, we are interested in computing $\mathcal{L}^*(T, r)$ and the model that yields such value. \mathcal{L}^* , for $m \leq l$, satisfies the following recurrence equation:

$$\mathcal{L}^*(l, m) = \begin{cases} \max_{m-1 \leq l' < l} \{\mathcal{L}^*(l', m-1)U(l'+1, l)\} & \text{if } m \geq 2, \\ U(1, l) & \text{if } m = 1. \end{cases} \quad (\text{D.2})$$

Therefore, if we can compute $U(a, b)$ for every $1 \leq a \leq b \leq T$, we can then recursively compute $\mathcal{L}^*(l, m)$. In order to speed up the computation significantly, one can make use of dynamic programming, by building a $T \times r$ table in order to store computed values for future use.

Moreover, the segment boundaries can be obtained as a byproduct of the procedure. More specifically, see that the l' that maximizes $\mathcal{L}^*(l, m)$ shows exactly where the $(m - 1)$ -th segment ends. Hence, we can use a secondary $T \times r$ table ℓ to save the segment boundaries, where

$$\ell(l, m) = \begin{cases} \arg \max_{m-1 \leq l' < l} \{\mathcal{L}^*(l', m - 1)U(l' + 1, l)\} & \text{if } m \geq 2, \\ l & \text{if } m = 1. \end{cases} \quad (\text{D.3})$$

In the end of the procedure, we can backtrack table ℓ in order to obtain the $r - 1$ unknown segment boundaries. For instance, $\ell(T, r)$ indicates where the $(r - 1)$ -th segment ends.

The problem that remains is how to compute table U . See that, for each $a \leq b$ we need to compute an $m \times 1$ probability matrix \mathbf{P} that maximizes \mathcal{L} when segment $\mathbf{X}[a, b]$ in one segment. In other words

$$\mathbf{P} = \arg \max_{\mathbf{P}' \in [0, 1]^{m \times 1}} \prod_{t=a}^b \prod_{i=1}^m (p'_{i1})^{x_{it}} (1 - p'_{i1})^{1-x_{it}} \quad (\text{D.4})$$

$$= \arg \max_{\mathbf{P}' \in [0, 1]^{m \times 1}} \sum_{t=a}^b \sum_{i=1}^m (x_{it} \log p'_{i1} + (1 - x_{it}) \log(1 - p'_{i1})). \quad (\text{D.5})$$

Define

$$u_{a,b} = \sum_{t=a}^b \sum_{i=1}^m (x_{it} \log p'_{i1} + (1 - x_{it}) \log(1 - p'_{i1})). \quad (\text{D.6})$$

Inspecting the first and second partial derivatives of $u_{a,b}$ with respect to the p'_{i1} 's we

have that $u_{a,b}$ is maximized when

$$p'_{i1} = \frac{1}{b-a+1} \sum_{t=a}^b x_{it}, \quad i = 1, \dots, m. \quad (\text{D.7})$$

Therefore, table U can be easily obtained in polynomial time, more specifically, in time $O(T^3m)$. However, it is possible to improve such time to $O(T^2m)$ by using another level of dynamic programming. In that case the total time to obtain $\mathcal{M} = (S, \mathbf{P})$ is $O(T^2r + T^2m)$, where $O(T^2r)$ is necessary to fill tables \mathcal{L}^* and ℓ , and $O(T^2m)$ is necessary to compute table U . The remaining of this section is used to show how to compute U in time $O(T^2m)$.

