MEASURING AND MODELING INTERNET ROUTING FOR REALISTIC SIMULATIONS

A Dissertation
Presented to
The Academic Faculty

By

Christos Xenofontas Dimitropoulos

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in
Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
August 2006

MEASURING AND MODELING INTERNET ROUTING FOR REALISTIC SIMULATIONS

Approved by:

Dr. George Riley, Advisor
Asst. Professor, School of ECE
Georgia Institute of Technology

Dr. John Copeland
Professor, School of ECE
Georgia Institute of Technology

Dr. Henry Owen
Professor, School of ECE
Georgia Institute of Technology

Dr. KC Claffy
Adj. Professor, School of CS
University of California, San Diego

Dr. Doug Blough
Professor, School of ECE
Georgia Institute of Technology

Dr. Nick Feamster
Asst. Professor, School of CS
Georgia Institute of Technology

Date Approved: April 2006

To my family

ACKNOWLEDGMENTS

I want to thank my advisor George Riley for his consistent help throughout the PhD period. George has been a great advisor. He pointed me to the correct directions and was a consistent source of support, intellect, and encouragement. Dmitri Krioukov has been a great teacher. He spent endless hours educating me and broadening my thinking horizons. Dmitri and KC Claffy inspired to me their pure love for scientific research. I am thankful to Nick Feamster for many insightful comments on earlier versions of this dissertations as well as for useful scientific discussions. I owe debt to my PhD committee members Henry Owen, Doug Blough, and John Copeland for helping me improve earlier versions of this dissertation. Spyros Denazis was a valuable source of advice and kindly hosted me in his laboratory during my visits to Greece. Besides, I want to thank my colleagues Patrick Verkaik, Bradley Huffaker, Ravi Sundaram, Andre Broido, Marina Fomenkova, and the other CAIDA elfs, with which I had valuable scientific discussions that helped me towards this dissertation.

I am grateful to Christina Routi for her love, care, and invaluable support throughout my graduate studies. I am lucky to have had good friends. Stathis Velenis, Simos Nikolaou, Bangelis Siminos, Antony Fornaro, Christina Vlachou, Nick Athanasaides, Nick Valantasis, and Harry Schnidermann were a great company both in the good and bad times of this journey.

Last but not least, I want to thank my family. I owe a lot to my parents, Antonios and Iro Dimitropoulou, who pushed me towards education and taught me to admire studying and learning new things. Finally, I want express my special thanks to my sister, Christiana Dimitropoulou, and my brother in law, John Catravas. The help and support I received from Christiana and John upon my arrival to Atlanta and through the course of my studies were invaluable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The Internet is composed of thousands of diverse networks that exchange routing information using the Border Gateway Protocol (BGP). BGP is one of the most critical protocols of the Internet, since it connects these diverse networks to enable communication between remote domains. Despite its critical nature, BGP suffers from a variety of serious problems, which have triggered substantial research on developing improved versions of BGP and new routing architectures.

In this dissertation, we introduce necessary tools and data-mining techniques for analyzing the present routing architecture and for evaluating new routing protocols. We focus on the problem of performing realistic BGP simulations and we first develop a BGP simulator enabling detailed and large-scale BGP simulations. Then, we introduce techniques to collect vital Internet routing data, which are essential in conducting realistic BGP simulations. Finally, we introduce models of the collected data.

Realistic simulations are necessary for analyzing large and complex systems like the inter-domain routing infrastructure. For this reason, we develop BGP++, a BGP simulation module optimized for performing detailed and large-scale BGP simulations. BGP++ is the outcome of integrating an open-source BGP router into a popular packet-level simulator. In particular, we integrate the Zebra BGP implementation into the popular ns-2 simulator, yielding a detailed BGP implementation. Then, to improve BGP++'s scalability, we introduce a compact routing table data structure, which enables us to simulate up to a few thousand BGP routers on a single workstation. In addition, we implement support for parallel and distributed BGP simulations, making possible the use of distributed clusters or supercomputer centers to realize larger experiments. Finally, we implement a user-friendly interface that uses a Cisco-like configuration language to configure simulated routers.

Having built our simulator, we focus on the problem of performing realistic BGP simulations that reflect actual trends and patterns present in Internet routing. To capture such

trends we introduce measurement and inference techniques and extract relevant Internet data. In particular, we introduce 1) a technique to measure the AS topology of the Internet, 2) a set of heuristics to infer business relationships between ASs, and 3) an algorithm to classify ASs into classes reflecting their network properties. For mapping the AS topology, we evaluate the utility of using BGP updates over the commonly used BGP tables. We find that because of short-lived BGP instabilities, BGP updates can reveal more topological data than BGP tables can. Our inference heuristics infer customer-to-provider (c2p), peer-to-peer (p2p), and sibling-to-sibling (s2s) AS relationships from publicly available BGP data. We validate our inferred AS relationships by conducting a survey with ASs' network administrators and demonstrate that they are highly accurate. Finally, we propose an AS taxonomy reflecting ASs with different network characteristics and introduce an algorithm for classifying ASs into appropriate classes based on a set of AS data. We validate the classifications of our algorithm and demonstrate that they are highly accurate. We also make publicly available a set of Internet data, which are essential in conducting realistic BGP simulations.

Finally, we introduce a topology generation framework. In contrast to previous approaches, our framework besides modeling a network topology it also models different link types. Link types reflect different link characteristics like bandwidth or latency that can be of interest in modeling network topologies. Though our framework is generic, we focus on modeling measured AS topologies with links of inferred c2p or p2p type. Thus, we derive an AS topology generator that besides generating realistic AS topologies it also assigns realistic c2p and p2p annotations. We analyze synthetic AS topologies produced with our generator and demonstrate that they effectively reproduce a series of topological characteristics found in measured AS topologies. In addition, we show that the c2p and p2p annotations in synthetic topologies are consistent with c2p and p2p patterns in the Internet.

Overall, this dissertation makes available novel tools, measurement and inference techniques, and vital Internet data that are tailored for conducting realistic BGP simulations.

We believe that these contributions will provide significant aid in the challenging problem of fixing the routing architecture of the Internet.

CHAPTER 1

INTRODUCTION

1.1 Overview of the Internet Routing

The Internet is a vast, distributed system composed of thousands of autonomous networks. These networks are called Autonomous Systems (ASs) and are spread in diverse geopolitical locations. An AS is typically an Internet Service Provider (ISP), an Internet eXchange Point (IXP), a company, or a university network. These networks interconnect in one or more peering points at which they exchange traffic. Packets traveling the Internet typically pass through multiple ASs. For this reason, ASs exchange routing information to effectively route packets to remote destinations. Routing information is exchanged using routing protocols, which are employed in a hierarchical architecture. The routing hierarchy has two layers, an intra-domain and an inter-domain layer.

The intra-domain layer refers to the network within an AS, in which an Interior Gateway Protocol (IGP) ensures that the routing tables of the internal routers are up-to-date and in sync. ASs have the flexibility to select any IGP protocol from a variety of different flavors like Open Shortest Path First (OSPF), Enhanced Interior Gateway Routing Protocol (EIGRP), or Intermediate System-to-Intermediate System (IS-IS). Intra-domain networks range in size from a few routers up to hundreds or even thousands of routers in large backbone providers. Though an IGP protocol keeps routing information for intra-domain destinations, it can not directly retrieve information for external destinations. The latter is accomplished via an External Gateway Protocol (EGP) that runs on the inter-domain layer. The inter-domain layer reflects the global network of ASs that is used to spread routing information for remote destinations. In this layer all the ASs run a single EGP protocol, BGP [3]. BGP is deployed on the border routers of an AS, which exchange routing information with other BGP routers in neighboring ASs or in the same AS. The BGP and IGP routers of an AS communicate to exchange information about internal and external

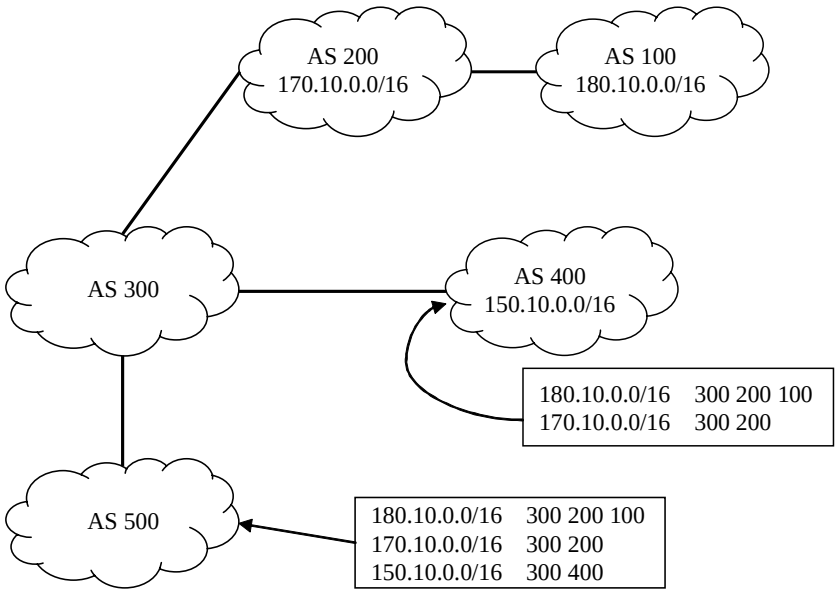


Figure 1. Simple AS topology demonstrating route announcements and AS paths received by ASs 400 and 500.

destinations.