Consider

$$\bar{x}_{i,a,b} = \frac{1}{b-a+1} \sum_{t=a}^b x_{it} \quad (\text{D.8})$$

and

$$\tilde{x}_{i,t} = \sum_{w=1}^t x_{iw}. \quad (\text{D.9})$$

Then we have

$$U(a, b) = \prod_{t=a}^b \prod_{i=1}^m (\bar{x}_{i,a,b})^{x_{it}} (1 - \bar{x}_{i,a,b})^{1-x_{it}} \quad (\text{D.10})$$

$$= \prod_{i=1}^m \prod_{t=a}^b (\bar{x}_{i,a,b})^{x_{it}} (1 - \bar{x}_{i,a,b})^{1-x_{it}} \quad (\text{D.11})$$

$$= \prod_{i=1}^m (\bar{x}_{i,a,b})^{\sum_{t=a}^b x_{it}} (1 - \bar{x}_{i,a,b})^{b-a+1-\sum_{t=a}^b x_{it}}. \quad (\text{D.12})$$

$$(\text{D.13})$$

Now see that for $b \geq a$

$$\bar{x}_{i,a,b} = \begin{cases} \frac{\tilde{x}_{i,b} - \tilde{x}_{i,a-1}}{b-a+1}, & a > 1 \\ \frac{\tilde{x}_{i,b}}{b}, & a = 1 \end{cases} \quad (\text{D.14})$$

and

$$\sum_{t=a}^b x_{it} = \begin{cases} \tilde{x}_{i,b} - \tilde{x}_{i,a-1}, & a > 1 \\ \tilde{x}_{i,b}, & a = 1. \end{cases} \quad (\text{D.15})$$

Hence, if one has all $\tilde{x}_{i,t}$'s values in hand, the table U can be computed in time $O(T^2m)$. In fact, it is possible to compute $\tilde{x}_{i,t}$ in time $O(Tm)$ by using a third level of dynamic programming based on the recurrence below

$$\tilde{x}_{i,t} = \begin{cases} x_{it}, & t = 1 \\ \tilde{x}_{i,t-1} + x_{it}, & 1 < t \leq T \\ 0, & \text{otherwise.} \end{cases} \quad (\text{D.16})$$

Therefore, the total time for computing U is $O(T^2m)$.

Appendix E

Computing $q_{r,t}$

The score we use, denoted by $q_{r,t}$, computes the quality of t as a segment boundary, and it is defined as

$$q_{r,t} = \frac{\sum_{S \in \mathcal{S}_{r,t}} \max_{\mathbf{P}} \mathcal{L}((S, \mathbf{P}) | \mathbf{X})}{\sum_{S \in \mathcal{S}_r} \max_{\mathbf{P}} \mathcal{L}((S, \mathbf{P}) | \mathbf{X})}, \quad (\text{E.1})$$

where \mathcal{S}_r is the set of all possible segmentation boundaries of $1, \dots, T$ in r segments, and $\mathcal{S}_{r,t}$ is the set of all possible segmentation boundaries in which t is present as a boundary. Therefore, $q_{r,t}$ indicates the total likelihood of segmentations that contain t as a boundary over the total likelihood of all possible segmentations.

Again we assume a fixed value of r in order to compute $q_{r,t}$. However, if we can identify sharp peaks in $q_{r,t}$, corresponding to the boundaries obtained by maximizing $\mathcal{L}(\mathcal{M} | \mathbf{X})$, we have that the data has in fact a good r -segmentation, and such segmentation is represented by \mathcal{M} . Hence, computing $q_{r,t}$ for different values of r , starting at $r = 1$ and incrementing it, until reaching a point where $q_{r,t}$ values have the desired structure is a heuristic to choose proper values of r .

In order to efficiently compute $q_{r,t}$, similarly to the computation of \mathcal{M} , we rely on dynamic programming. For $1 \leq a \leq b \leq T$ and $l \leq b - a + 1$, define

$$\mathcal{U}_l(a, b) = \sum_{\{s_0, \dots, s_l\} \in \mathcal{S}_r[a, b]} \prod_{j=1}^l U(s_{j-1}, s_j - 1), \quad (\text{E.2})$$

where $\mathcal{S}_r[a, b]$ is the set of all possible $l+1$ distinct segmentation boundaries of a, \dots, b in l segments with $s_0 = a$ and $s_l = b + 1$.