BGP belongs to the class of path-vector routing protocols, which means that each route announcement carries a path that is used to reach the announced destination. In particular, BGP announcements, which are called BGP updates, carry an *AS path*, which is a list of ASs in the order that they will be transversed to reach the destination. For example, in Figure 1 we illustrate the route announcements and the included AS paths that ASs 400 and 500 would receive for the destination IP prefixes 180.10.0.0/16, 170.10.0.0/16, and 150.10.0.0/16. These prefixes are originated from ASs 100, 200, and 400, respectively. The AS 400 will receive route announcements with the AS paths 300 200 100 and 300 200 for the prefixes 180.10.0.0/16 and 170.10.0.0/16, respectively. On the other hand, AS 500 will receive route announcements with the AS paths 300 200 100, 300 200, and 300 400 for the prefixes 180.10.0.0/16, 170.10.0.0/16, and 150.10.0.0/16, respectively.

Besides being a regular routing protocol, BGP also helps in realizing an economy of ASs. AS links are the result of business agreements, thus they reflect not only traffic flows

but also money "flows". For example, small ASs pay larger ASs to connect to the Internet, whereas large ASs seek to attract customer ASs to increase their revenues. These business relationships influence how packets are routed between ASs. For example, ASs may wish to send packet via a certain upstream provider that is less expensive; small ASs typically avoid to transit traffic between their providers, since this would overwhelm their small capacity links. Thus, business relationships, or AS relationships as they are also known, have a direct impact on how BGP routers are configured and translate in technical specifications. AS relationships can be classified in three categories. In the customer-to-provider (c2p) category, a customer AS pays a provider AS for carrying traffic originated from the customer as well as for delivering traffic destined to the customer. In the peer-to-peer (p2p) category, two ASs exchange traffic between their customers but do not exchange traffic from or to their providers or peers[1]. p2p relationships are typically established between ASs of similar size and do not involve money exchange. Instead, two p2p ASs have mutual incentives to reduce their transit costs by establishing a p2p relationship. In a sibling-to-sibling (s2s) relationship, two ASs exchange traffic between their providers, customers, peers, or other siblings. Sibling ASs usually belong to the same organization or to strongly affiliated organizations. For example, the relationship between the European and North American divisions of a global ISP would be s2s. AS relationships result in restrictions on what AS paths can be used to route traffic between a source AS and a destination AS. In particular, a valid AS path must have the following hierarchical structure: an uphill segment of zero or more c2p or s2s links, followed by zero or one p2p link, followed by a downhill segment of zero or more provider-to-customer (p2c) or s2s links. The paths that follow this hierarchical structure are called *valley-free* or *valid* [4].

The inter-domain infrastructure of the Internet, its overlay AS topology, and AS relationships are of central interest in this dissertation.

[1]The meaning of the word "peer" is overloaded in the terminology used for BGP. In this dissertation we refer as peers ASs that have a p2p relationship. Moreover, we use the term "peering point" or "peering location" to refer to locations in which ASs exchange traffic, regardless of their AS relationship.

1.2 Inter-domain Routing Problems and Motivation

The inter-domain infrastructure suffers from a variety of problems. The spectrum of these problems is very wide. It starts from the prevalence of BGP misconfigurations [5] and expands to the lack of automatic problem diagnosis mechanisms, the absence of well-designed traffic engineering techniques, and the instability of the control plane [6, 7]. These inefficiencies require significant administration resources, which result in overwhelming economical overhead for the ISPs. Moreover BGP suffers from security problems [8], policy conflicts [9, 10], routing table growth [11], and path inflation [12, 13, 14].

These limitations are pronounced by the central role of BGP in the Internet architecture. BGP is the "glue" that connects the otherwise independent networks to yield a unified communication infrastructure. Thus, BGP problems can and have caused global connectivity disruptions. For example, in the AS 7007 incident [15], an AS accidentally generated BGP advertisements for most parts of the Internet, mis-leading several ASs to sending packets into a black hole. Moreover, in 2002, a BGP problem in Worldcom's network caused an 9-hour outage affecting a large fraction of the Internet's traffic [16].

For these reasons, BGP is often deemed insufficient, requiring major upgrades or even replacement. As a result, a number of fixes [17, 18, 19, 20, 21] and new routing architectures [22, 23, 24, 25, 26, 27] have been proposed. However, thus far none of these has gained enough recognition to be adopted. At the same time, the fast growth of the Internet and its evolution into a critical infrastructure press for comprehensive studies on how the protocol will meet the increased future demands.

The main reason the future of the routing architecture is not clear, is the lack of necessary tools 1) to comprehensively understand the current routing infrastructure and 2) to evaluate new alternatives.

First, given the currently available data, it is hard to develop a comprehensive understanding of the BGP infrastructure. Most ASs are commercial, self-managed domains that usually do not release to the public internal information about their networks. As a result,

access to BGP data is limited and vital information for monitoring, modeling, and evaluating BGP are not generally available. This limitation is further reinforced by the lack of provisioning for sufficient measurement and accounting protocols in the early days of the Internet. Thus, collecting representative data from the Internet is challenging yet essential for identifying BGP limitations and for modeling Internet routing.

Second, reliable performance evaluation tools are necessary for testing new routing protocols and modifications. Proposed fixes and architectures often cannot convince that if deployed they will attain the expected benefits and they will not trigger other unforeseen problems. Performance evaluation techniques range from purely mathematical analysis to laboratory testbeds, Internet-based testbeds, and simulation experiments. Mathematical analysis techniques suffer from the limitation that they typically rely on abstract, simplified formulations that cannot capture the complexity of BGP, which is an complex protocol documented in more than 30 Requests For Comments (RFCs). Laboratory testbeds are vital for small-scale testing but their results cannot pinpoint limitations that may arise in a real Internet-scale environment. As such, they can only be used as a first step in the performance evaluation process. Recently emerging Internet-based testbeds, like the Global Environment for Network Investigations (GENI) [28] initiative, provide the benefit of on-the-field testing for new clean slate architectures. However, they require extensive economical resources and need to meet the challenges of developing a truly representative environment that reflects the commercial nature of the Internet and of delivering a large-scale testbed unbiased from geopolitical restrictions. Simulation experiments have a low-cost barrier and enable large-scale experiments of thousands or even millions of Internet routers. On the other hand, they are conditioned to the availability of representative data and accurate models. Although these performance evaluation techniques are complementary and combined provide the means for better analysis of Internet protocols, in this dissertation we focus on simulation techniques since they have three key benefits: 1) flexibility, 2) low cost, and 3) ability to perform large-scale experiments.

In Figure 3, we illustrate the methodological requirements that we consider necessary for fixing the present routing infrastructure and for developing new routing architectures. At the root of the flow chart are the areas of collecting Internet data and of developing performance evaluation tools. Internet data make possible the development of better models and also enable us to identify problems and limitations of the routing architecture. Thus, they promote research on analyzing these problems, on identifying their root causes, and on redesigning routing so that it addresses these problems. New designs are then evaluated using realistic models and appropriate performance evaluation tools. High quality evaluation tools and models are required to assess the sustainability of new routing protocols and to detect unforeseen problems. Then, we can reject insufficient designs and find solutions that will truly lead to evolving and improving Internet routing.

In this dissertation, we first develop a BGP simulator tailored for performing detailed and large-scale BGP simulations (cf. Chapter 2). Then, we introduce measurement and inference techniques to mine vital BGP data (cf. Chapters 3, 4, and 5). Finally, we develop a framework for modeling the acquired data and for producing synthetic BGP data necessary for evaluation studies (cf. Chapter 6). In our road-map on Figure 3, our study focuses on the three highlighted problems on the left: collecting Internet data, modeling the collected data, and developing high quality evaluation tools.

1.3 Contributions

In this section, we summarize the contributions of this dissertation.

1.3.1 BGP Simulation

In Chapter 2 we introduce BGP++, a BGP simulation module capable of realizing detailed and large-scale experiments. We address the need for detail by incorporating Zebra bgpd [29], a popular UNIX-based BGP router, into the ns-2 packet-level simulator. Zebra bgpd provides a full-versed BGP implementation, which we combine with the packet-level

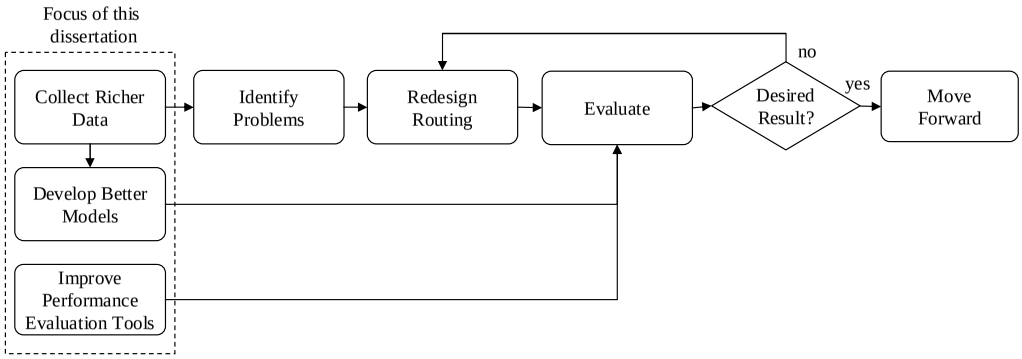


Figure 2. Methodological requirements for analyzing the present routing architecture and for developing appropriate fixes and new routing architectures. The dashed bounding box highlights the problems we focus on this dissertation. In our flow chart, better models can also be the input for improving performance evaluation tools. For example, we can improve the performance of a simulator by optimizing it for the target model. However, knowledge of the target model is not mandatory for developing better evaluation tools. For this reason, we do not have a link between the two bottom left boxes.

simulator to yield a precise BGP simulator. In addition, we implement a compact routing table data structure that results in significant memory savings for representing BGP routing tables. Using our data structure we demonstrate that we can effectively simulate up to 4,000 BGP routers on a workstation with 2Gbytes of physical memory. Then, we extend BGP++ to support parallel and distributed simulations by integrating it with pdns, i.e., the parallel and distributed version of ns-2. Thus, we enable the use of parallel and distributed computing platforms, which provide a wealth of computational resources for realizing large-scale simulations. Finally, we preserve the CISCO-like configuration language of the original Zebra bgpd, thus providing a user-friendly configuration interface for simulated BGP routers.

1.3.2 Measurements and Inference

We introduce methodologies to measure or infer the following central pieces of the inter-domain routing infrastructure:

1. **AS topology**

   We first focus on mapping the AS topology of the Internet, i.e., the graph in which nodes represent ASs and links represent interconnections between ASs. In Chapter 3, we evaluate the utility of AS paths that appear in BGP update messages as a source of AS-level topological data. We find that, because of short-lived BGP instabilities, update messages capture much more AS links than BGP tables, which are the typical source of AS topologies.

2. **AS relationships**

   In Chapter 4, we introduce heuristics to infer c2p, p2p, and s2s relationships. Our heuristics are based on a novel multiobjective optimization formulation of the inference problem. In contrast to previous heuristics, our method does not make the assumption that all or a maximal number of paths in the Internet follow the valley-free model. In contrast, we accept that in reality AS paths may deviate from the valley-free model because of misconfigurations and more complex AS relationships. We demonstrate that our formulation avoids shortcomings of the above assumption. In addition, we tackle the challenging problem of validating inferred AS relationships by conducting a survey with ASs' network administrators. We collect valuable data on actual relationships configured in the surveyed ASs and use this data to validate our inferences. We find that our heuristics accurately infer c2p, p2p, and s2s relationships.

3. **AS types**

   In Chapter 5, we introduce an AS taxonomy composed of the following classes: large ISPs, small ISPs, customer ASs, IXPs, Network Information Centers (NICs), and universities. Then, for each AS we collect the following attributes: advertised IP prefixes, AS adjacencies, and IRR description records. We build a training set of ASs, for which we manually identify their true class, and apply AdaBoost, a novel

machine-learning algorithm, on the attributes of the training ASs to find patterns associated with different AS classes. We express these patterns into a set of classification rules, which we apply to the full set of ASs to derive macroscopic statistics on the different types of ASs in the Internet. Finally, we validate our results, demonstrate that they are reasonably accurate, and develop a website [30] in which we make our AS classifications and AS attributes publicly available.

These three central components of the inter-domain infrastructure are directly related to conducting realistic BGP simulations. Knowing the AS topology of the Internet is necessary for using appropriate topologies in simulation experiments. Measured AS topologies can be incorporated in simulations directly or can form the basis for generating synthetic topologies.

Knowing AS relationships is necessary for simulating realistic routing over AS topologies. Routing in the Internet is subject to AS relationships and typically follows the valley-free model. Without knowledge of AS relationships, simulations must assume shortest-path routing, which is unrealistic.

AS types are mandatory for augmenting AS topologies with realistic intra-AS and inter-AS router-level topologies. For example, we expect the network of a dual-homed university to be drastically different from that of a dual-homed small company. The university will likely contain dozens of internal routers, thousands of hosts, and many other network elements (switches, servers, firewalls). On the other hand, the small company will most probably have a single router and a simple network topology. Since there is such a diversity among different network types, we cannot accurately augment the AS-level topology with appropriate router-level topologies if we cannot characterize the composing ASs.

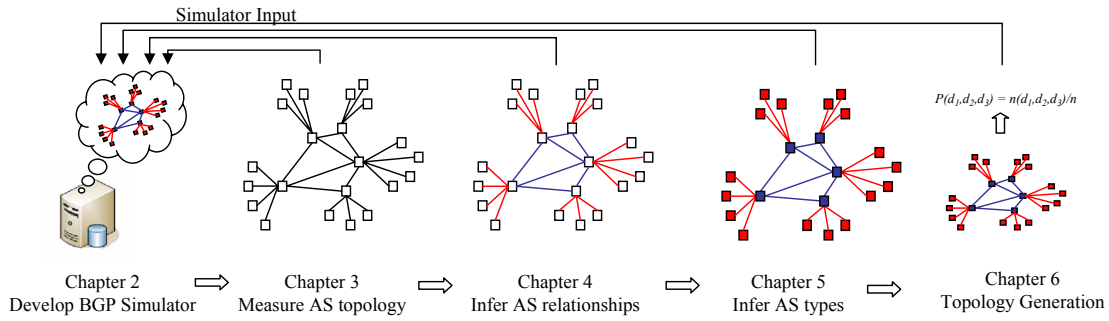


Figure 3. Structural and conceptual flow of the dissertation. In Chapter 2 we build a BGP simulator. In Chapter 3 we introduce a technique to measure the AS topology of the Internet. In Chapter 4 we augment measured topologies with inferred AS relationships, which we depict with link colors in this figure. In Chapter 5, we further augment measured topologies with inferred AS types, which we depict with node colors in this figure. Finally, in Chapter 6 we introduce a topology generator that generates synthetic AS topologies with realistic AS relationship assignments. The arrows on the top illustrate that the techniques we develop in Chapters 3 to 6 serve to provide realistic input data for our BGP simulator.

1.3.3 Topology Generation

In Chapter 6 we introduce a topology generation framework for generating annotated graphs. Annotated graphs are graphs with link annotations representing links with different characteristics. For example, one annotation may represent low-capacity links, while a second annotation may represent high-capacity links. Generally, link annotations provide the flexibility of encoding link characteristics of interest into graphs. We use our framework to generate AS topologies annotated with synthetic c2p and p2p relationships. We compare generated AS topologies with measured AS topologies annotated with inferred c2p and p2p annotations. Our comparison demonstrates that our framework effectively captures important properties of measured AS topologies. In addition, our synthetic annotations reproduce c2p and p2p patterns observed in the Internet.

1.4 Dissertation Organization

In Figure 3 we illustrate the structural and conceptual flow of this dissertation. First, in Chapter 2, we introduce BGP++, outline its development approach, and describe its features. In Chapters 3, 4, and 5 we focus on measurement and inference techniques. In

particular, in Chapter 3 we introduce a technique to measure the AS topology of the Internet. Then, in Chapter 4 we introduce an inference technique to augment measured AS topologies with inferred AS relationships. We conceptualize AS relationships and illustrate them in Figure 3 as link colors. In Chapter 5 we further augment AS topologies with inferred AS types, which we depict in Figure 3 with node colors. In Chapter 6 we introduce an topology generation framework for generating synthetic AS topologies annotated with synthetic AS relationship. Finally, in Chapter 7 we conclude this dissertation.