With the definition of \mathcal{U} we can rewrite $q_{r,t}$ as

$$q_{r,t} = \sum_{1 \leq l < r} \frac{\mathcal{U}_l(1,t)\mathcal{U}_{r-l}(t+1,T)}{\mathcal{U}_r(1,T)}. \quad (\text{E.3})$$

The advantage of rewriting $q_{r,t}$ as in Equation (E.3) is that $\mathcal{U}_l(1,t)$ and $\mathcal{U}_l(t,T)$ can be efficiently computed by using their recursive nature combined with dynamic programming. More specifically, we have

$$\mathcal{U}_i(1,b) = \begin{cases} \sum_{i \leq a \leq b} \mathcal{U}_{i-1}(1,a-1)U(a,b), & 2 \leq i \leq b \\ U(1,b), & i = 1, \end{cases} \quad (\text{E.4})$$

and

$$\mathcal{U}_i(a,T) = \begin{cases} \sum_{a \leq b \leq T-i+1} \mathcal{U}_{i-1}(b+1,T)U(a,b), & 2 \leq i \leq a \\ U(a,T), & i = 1. \end{cases} \quad (\text{E.5})$$

The total cost to compute $q_{r,t}$ for a given r and $1 \leq t \leq T$ is $O(T^2r + T^2m)$. In details:

- $O(T^2m)$ to compute $U(1,a)$ and $U(b,T)$ for $1 \leq a \leq T$ and $1 \leq b \leq T$ (using similar strategy as used in Appendix D to compute $U(a,b)$ for all pairs a, b with $a \leq b$); and
- $O(T^2r)$ to fill tables $\mathcal{U}_i(1,b)$ and $\mathcal{U}_i(a,T)$.

Appendix F

Additional data cleaning

Initial segmentation analysis identified a group of ASes consistently appearing as unusual together (in a single group as output of Algorithm 3) between the end of 2006 and beginning of 2009. Membership in this group was subsequently identified to be a set of ASes in Australia with the property that every AS in the group obtained connectivity to the non-Australian Internet through the Australian Academic and Research Network (AARNet). Table F.1 presents the ASes forming this group.

Table F.1: ASes that we removed from segmentation analysis

AS Number	AS Name
AS7575	AARNET-AS-AP AARNet
AS38083	CURTIN-UNI-AS-AP Curtin University
AS7476	QUESTNET-ACADEMIC-AP The University of Queensland
AS24433	CQU-AS-AP Central Queensland University
AS24434	JCU-AS-AP James Cook University
AS24437	UWA-AS-AP University of Western Australia
AS4822	NATIONAL-LIBRARY-AU National Library of Australia
AS10148	UNIMELB-AS-AP The University of Melbourne
AS7645	DEAKIN-AS-AP Deakin University
AS23719	USYD-AS-AP University of Sydney
AS18062	GRANGENET-AS-AP GRid And Next Generation NETwork
AS7573	UTAS The University of Tasmania
AS7572	AARNET-ACT-RNO AARNet
AS7570	AARNET-NSW-RNO AARNet
AS24390	USP-AS-AP University of the South Pacific
AS6262	CSIRO Commonwealth Scientific and Industrial
AS4738	AARNET-TEST-AS-AP AARNET
AS24101	UNE-AS-AP University of New England
AS1851	ADELAIDE-UNIVERSITY-AS-AP The University of Adelaide
AS23859	UNSW-AS-AP University of New South Wales
AS38474	ANTARCTIC-DIVISION-AS-AU Australian Antarctic Division

Because this group composition was not related to broad trends in the Internet evolution, but rather the specifics of Australian Internet connectivity, we removed the members of the group from the analysis shown in the dissertation. The removal was

necessary because such groups of ASes was significantly interfering with the results of the segmentation.

Appendix G

Proof of Proposition 2

For G_1 : the next-hop from any source towards any destination x is always x itself. Hence, between two destinations x and y , the next-hops from all sources will always differ, which implies $\text{RSD}(x, y) = 1$.

For G_2 : proof by induction on the length of the path between x and y . In the base case consider that x and y are adjacent nodes. Then for any z $N(z, x) = N(z, y)$, except when $z = x$ or $z = y$, where the equality never holds. So $\text{RSD}(x, y) = \frac{2}{|V|} = \frac{1 + \text{treeDist}(x, y)}{|V|}$. Assume that the proposition is true for any two nodes u and v which $\text{treeDist}(u, v) = l$. Take now x and y , such that $\text{treeDist}(x, y) = l + 1$, and let z be the adjacent neighbor of y in the shortest path from x to y . We have $\text{treeDist}(x, z) = l$, and hence $\text{RSD}(x, z) = \frac{1+l}{|V|}$. See now that if the next-hops towards x and z differ for $1 + l$ sources, then they will differ in $1 + (l + 1)$ for x and y , because of z . Hence, $\text{RSD}(x, y) = \frac{1+(l+1)}{|V|} = \frac{1 + \text{treeDist}(x, y)}{|V|}$.

References

- Ager, B., Chatzis, N., Feldmann, A., Sarrar, N., Uhlig, S., and Willinger, W. (2012). Anatomy of a Large European IXP. In *Proceedings of the ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, pages 163–174.
- Anwar, R., Niaz, H., Choffnes, D., Cunha, I., Gill, P., and Katz-Bassett, E. (2015). Investigating Interdomain Routing Policies in the Wild. In *Proceedings of the ACM Internet measurement conference*, pages 71–77.
- Arnbak, A. and Goldberg, S. (2015). Loopholes for Circumventing the Constitution: Unrestrained Bulk Surveillance on Americans by Collecting Network Traffic Abroad. *Michigan Telecommunications and Technology Law Review*, 21(2).
- Augustin, B., Krishnamurthy, B., and Willinger, W. (2009). IXPs: Mapped? In *Proceedings of the ACM Internet measurement conference*, pages 336–349.
- Bellman, R. (1952). On the Theory of Dynamic Programming. *Proceedings of the National Academy of Sciences*.
- Berry, M. W., Browne, M., Langville, A. N., Pauca, V. P., and Plemmons, R. J. (2007). Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155–173.
- Brito, S. H. B., Santos, M. A. S., Fontes, R. d. R., Perez, D. A. L., and Rothenberg, C. E. (2016). Dissecting the Largest National Ecosystem of Public Internet eXchange Points in Brazil. In *Proceedings of the Passive and Active Measurement Conference*, pages 333–345. Springer International Publishing.
- Broido, A. and claffy, k. (2001). Analysis of RouteViews BGP data: policy atoms. In *Network Resource Data Management Workshop*.
- Cai, X., Heidemann, J., Krishnamurthy, B., and Willinger, W. (2012). An organization-level view of the Internet and its implications (extended). Technical report, ISI-TR-2009-679, USC/ISI.
- Calder, M., Fan, X., Hu, Z., Katz-Bassett, E., Heidemann, J., and Govindan, R. (2013). Mapping the Expansion of Google’s Serving Infrastructure. In *Proceedings of the ACM Internet measurement conference*, pages 313–326.