CHAPTER 2

BGP SIMULATION

2.1 Introduction

Modeling and simulation analysis has played a key role in the field of computer networks. Typically, vendors and researchers evaluate prospective architectures and perform comprehensive "what if" analysis using simulation. Simulation is also used for parameter-tuning, problem diagnosis and performance optimizations. It is immensely important in research on large, complex, and heterogeneous systems, like the Internet, where analytical models and laboratory testbeds do not capture the detail or the sheer volume of the system. One of these systems is the BGP infrastructure of the Internet.

BGP simulations and especially large-scale simulations are vital in analyzing reliability, scalability, performance and security problems that depend on the interactions of many routers. For example, large-scale BGP simulations are important in analyzing BGP convergence, since it is not clear how to extrapolate results, like the optimal value and setting of the MRAI timer [31], from small-scale experiments to an Internet-scale infrastructure. Large-scale BGP simulations are also important in analyzing large ISPs' networks. For example, ISPs often need to evaluate network design alternatives or the impact of scheduled internal or external connectivity or configuration changes, without having to actually change their networks. These tasks are possible using large-scale BGP simulations. Other possible applications include analyzing traffic engineering, sink-holes placement, evaluating reliability to failures or to flash crowds, evaluating attack mitigation techniques [32], optimizing iBGP design, analyzing interactions between underlay and overlay routing [33] or interactions between intelligent route controllers [34].

In this chapter we are concerned with building a simulation tool that will help researchers shed light on the design flaws of the current BGP infrastructure and evaluate the performance of new architectures. BGP++ is designed along the *scalability-realism*

tradeoff, enabling large-scale as well as detailed BGP simulations. BGP++ is not a new bottom-up implementation of BGP, but it capitalizes on existing high quality software on network simulation, routing, and parallel and distributed simulation. The following software are the basic components on which we build BGP++:

1. ns-2 [35] is a discrete-event network simulator that serves as a common platform on which researchers can test and compare their proposals. ns-2 has evolved into the most widely used simulator in the networking literature. It includes numerous implementations of protocols with a special emphasis on TCP variants.

2. GNU Zebra [29] is a detailed open-source implementation of BGP. It is used by a large community of providers, vendors and researchers for testing and experimenting with BGP. It is also used for routing by small ASs that cannot afford to buy expensive routers.

3. pdns [36] is the parallel and distributed version of the ns-2 simulator. It enables large-scale simulations by distributing the simulation model on multiple workstations, thereby granting more physical resources. pdns supports both shared-memory multiprocessors and distributed-memory clusters of workstations. Support of distributed architectures offers more physical resources, overwhelming the limits inherent in parallel architectures. Remarkably, the recent work by Fujimoto *et al.* [37] used pdns to claim the largest network simulations ever, of more than 5 million network nodes.

Our contributions can be summarized as follows:

1. We develop and make publicly available [38] a packet-level BGP simulation module for the widely-used ns-2 simulation platform.

2. We integrate the BGP implementation of Zebra into ns-2, making the minimum possible changes to the original software. We realize an accurate BGP simulator that

supports most of the details of Zebra's BGP implementation, including a Cisco-like configuration language.

3. We use and extend pdns to support parallel and distributed BGP simulations. Then, we evaluate the performance of alternative model partitioning algorithms for parallel and distributed BGP simulations.

4. We identify the representation of BGP routing tables as the main source of memory consumption in BGP simulations. To mitigate this problem, we introduce a compact routing table data structure that exploits the redundancy of information in BGP routing tables and results in up to 62% memory savings in our evaluation experiments.

5. We propose and develop a simple generic technique to speed up simulation trials using process checkpointing.

6. We develop a seamless partitioning and configuration engine for parallel and distributed BGP simulations that hides the complexity of pdns configuration and brings parallel and distributed simulation closer to the general ns-2 user. We also develop an automatic generator of Cisco-like configuration for setting common BGP policies on simulated routers. We combine these tools into a toolset that expedites BGP simulation practices.

The remainder of this chapter is organized as follows. Section 2.2 elaborates on previous efforts on BGP simulation. Section 2.3 describes in detail the development of BGP++. Section 2.4 introduces three techniques that we use to improve the scalability of the simulator. Section 2.5 introduces our toolset to expedite configuration, partitioning, and scheduling of BGP simulations. Finally, section 2.6 we conclude this chapter.

2.2 Related Work

The last few years there has been a considerable work on BGP simulation research. The BGP model by Premore in the SSFnet [39] simulator was the first detailed simulation model

of BGP and is currently the most widely-used BGP simulator. It supports most important BGP features and extensions. The main limitation of SSFnet is that it exhibits considerable memory demand, thereby preventing simulations larger than a few hundred of BGP routers.

Independently and in parallel with this work a number of other efforts have developed BGP simulators. Among these effors, the C-BGP [40] simulator and the work by Hao and Koppol [41] focus on large-scale BGP simulations. C-BGP is a BGP decision process simulator that, like BGP++, can read Cisco-like configuration files and can simulate large-scale topologies. Nevertheless, it only implements the BGP route selection process, ignoring several details of the protocol, namely timers and BGP messages.

The recent work by Hao and Koppol [41] addresses the challenge of large-scale BGP simulations by ignoring the protocol stack bellow the application layer. Their simulator can perform large-scale experiments. Nevertheless, the simulator is not available in the public domain and the relevant paper [41] does not discuss sufficiently the features of the simulator to develop a comprehensive picture of its capabilities.

Other efforts parallel or after BGP++ are [42, 43, 44] that ported the SSFnet BGP implementation in ns-2, JavaSim [45] and Genesis [46], respectively. In contrast to discussed works, BGP++ is optimized for performing detailed as well as scalable simulations.

2.3 BGP++: A BGP simulation module

In this section, we first outline challenges of integrating Zebra's software in the ns-2 network simulator and the techniques we used to address these challenges. Next, we discuss our experiments to validate that BGP++ provides an accurate implementation of BGP. Finally, we highlight features of BGP++'s configuration language, which enables the user to write configuration files using Cisco's well-known configuration language.

2.3.1 BGP++ development

The process of modeling a system is subject to the abstraction-scalability tradeoff. Higher abstraction results in simpler models, which therefore are more scalable. On the other hand,

detail is required to thoroughly capture the characteristics of a system. To create detailed simulation models we chose to incorporate the Zebra open-source BGP implementation into the ns-2 simulation environment. The main advantage of our approach is that the simulator inherits the detail, functionality, and maturity of the original open-source software.

Zebra is written in C and implements three routing protocols: BGP, RIP and OSPF (see Figure 4). Each of the protocols is a separate daemon that can be run as a stand-alone process. An additional daemon, called the Zebra daemon, takes care of communication between routing daemons and the kernel routing table or other routing daemons. This scheme provides a modular architecture with independent, well-separated implementations for each daemon. We use the BGP daemon (bgpd) and Zebra's library methods to build BGP++.

Throughout the integration process we take extra care to leave intact the fundamental logic of the code that implements BGP. However, Zebra and ns-2 are two intrinsically different software. The high level differences between the two software are the following:

1. ns-2 is written in C++ and makes extensive use of the object-oriented programming paradigm. Each network element is a C++ class, which enables the user to instantiate multiple objects of an element during a simulation. In contrast, Zebra is written in C and a single process instantiates a unique bgpd.

2. ns-2 uses discrete event scheduling algorithms, while Zebra uses UNIX process-based scheduling algorithms.

3. ns-2 has multiple TCP implementations, while Zebra uses the Berkeley Software Distribution (BSD) sockets TCP implementation.

4. ns-2 does not use blocking routines, while Zebra employs blocking routines.

These are the main differences we deal with during the integration process. In the following four steps we describe in detail how we merge Zebra bgpd and ns-2 to realize a unified, coherent system.

Step 1: The first difference between Zebra and ns-2 is that the former is designed to run one bgpd per process, while the latter needs to instantiate multiple BGP routers in the same process. To satisfy this requirement, we create a BGP class in ns-2 that incorporates all the original C code of Zebra bgpd. We convert Zebra's C functions into C++ member functions and Zebra's global variables into member variables of the BGP class. Thus, we can instantiate multiple bgpd in the simulator process as multiple objects of the BGP class.

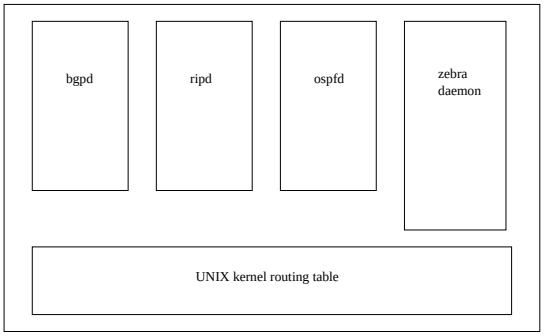


Figure 4. GNU Zebra modular architecture. This figure is a recreation of the prototype in [1].