- Calvert, K. I., Doar, M. B., and Zegura, E. W. (1997). Modeling Internet Topology. *IEEE Communications Magazine*, 35(6):160–163.
- Carmi, S., Havlin, S., Kirkpatrick, S., Shavitt, Y., and Shir, E. (2007). A model of Internet topology using k-shell decomposition. *Proceedings of the National Academy of Sciences*, 104(27):11150–11154.
- Cerf, L., Besson, J., Robardet, C., and Boulicaut, J.-F. (2009). Closed Patterns Meet n-ary Relations. *ACM Transactions on Knowledge Discovery from Data*, 3(1).
- Chatzis, N., Smaragdakis, G., Böttger, J., Krenc, T., and Feldmann, A. (2013). On the Benefits of Using a Large IXP As an Internet Vantage Point. In *Proceedings of the ACM Internet measurement conference*, pages 333–346.
- Chen, K., Choffnes, D. R., Potharaju, R., Chen, Y., Bustamante, F. E., Pei, D., and Zhao, Y. (2009). Where the Sidewalk Ends: Extending the Internet As Graph Using Traceroutes from P2P Users. In *Proceedings of the International Conference on Emerging Networking Experiments and Technologies*, pages 217–228.
- Chierichetti, F., Kumar, R., Pandey, S., and Vassilvitskii, S. (2010). Finding the Jaccard Median. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 293–311.
- Chiu, Y.-C., Schlinker, B., Radhakrishnan, A. B., Katz-Bassett, E., and Govindan, R. (2015). Are We One Hop Away from a Better Internet? In *Proceedings of the ACM Internet measurement conference*, pages 523–529. ACM.
- Cicalese, D., Auge, J., Joumblatt, D., Friedman, T. u., and Rossi, D. (2015a). Characterizing IPv4 Anycast Adoption and Deployment. In *Proceedings of the ACM Conference on emerging Networking EXperiments and Technologies*.
- Cicalese, D., Joumblatt, D., Rossi, D., Buob, M.-O., Augé, J., and Friedman, T. (2015b). A fistful of pings: Accurate and lightweight anycast enumeration and geolocation. In *Proceedings of the IEEE INFOCOM Conference on Computer Communications*.
- Cichocki, A., Zdunek, R., Phan, A. H., and Amari, S.-i. (2009). *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley Publishing.
- Comarela, G., Gürsun, G., and Crovella, M. (2013). Studying interdomain routing over long timescales. In *Proceedings of the ACM Internet measurement conference*, pages 227–234.

- Cunha, I., Marchetta, P., Calder, M., Chiu, Y.-C., Schlinker, B., Machado, B. V. A., Pescapè, A., Giotsas, V., Madhyastha, H. V. A., and Katz-Bassett, E. (2016). Sibyl: A Practical Internet Route Oracle. In *Proceedings of the Usenix Conference on Networked Systems Design and Implementation*, pages 325–344.
- Dhamdhere, A., Cherukuru, H., Dovrolis, C., and Claffy, K. (2012). Measuring the Evolution of Internet Peering Agreements. In *Proceedings of the International IFIP Conference on Networking*, pages 136–148.
- Dhamdhere, A. and Dovrolis, C. (2008). Ten years in the evolution of the Internet ecosystem. In *Proceedings of the ACM Internet measurement conference*, pages 183–196.
- Dhamdhere, A. and Dovrolis, C. (2010). The Internet is Flat: Modeling the Transition from a Transit Hierarchy to a Peering Mesh. In *Proceedings of the ACM Conference on emerging Networking EXperiments and Technologies*.
- Dhamdhere, A. and Dovrolis, C. (2011). Twelve Years in the Evolution of the Internet Ecosystem. *IEEE/ACM Transactions on Networking*, 19(5):1420–1433.
- Edwards, B., Hofmeyr, S. A., Stelle, G., and Forrest, S. (2012). Internet Topology over Time. *CoRR*, abs/1202.3993.
- Elmokashfi, A., Kvalbein, A., and Dovrolis, C. (2012). BGP Churn Evolution: A Perspective From the Core. *IEEE/ACM Transactions on Networking*, 20(2):571–584.
- Erdős, D. and Miettinen, P. (2013). Walk 'n' Merge: A Scalable Algorithm for Boolean Tensor Factorization. In *Proceedings of the International Conference on Data Mining*, pages 1037–1042.
- Faloutsos, M., Faloutsos, P., and Faloutsos, C. (1999). On Power-law Relationships of the Internet Topology. In *Proceedings of the ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, pages 251–262.
- Feldmann, A., Maennel, O., Mao, Z. M., Berger, A., and Maggs, B. (2004). Locating Internet Routing Instabilities. In *Proceedings of the ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, pages 205–218.
- Gao, L. (2001). On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Transactions on Networking*, 9(6):733–745.
- Gao, L. and Rexford, J. (2001). Stable Internet Routing Without Global Coordination. *IEEE/ACM Transactions on Networking*, 9(6):681–692.

- Gao, L. and Wang, F. (2002). The extent of AS path inflation by routing policies. In *Proceedings of the IEEE Global Communications Conference*, pages 2180–2184. IEEE.
- Gill, P., Arlitt, M., Li, Z., and Mahanti, A. (2008). The Flattening Internet Topology: Natural Evolution, Unsightly Barnacles or Contrived Collapse? In *Proceedings of the Passive and Active Measurement Conference*. Springer Berlin Heidelberg.
- Giotsas, V., Luckie, M., Huffaker, B., and claffy, k. (2014). Inferring Complex AS Relationships. In *Proceedings of the ACM Internet measurement conference*, pages 23–30.
- Giotsas, V., Luckie, M., Huffaker, B., and Claffy, K. (2015). IPv6 AS Relationships, Cliques, and Congruence. In *Proceedings of the Passive and Active Measurement Conference*, pages 111–122. Springer International Publishing.
- Glass, K., Colbaugh, R., and Planck, M. (2010). Automatically identifying the sources of large Internet events. In *IEEE International Conference on Intelligence and Security Informatics*, pages 108–113.
- Gregori, E., Improta, A., Lenzini, L., and Orsini, C. (2011). The Impact of IXPs on the AS-level Topology Structure of the Internet. *Computer Communications*, 34(1):68–82.
- Gregori, E., Improta, A., Lenzini, L., Rossi, L., and Sani, L. (2012). On the Incompleteness of the AS-level Graph: A Novel Methodology for BGP Route Collector Placement. In *Proceedings of the ACM Internet measurement conference*, pages 253–264.
- Gregori, E., Improta, A., Lenzini, L., Rossi, L., and Sani, L. (2015). A Novel Methodology to Address the Internet AS-level Data Incompleteness. *IEEE/ACM Transactions on Networking*, 23(4):1314–1327.
- Gregori, E., Lenzini, L., and Orsini, C. (2013). k-Dense communities in the Internet AS-level topology graph. *Computer Networks*, 57(1):213 – 227.
- Gürsun, G. (2013). *Inferring Hidden Features in the Internet*. PhD thesis, Boston University.
- Gürsun, G., Ruchansky, N., Terzi, E., and Crovella, M. (2012a). Inferring visibility: who’s (not) talking to whom? In *Proceedings of the ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, pages 151–162.
- Gürsun, G., Ruchansky, N., Terzi, E., and Crovella, M. (2012b). Routing state distance: a path-based metric for network analysis. In *Proceedings of the ACM Internet measurement conference*, pages 239–252.