Step 2: The most challenging aspect of the integration is interleaving Zebra bgpd scheduler with ns-2 scheduler. Discrete event simulators use queue-based schedulers. The entries of the queue are events that are sorted according to their time-stamp. On the other hand, system software has no standard scheduling architecture. Scheduling depends on the selected design and varies from simple to arbitrarily complex. Typical networking software has one or more blocking routines that blocks until an event triggers a response. To incorporate Zebra bgpd in the simulator, we modify the Zebra scheduler to communicate with the queue scheduler. At a high-level the interleaved scheduling works as follows: whenever there is an event for Zebra bgpd, e.g., the start event, the simulator gives control to the bgpd to execute the associated response. The bgpd continues until the first blocking routine is reached; then, instead of blocking, it returns control to the scheduler. Note, that the bgpd should not block since the simulation does not run on wall-clock time. The bgpd is given control again when the blocking routine would unblock. Events that could unblock the bgpd are read events, e.g., a packet arrival; write events, e.g., a buffer becomes writable;

timer expiration events, and user triggered events.

Zebra scheduling is based on the *select()* system call. *select()* takes as arguments a list of file descriptors and a timeout value. It blocks until a file descriptor changes status or until the timeout expires. The file descriptors indicate I/O streams, while the timeout value is set to the next timer expiration time. When *select()* unblocks, execution continues by handling the event or events that caused the interrupt. *select()* is reached again through an infinite loop.

Our interleaved scheduling works as follows: at the start event ns-2 calls the simulated bgpd to make the required initializations, i.e., to initialize BGP data structures and to start the BGP Finite State Machine (FSM). The simulated bgpd proceeds with executing code associated with the FSM until the blocking routine *select()* is reached. Then, instead of blocking, it enters an event for the calculated timeout value into the ns-2 queue scheduler and returns control to the simulator. If we disregard events that unblock the simulated bgpd, the latter will take control as soon as the timeout expires. However, *select()* could unblock before the timeout expires. For instance, Zebra bgpd *select()* unblocks when there is a read or write event [1]. For this reason, the simulator has to invoke the simulated bgpd upon a read or write event. In operating systems, read events occur when the TCP stream has new bytes available. In the simulator, upon a packet arrival, we modify ns-2 to cancel the future timeout event for the appropriate bgpd and give control to it. Data packets are handed to the bgpd by the simulator's underlying TCP implementation. In Zebra, write events result from the fact that non-blocking output routines, namely *write()* and *writev()*, do not copy the application buffer to the kernel output buffer immediately. If the kernel output buffer is full, the copy operation is postponed until the buffer becomes writable, i.e., write event. This time interval is typically very small and for this reason we ignore it and do not simulate write blocks.

A common simplification in network protocols modeling is the omission of the CPU

[1] User interrupts are treated as read events, since user communication is done through a telnet interface.

processing time, which is a valid assumption if the processing time is very small. BGP routers can exhibit long processing time, especially when their routing tables are large. For this reason, we implement a workload model that adds delay, representing the finite execution time of the CPU. We model the workload as follows: when a simulated bgpd completes an operation, such as parsing a packet that just arrived, it selects a *busy-period* time value. This time value represents the finite execution time of the operation that just completed. Then, it returns control to the simulator and for the next *busy-period* time it does not respond to any events, e.g., packet arrivals. Instead, all events are buffered and executed in FIFO order as soon as the bgpd resumes. Note that each of the buffered events will result in new *busy-period* intervals. We implement two workload models that differ in the way they choose the *busy-period* value. In the *uniform* workload model, the *busy-period* is a uniform random variable within a user specified range. In the *time-sample* model, the *busy-period* is the CPU time that was allocated for the operation that just completed by the workstation the simulation is running on. Using a kernel patch [47], BGP++ monitors the number of cycles its operations consume. Then, it calculates the *busy-period* as the product of the CPU clock frequency and the count of consumed cycles. This scheme can easily be extended to let the user specify the CPU clock frequency, enabling simulation of routers of different processing capacity.

Step 3: The third step is to substitute the BSD socket API with the corresponding TCP implementation of ns-2. We choose to use the *ns-2 FullTcp* implementation and modify *FullTcp* to simulate the socket API. Our modifications notify the application that started the *FullTcp* instance as soon as the connection moves from *SYN_RCVD* or *SYN_SENT* to *ESTABLISHED* and from *ESTABLISHED* to *CLOSE_WAIT*. The first two transitions correspond to the BSD sockets non-blocking *connect()* and *accept()*, respectively. In both cases they notify the application that the three way hand shake has completed successfully. The third transition notifies the application upon a passive connection termination.

Step 4: The last step to introduce Zebra routing software in *ns-2* simulation environment is to replace system calls with corresponding simulator functions, replace wall-clock time functions with simulation time functions, and remove unnecessary code. Zebra supports a telnet interface that is used to configure or query the routing daemons at run-time. We remove the telnet interface since it is not useful in a simulation environment. Moreover, we replace the functionality of the telnet interface with a new interface that enables us to query and reconfigure the simulated routers at run-time. We describe in more detailed our new interface in Section 2.3.3.

2.3.2 BGP++ validation and verification

According to [48], validation is the process to evaluate how accurate a model reflects a real-world phenomenon. In our case, instead of a real-world phenomenon we model BGP. However, we do not develop our BGP simulator from scratch, but we use pre-existing software. Thus, the validity of BGP++ is determined by the validity of Zebra's BGP implementation, the validity of ns-2 simulator and the validity of our integration methodology. Both Zebra and ns-2 open-source software have been used for a long time by the networking community. Also, our integration methodology replaces Zebra OS-related functions with corresponding ns-2 functions. Since both ns-2 and Zebra software have been widely-used, we argue that BGP++ provides an accurate implementation of BGP.

Furthermore, verification of BGP++ is required. Verification is the process of evaluating how faithfully the implementation of a model matches the developer's intent [48]. For BGP++, this definition translates to how accurately we implement our integration methodology. To verify BGP++, we develop several scenarios, ranging from simple to more complicated and perform simulations testing the behavior of BGP++. For each scenario we examine the results and make sure that the observed behavior agrees with the expected behavior. We develop test scenarios of the following types: basic behavior tests, policy related tests, logging facilities tests and advanced features tests. The following list enumerates the tested features:

- Basic behavior tests: connection establishment, session termination, connection reset, route distribution, route selection algorithm.

- Policy related tests: *route-maps*, *match* and *set* commands, *ip access-lists*, *ip community-lists*, *ip as-path access-lists*, *ip prefix lists*.

- Logging facilities tests: *show* command variants, binary dumps, debugging facilities.

- Advanced features tests: confederations, route-reflection, capability negotiation, soft reconfiguration, refresh capability.

We also perform additional tests to compare the behavior of BGP++ with the original unmodified Zebra software. For this purpose, we set up small testbeds of Zebra routers and compare the observed behavior with corresponding simulations of the same topology and configuration. The results confirm that BGP++ effectively and accurately models the behavior observed in the testbed environment.

2.3.3 BGP++ configuration

The development approach adapted for BGP++ preserves most of the features of the original software, among which the configuration language. Each simulated BGP router parses a configuration file, written in the configuration language used by Zebra routers. The Zebra configuration language is very similar to the well-known configuration language used by Cisco routers. For example, configuration structures like *route-maps*, *access-lists*, and *prefix-lists* are parsed by the simulated routers. The configuration of BGP++ is a two step process. First, the simulated topology has to be configured with an ns-2 *tcl* script. Then, each simulated router is configured using a distinct configuration file. The tcl script specifies a topology and associates BGP bgpd with nodes. A sample tcl configuration script is shown in Figure 5. To create a BGP router, it is necessary to instantiate a bgpd, attach it to a node, assign a configuration file, register the daemon, and set the finish time of the simulation. Registration is the process of inserting an entry in a global table that maps BGP

```
set ns [new Simulator]

set n1 [$ns node]
set n2 [$ns node]

$ns duplex−link $n1 $n2 1.5Mb 1ms DropTail

set r [new BgpRegistry]
set fin 400

set BGP1 [new Application/Route/Bgp]
$BGP1 register  $r
$BGP1 finish−time  $fin
$BGP1 config−file /sth/bgpd1.conf
$BGP1 attach−node $n1

set BGP2 [new Application/Route/Bgp]
$BGP2 register  $r
$BGP2 finish−time  $fin
$BGP2 config−file /sth/bgpd2.conf
$BGP2 attach−node $n2

$ns at $fin  halt

$ns run
```

Figure 5. Sample tcl script that creates two BGP daemons

instances to IP addresses. This table is called BGP registry and is required because BGP++ uses a dual addressing scheme, i.e., BGP routers are identified by both actual IP addresses and ns-2 addresses. Nevertheless, ns-2 addressing is invisible to the user. The second step of the configuration is to set up the simulated routers. Each router setup is done in a separate file using the same syntax used to configure Zebra bgpd. The configuration commands, such as those in the sample configuration in Figure 6, or slight variations thereof, are also used by many commercial BGP implementations. The following list identifies supported RFCs:

- RFC 1771, A Border Gateway Protocol 4
- RFC 1965, Autonomous System Confederations for BGP
- RFC 1997, BGP Communities Attribute
- RFC 2796, BGP Route Reflection

```
!Local AS and local IP address
router bgp 1
bgp router−id 192.38.14.1

!Neighbors
neighbor 192.38.14.2    remote−as 2

!Local networks
network  190.0.0.0   mask 255.0.0.0
network  189.0.0.0   mask 255.0.0.0

!Enable debugging
debug bgp
debug bgp fsm
debug bgp keepalives
debug bgp filters
debug bgp events
debug bgp updates

dump bgp all dump1.log
log file bgpd1.log
```

Figure 6. Sample BGP++ router configuration file

- RFC 2918, Route Refresh Capability for BGP-4

Finally, BGP++ provides support for run-time commands. Run time commands replace the functionality available in Zebra software through the telnet interface. The user can instruct a simulated router at a given point during a simulation to execute a specific command, like *show ip bgp*, which would otherwise be entered through the telnet interface. This is performed using a new tcl command. The tcl command takes as arguments the BGP configuration command to be called, the calling time, and the bgpd by which the command should be executed. Several BGP++ configuration examples as well as the full set of supported Cisco-like commands are available in BGP++ home page [38].

2.4 Scaling BGP++

Large-scale BGP simulations are one of the main concerns in BGP++ design. In this section we first show that the memory demand for BGP simulations is driven by the memory

required to represent routing tables. To address this problem we introduce a routing table representation data structure, which exploits the redundancy of routing table information across BGP instances. Our experiments indicate that our compact routing table data structure results in up to 62% memory reduction in the total memory required for the simulation. Using the compact routing table data structure we make proof-of-concept simulations of up to 4,000 simulated bgpd in a single workstation with 2GB of memory.

Next, we integrate BGP++ with pdns to make parallel and distributed BGP simulations possible and evaluate the performance of different model partitioning algorithms for parallel and distributed BGP simulations.

In Section 2.4.3 we introduce a simple and efficient technique to speed up execution time in simulation trials using process checkpointing. Although we implement our technique on BGP++ and evaluate it using BGP simulations, it is generally applicable to discrete event simulations.

our technique is generic, however we implement it on BGP++ and evaluate it using BGP simulations.

2.4.1 Efficient Routing Table Representation

The memory consumption of large-scale BGP simulations depends mainly on the following parameters: the size of the topology, the size of the routing tables, the number of neighbors per router, the simulation dynamics, and the simulator memory footprint. A BGP router maintains three Routing Information Bases (RIB): the Adj-RIB-in, the Loc-RIB, and the Adj-RIB-out [3]. The Adj-RIB-in stores routes received by neighboring BGP routers, the Loc-RIB stores routes selected by the decision process, and the Adj-RIB-out stores routes advertised to other BGP routers. Let N denote the number of BGP routers in a simulation; p the average number of prefixes originated per router; r the average number of neighbors per router; and α, β, γ, and δ proportionality constants. Then, the following formula denotes the total memory demand for a BGP simulation:

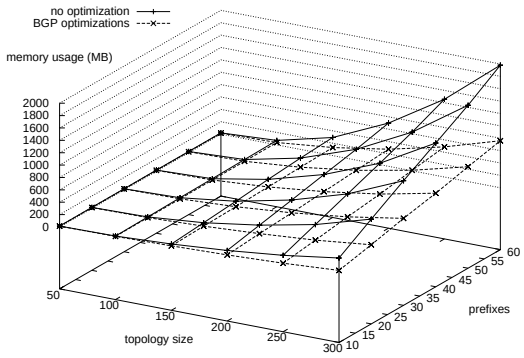


Figure 7. Memory usage versus topology size (N) and originated prefixes per router (p).

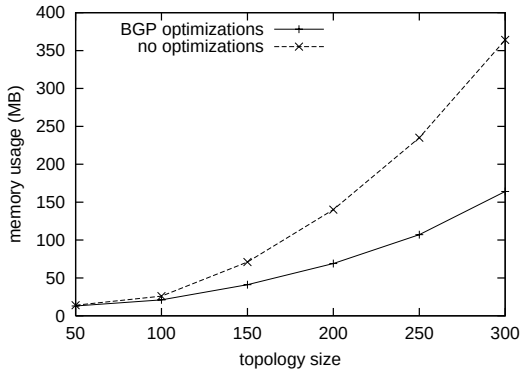


Figure 8. Projection of Figure 7 for 10 originated prefixes per router

$$Memory \approx \alpha N + \beta pN^2 + \gamma pN^2 r + \delta \quad (1)$$

The term αN represents the memory cost to initialize BGP routers or, in other words, the memory cost for routers with empty routing tables. The term βpN^2 accounts for the Loc-RIBs memory consumption, assuming that each router after convergence has pN entries in the routing table. The term $\gamma pN^2 r$ is the worst possible memory consumption for the Adj-RIB-ins and Adj-RIB-outs, assuming that each router receives pN routes from each of its r neighbors. The term δ represents the simulator footprint. It follows that for $r = N$, i.e., in a full-mesh topology, the memory demand is $O(N^3)$ and is driven by the routing table term $\gamma pN^2 r$.

Our compact routing table data structure relies on the observation that BGP routing

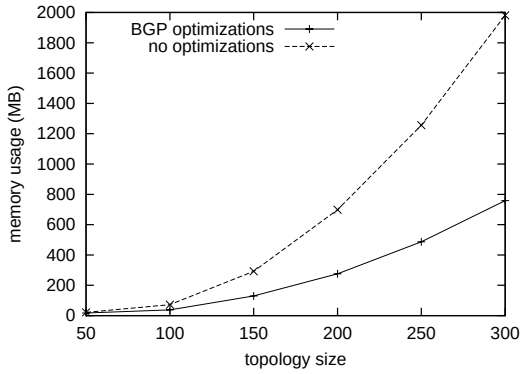


Figure 9. Projection of Figure 7 for 60 originated prefixes per router

tables contain a lot of redundant information. Redundant information lie across the following three dimensions: 1) within a routing table, across different entries; 2) within a router, across different routing tables (Adj-RIB-in, Loc-RIB, Adj-RIB-out); and 3) within a simulation, across different routers. For each routing table entry a BGP router stores several BGP attributes. For example, a routing table entry in BGP++ stores the following attributes: AS path, origin, next hop, local preference, multi-exit discriminator (MED), communities, extended communities, and unknown attributes[2]. These attributes account for most of the memory required for a table entry. For this reason, we create a data structure that shares common attributes among different routing tables.

We associate each route with a structure *struct attr* that has one entry for each of the possible attributes. For attributes with size less or equal to 32 bits the entry is the value of the attribute, otherwise a pointer to another data structure specific to the attribute in question. For example, *struct attr* contains the value of the MED attribute, which is a 32-bit unsigned integer. In contrast, it contains a pointer to an AS path data structure, which can be of arbitrary size. The AS path data structure is associated with a reference counter and stored in a global hash table. There is one global hash table for each type of attribute with size greater than 32 bits. A router, after allocating and initializing memory for a new attribute, it searches the relevant hash table for the newly allocated attribute. In case of a

[2]BGP specifications require to transit unknown attributes.

match, the allocated memory is deallocated and a reference to the attribute in the hash table is used. In addition, it increments the reference counter, which is used to track the number of pointers to the attribute. If the attribute is not found in the hash table, then the router creates a new entry for the attribute, so that subsequent searches will succeed. If a router wishes to remove an attribute, it decrements its reference counter. When the reference counter becomes zero the attribute is removed from the hash table and deallocated. This structure handles redundancy by creating a centralized pool of attributes, which are shared among different routers, tables and table entries. Similarly, a second level hash table is used to store and share *struct attr* structures.

We evaluate the effectiveness of our compact table data structure using simulation. For the simulation topology, we use a random connected subgraph of an Internet AS topology constructed from RouteViews data [49] of size N; each AS has a single BGP router that originates p prefixes. We infer c2p and p2p relationships between the simulated ASs and use BGP communities to configure them as follows: a provider advertises to a customer all the routes it knows; a customer advertises to a provider routes that are either locally originated or learned by its customers; likewise a peer advertises to a peer routes that are either locally originated or learned by its customers. These configurations are consistent with typical operator practices [50]. We set the delay and the data rate of all links to 10ms and 10Mbps, respectively and we ignore the processing workload of the routers. Finally, we run each simulation until the system reaches steady state, i.e., no updates are exchanged. For the remainder of this chapter we refer to this setup as *Internet*(N, p).