- He, Y., Siganos, G., Faloutsos, M., and Krishnamurthy, S. (2009). Lord of the Links: A Framework for Discovering Missing Links in the Internet Topology. *IEEE/ACM Transactions on Networking*, 17(2):391–404.
- Javed, U., Cunha, I., Choffnes, D., Katz-Bassett, E., Anderson, T., and Krishnamurthy, A. (2013). PoiRoot: Investigating the Root Cause of Interdomain Path Changes. In *Proceedings of the ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, pages 183–194.
- Khan, A., Kwon, T., Kim, H.-c., and Choi, Y. (2013). AS-level Topology Collection Through Looking Glass Servers. In *Proceedings of the ACM Internet measurement conference*, pages 235–242.
- Khare, V., Ju, Q., and Zhang, B. (2012). Concurrent Prefix Hijacks: Occurrence and Impacts. In *Proceedings of the ACM Internet measurement conference*, pages 29–36.
- Kleinberg, J. and Tardos, E. (2005). *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Koivisto, M., Perola, M., Varilo, T., Hennah, W., Ekelund, J., Lukk, M., Peltonen, L., Ukkonen, E., and Mannila, H. (2003). An MDL Method for Finding Haplotype Blocks and for Estimating the Strength of Haplotype Block Boundaries. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 502–513.
- Kurose, J. F. and Ross, K. W. (2012). *Computer Networking: A Top-Down Approach (6th Edition)*. Pearson, 6th edition.
- Labovitz, C., Iekel-Johnson, S., McPherson, D., Oberheide, J., and Jahanian, F. (2010). Internet inter-domain traffic. *ACM SIGCOMM Computer Communication Review*, 41(4).
- Labovitz, C., Malan, G. R., and Jahanian, F. (1998). Internet routing instability. *IEEE/ACM Transactions on Networking*, 6(5):515–528.
- Lad, M., Massey, D., and Zhang, L. (2006). Visualizing Internet Routing Changes. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1450–1460.
- Lad, M., Zhang, L., and Massey, D. (2004). Link-Rank: A Graphical Tool for Capturing BGP Routing Dynamics. In *Proceedings of the Network Operations and Management Symposium*, pages 627–640.
- Li, J., Guidero, M., Wu, Z., Purpus, E., and Ehrenkranz, T. (2007). BGP routing dynamics revisited. *ACM SIGCOMM Computer Communication Review*, 37(2):5–16.

- Livadariu, I., Elmokashfi, A., and Dhamdhere, A. (2016). Characterizing IPv6 control and data plane stability. In *Proceedings of the IEEE INFOCOM Conference on Computer Communications*.
- Lodhi, A., Dhamdhere, A., and Dovrolis, C. (2012). GENESIS: An agent-based model of interdomain network formation, traffic flow and economics. In *Proceedings of the IEEE INFOCOM Conference on Computer Communications*, pages 1197–1205.
- Lodhi, A., Dhamdhere, A., and Dovrolis, C. (2014). Open peering by Internet transit providers: Peer preference or peer pressure? In *Proceedings of the IEEE INFOCOM Conference on Computer Communications*, pages 2562–2570.
- Luckie, M., Huffaker, B., Dhamdhere, A., Giotsas, V., and claffy, k. (2013). AS Relationships, Customer Cones, and Validation. In *Proceedings of the ACM Internet measurement conference*, pages 243–256.
- Mahadevan, P., Krioukov, D., Fomenkov, M., Dimitropoulos, X., claffy, k. c., and Vahdat, A. (2006). The Internet AS-level Topology: Three Data Sources and One Definitive Metric. *ACM SIGCOMM Computer Communication Review*, 36(1):17–26.
- Medina, A., Lakhina, A., Matta, I., and Byers, J. (2001). BRITE: An Approach to Universal Topology Generation. In *Proceedings of the International Symposium in Modeling, Analysis and Simulation of Computer and Telecommunication Systems*.
- Miettinen, P. (2011). Boolean Tensor Factorizations. In *Proceedings of the International Conference on Data Mining*, pages 447–456.
- Mühlbauer, W., Feldmann, A., Maennel, O., Roughan, M., and Uhlig, S. (2006). Building an AS-topology Model That Captures Route Diversity. *ACM SIGCOMM Computer Communication Review*, 36(4):195–206.
- Mühlbauer, W., Uhlig, S., Fu, B., Meulle, M., and Maennel, O. (2007). In Search for an Appropriate Granularity to Model Routing Policies. In *Proceedings of the ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, pages 145–156.
- Nomikos, G. and Dimitropoulos, X. (2016). traIXroute: Detecting IXPs in traceroute paths. In *Proceedings of the Passive and Active Measurement Conference*, pages 346–358. Springer International Publishing.
- Oliveira, R., Pei, D., Willinger, W., Zhang, B., and Zhang, L. (2010). The (in)Completeness of the Observed Internet AS-level Structure. *IEEE/ACM Transactions on Networking*, 18(1):109–122.