Figure 7 illustrates the total memory consumption with and without routing table optimizations versus N and p in an *Internet*(N, p) setup. Projections of this figure are shown in Figures 8 and 9 for 10 and 60 originated prefixes per router, respectively. We observe that our compact table data structure yields a consistent and significant reduction in the total memory consumption. The highest memory reduction achieved is 62% with a mean of 47% with respect to the total memory. Note that these numbers are conservative since

the total memory consumption is the result of several sources of memory demand besides the routing tables. The memory reduction with respect to the memory required for the unmodified routing tables is even larger.

To further improve the memory demand of BGP++, we employ an existing memory reduction scheme by Riley *et al*. [51], called Nix-Vector routing. Nix-Vector routing reduces the required memory to represent the Forwarding Information Bases (FIB) in a simulation by computing routes on demand, as needed, rather than pre-computing all possible routes. We integrate Nix-Vector routing with BGP++ and evaluate the memory we save by using Nix-Vector routing as compared to ns-2 default static routing. In Figure 10 we illustrate that in *Internet*$(N, 1)$ setups Nix-Vector routing can further reduce the memory consumption. The peak total memory reduction is 36% and the mean 22%. For $p > 1$ we find that the memory savings of Nix-Vector routing are overshadowed by the memory required to represent BGP routing tables.

Using our compact routing table data structures and Nix-Vector routing we conduct proof-of-concept large-scale BGP simulations. We effectively simulate up to 4,000 Zebra bgpds in a single workstation with 2GB of physical memory. The simulation setup is an *Internet*$(4000, 1)$. To our knowledge, the size of these simulations is an order of magnitude larger than the size of any other detailed simulation reported in previous works using simulation tools prior to BGP++.

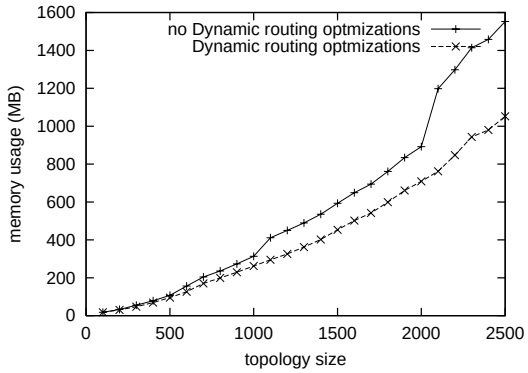


Figure 10. Memory savings by using Nix Vector routing

2.4.2 Parallel Distributed BGP Simulations

In this section we discuss the parallel and distributed version of BGP++ and evaluate different approaches to automatic partitioning of simulation models into multiple computing platform. Parallel and distributed computing is a promising venue to faster and larger network simulations. The ns-2 simulator has been extended by Riley *et al*. [36] to support parallel and distributed simulations. The parallel and distributed version of ns-2 is called pdns and enables substantially larger simulations than the original sequential version.

We extend pdns and ns-2 to support parallel and distributed BGP simulations. Our extensions use the RTIKIT [52] synchronization library to enable BGP routers residing on different simulator instances to establish BGP sessions and exchange BGP messages. Different simulator instances simulate different parts of the network topology. For this reason pdns belongs to the class of space-parallel simulators. A critical decision with space-parallel simulators is the distribution of the network topology model. This is because the distribution of the model has a direct impact on the performance of the sychronization protocol. To illustrate this consider how a distributed simulation works. First, the simulation is divided into epochs, which are separated by synchronization intervals. During a synchronization interval cross-simulator events are delivered. Let ET_{ij} and CT_{ij} denote the execution and communication time of the i-th simulator instance (or federate in pdns terminology) at epoch j, respectively. Also, let ST_j be the synchronization time at epoch j. Then, the execution time of the event processing phase of a simulation is:

$$ET = \sum_{0}^{N-1} (\max_{i} \{ET_{ij} + CT_{ij}\} + ST_j) \quad (2)$$

where N is the number of epochs [3]. Partitioning of the model has a direct impact on the terms ET_{ij}, CT_{ij}, ST_j, and N of equation 2. In particular, partitioning should:

- Balance the processing workload to minimize the maximum ET_{ij} over i.

[3] A similar formalization can be found at [53].

- Minimize the maximum communication load between any two federates. This has a direct impact on both CT_{ij} and ST_j since cross-simulator events are delivered during synchronization.

- Maximize the length of the epochs to increase parallelism and reduce N.

To automatically and efficiently partition a simulated topology we can use a graph G to represent it and associate weights with the nodes and the edges of the graph that reflect their expected computational and communication load, respectively. Then, we can formulate the topology partition problem as the following graph partitioning problem [54, 55]:

Given an undirected graph $G = (V, E)$, where V the set of vertices and E the set of edges, find a partition of G in k parts that minimizes the edge-cut under the constraint that the sum of the vertices' weights in each part is balanced.

In this formulation the edge-cut between two partitions reflects the communication load between two simulator instances. Moreover, the sum of the vertices' weights reflects the total computational load of a particular simulator instance. Thus, by mapping the topology distribution into this problem we can automatically partition the simulated topology so as to optimize equation 2. Nevertheless, this approach suffer from two limitations. First, the above graph partitioning problem is NP-complete. To overcome this difficulty we evaluate different polynomial time approximations that ship with the popular Chaco [56] and Metis [57] graph partitioning packages. Secondly, a method is necessary for assigning edge and vertex weights that reflect the expected communication and computation load, respectively. In general this problem is hard since its difficult to predict in advance of a simulation what will happen during the simulation. We address this problem by implementing two weight assignment heuristics that capture deterministic aspects of a simulation Both of our heuristics deem all the links of the topology as equal, and differ in how they model node weights. In the *degree-weight* approach the weight of node is set to its degree in the graph. This approach is based on the intuition that the computational load of a node

is proportional to the number of its neighbors. Our second heuristic assumes that all the partitions should have the same number of routers. For this reasons, it assigns the same weight to all the vertices of the graph. We call this heuristic *equal-weight* partitioning.

We first compare the performance of the following polynomial time approximations for the graph partitioning problem: Chaco multilevel Kernighan-Lin (KL), Chaco spectral, Chaco linear, METIS multilevel KL and Chaco random. In the last one, vertices are assigned randomly in partitions subject to the balance constraint, thus we use it as a worse case reference. To evaluate the partitioning algorithms we use a variant of the *Internet*$(N, 1)$ setup denoted as *Internet*$(N, 1, 200)$, in which the simulation time is fixed to 200 seconds[4]. We assign weights with the degree-weight and equal-weight approaches and partition the graph into two federates ($k = 2$). Figure 11 shows the execution time for the equal-weight case. We observe that the examined algorithms yield similar performance, while the savings with respect to the Chaco random algorithm increase with the model size. Though, the partitioning algorithms exhibit similar performance, we find that the METIS multilevel KL algorithm consistently outperforms the other algorithms both with degree-weight and equal-weight partitioning. Thus, we adapt the METIS multilevel KL algorithm for the remaining of our experiments and recommend it for future use in parallel and distributed BGP simulations.

Next, we compare the degree-weight to the equal-weight partitioning approaches. In Figures 12 and 13 we show our results for topologies partitioned in two federates. Figure 12 illustrates that the degree-weight approach results in worse performance than the equal-weight approach in an *Internet*$(N, 1, 200)$ setup. The same is also illustrated in Figure 13 for an *Internet*$(N, 0, 200)$ setup. We find that the degree-weight approach exhibits a worse performance because it results in a significant increase in the cross-federate communication volume. In Table 1 we show the edge-cut of the computed partitions for the degree-weight

[4]We choose to fix the simulation time, instead of using an *Internet*$(N, 1)$ setup, which runs for as many seconds as required for the routers to convergence. This is because in distributed simulations it requires substantial effort to automatically detect when the routers converge.

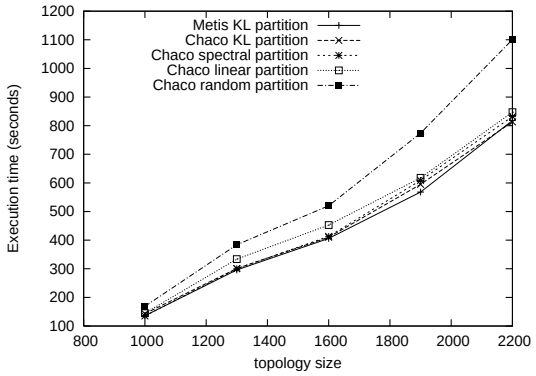


Figure 11. Total execution time using different partitioning algorithms.