- Oliveira, R. V., Pei, D., Willinger, W., Zhang, B., and Zhang, L. (2008a). In Search of the Elusive Ground Truth: The Internet's As-level Connectivity Structure. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 217–228.
- Oliveira, R. V., Pei, D., Willinger, W., Zhang, B., and Zhang, L. (2008b). In Search of the Elusive Ground Truth: The Internet's As-level Connectivity Structure. *ACM SIGMETRICS Performance Evaluation Review*, 36(1):217–228.
- Orsini, C., King, A., Giordano, D., Giotsas, V., and Dainotti, A. (2016). BGPStream: A Software Framework for Live and Historical BGP Data Analysis. In *Proceedings of the ACM Internet measurement conference*, pages 429–444.
- Papadimitriou, D., Coras, F., and Cabellos, A. (2011). Path-vector routing stability analysis. *ACM SIGMETRICS Performance Evaluation Review*, 39(3):22–24.
- Rekhter, Y., Li, T., and Hares, S. (2006). A Border Gateway Protocol 4 (BGP-4). RFC 4271, RFC Editor. <http://www.rfc-editor.org/rfc/rfc4271.txt>.
- Rexford, J., Wang, J., Xiao, Z., and Zhang, Y. (2002). BGP Routing Stability of Popular Destinations. In *Proceedings of the ACM SIGCOMM Workshop on Internet measurement*, pages 197–202.
- Roughan, M., Willinger, W., Maennel, O., Perouli, D., and Bush, R. (2011). 10 lessons from 10 years of measuring and modeling the internet's autonomous systems. *IEEE Journal on Selected Areas in Communications*, 29(9):1810–1821.
- Shavitt, Y. and Weinsberg, U. (2012). Topological Trends of Internet Content Providers. In *Proceedings of the Workshop on Simplifying Complex Networks for Practitioners*, pages 13–18.
- Shi, X., Xiang, Y., Wang, Z., Yin, X., and Wu, J. (2012). Detecting Prefix Hijackings in the Internet with Argus. In *Proceedings of the ACM Internet measurement conference*, pages 15–28.
- Siganos, G., Tauro, S. L., and Faloutsos, M. (2006). Jellyfish: A conceptual model for the as Internet topology. *Journal of Communications and Networks*, 8(3):339–350.
- Singh, R. and Gill, P. (2016). PathCache: A Path Prediction Toolkit. In *Proceedings of the ACM SIGCOMM conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '16, pages 569–570, New York, NY, USA. ACM.
- Stewart, III, J. W. (1998). *BGP4: Inter-Domain Routing in the Internet*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

- Subramanian, L., Agarwal, S., Rexford, J., and Katz, R. H. (2002). Characterizing the Internet Hierarchy from Multiple Vantage Points. In *Proceedings of the IEEE INFOCOM Conference on Computer Communications*.
- Tanenbaum, A. S. and Wetherall, D. J. (2010). *Computer Networks*. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition.
- Terzi, E. (2006). *Problems and Algorithms for Sequence Segmentation*. PhD thesis, University of Helsinki.
- Wählich, M., Maennel, O., and Schmidt, T. C. (2012). Towards Detecting BGP Route Hijacking Using the RPKI. *ACM SIGCOMM Computer Communication Review*, 42(4):103–104.
- Wu, J., Mao, Z. M., Rexford, J., and Wang, J. (2005). Finding a Needle in a Haystack: Pinpointing Significant BGP Routing Changes in an IP Network. In *Proceedings of the Symposium on Networked Systems Design & Implementation*, pages 1–14.
- Xiang, Y., Yin, X., Wang, Z., and Wu, J. (2011). Internet Flattening: Monitoring and Analysis of Inter-Domain Routing. In *IEEE International Conference on Communications*, pages 1–6.

CURRICULUM VITAE