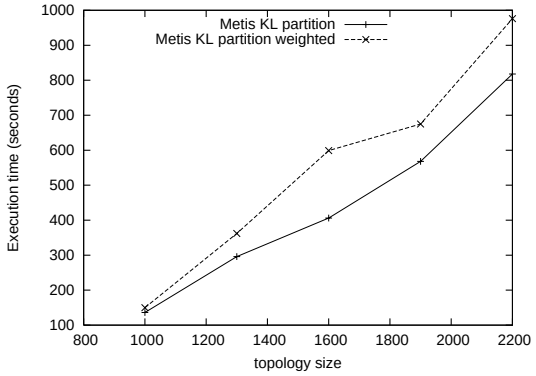


Figure 12. Total execution time of degree-weight versus equal-weight partitioning in an $Internet(N, 1, 200)$ **setup.**

and the equal-weight approaches. Clearly, the equal-weight approach results in much better partitions than the degree-weight approach. For this reason, we make the equal-weight heuristic the default choice for partitioning parallel and distributed BGP simulations.

Table 1. Edge-cut found by METIS multilevel KL algorithm for degree-weighted and equal-weighted graphs, split into two parts

Topology size	partition edge-cut	
	equal-weight	degree-weight
1000	33	119
1300	62	451
1600	90	442
1900	113	484
2200	175	719
2500	218	832

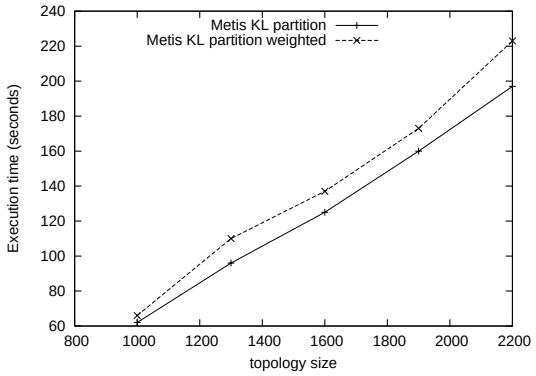


Figure 13. Total execution time of degree-weight versus equal-weight partitioning in an $Internet(N, 0, 200)$ **setup.**

2.4.3 Execution Time Optimizations

In this section we describe a simple and efficient technique to reduce simulation time in the presence of repeated simulation trials. Simulation trials are required to make simulation based results statistically significant. For example, assume a simulation in which we want to measure BGP convergence time after a BGP withdrawal. To make a reliable claim, it is necessary to repeat the same simulation many times and calculate the average over all runs.

Our execution time optimization exploits the fact that simulation trials repeat the same initial phase of a simulation, which in many cases is identical and can be avoided. In the most conservative scenario simulation trials repeat the same initialization process, which often can take significant time, before seeding their random number generator and entering the event processing phase. To avoid repeating the initialization, we can save an image of the simulation process just before the random number generator is used for the first time and start all subsequent trials from the saved image. In general this method can be used more aggressively. For example, in many cases we want to bring a simulation at a particular deterministic state, e.g., converged state, and then trigger an event and measure its impact. In this scenario, we can save an image of the simulation once it has reached the required state. Then we can start subsequent trials from the saved image, instead of running the full simulation from the beginning, reseed the random number generator and trigger the required event to measure its impact. If r is the ratio of the deterministic period

of a simulation to the total execution time and t the number of trials we want to perform, the speedup is[5]:

$$Speedup = \frac{ET_{normal}}{ET_{optimized}} = \frac{1}{1 - r + \frac{r}{t}}$$

The process of saving an image of a process is called checkpointing. We extend BGP++ to support process checkpointing using Condor [58], a workload management system that supports checkpointing and restarting a process. We implement a command to request a checkpoint of the simulator process as soon as the initialization has completed. The initialization phase includes construction of network objects, calculation of static routes (if needed), initialization of BGP data structures and parsing of BGP configuration files. We also implement a second command that enables the user to checkpoint the simulator process at any point during the simulation. In this case, the user has the option to change the configuration of the simulation just after the simulation process restarts. This way, multiple scenarios can be forked from the same image.

We examine the execution time savings of this approach in the conservative scenario of checkpointing an $Internet(N, 1)$ setup as soon as initialization has completed. In Figure 14 we depict the total execution time and the initialization time for different N. The mean r is 0.17. The total execution time depends on the dynamics of the BGP system, which in turn depend on the seed of the simulation. This variation explains why the total execution time curve is not smooth. In a second run, where the second optimization of section 2.4.1 is deployed, we find that r drops to 0.08, since the calculation of static routes during initialization is bypassed. Thus, we conclude that our checkpointing method can result in non-trivial savings even in the conservative scenario of checkpointing a simulation right after the end of the initialization phase. This method is expected to result in substantially higher executions time saving when used more aggressively, for example, in checkpointing a BGP simulation after the BGP routers have converged.

[5]It is assumed that the overhead to restart a simulation from a saved image is negligible.

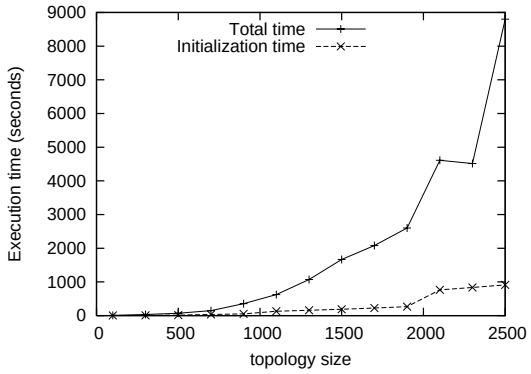


Figure 14. Total execution time and initialization time versus topology size.

2.5 BGP++ Configuration Toolset

We develop a toolset to expedite typical simulation configuration tasks. The simulation toolset can read simple user input files, generate Cisco-like configuration files, partition parallel and distributed simulations, generate configuration for pdns, and schedule multiple simulation runs. Figure 15 illustrates a block diagram of the software architecture of the toolset. There are three main components: a configuration generator, a partitioner and a scheduler.

The configuration component takes as input an AS topology that has been annotated with AS relationships. BGP communities and filters are used to configure typical BGP policies, like c2p and p2p relationships, employed between ASs. The user has the option to specify a variety of parameters like the topology size, the number of announced routes, the desired policy setup, and other.

The partitioner handles partitioning for parallel and distributed simulations. The user can choose among different partitioning algorithms supported through METIS and Chaco graph partitioning packages. Given a sequential configuration file for ns-2, it generates configuration for pdns and its BGP extensions. Thus, the partitioner provides a seamless interface to the parallel and distributed simulator, hiding from the user the complicated configuration of pdns.

Finally, the scheduler handles scheduling of multiple simulation runs. We implement a

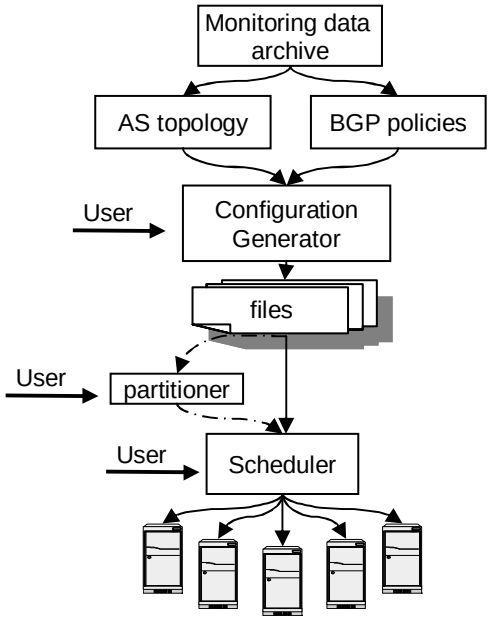


Figure 15. Block diagram of configuration toolset

master slave architecture, in which a master process schedules and distributes simulations to slave machines, taking into account available memory, existing CPU workload, and other user specified criteria.

2.6 Conclusion

In this chapter we introduced BGP++, a BGP simulation module aiming to help in simulation studies of the Internet routing architecture. BGP++ is built on top of high quality and popular software in network simulation (ns-2), routing (Zebra), and parallel and distributed simulation (pdns). The design goals of the simulator were to yield a detailed implementation of BGP that can also afford large-scale experiments. We met the first goal by incorporating the software of an open-source BGP router in a simulation environment. For the second goal, we introduced efficient memory management techniques and adapted BGP++ to support parallel and distributed simulations. We made several other contributions: 1) we introduced a generic framework to speed up simulation trials; 2) we developed an automatic topology partitioning and configuration toolset; and 3) we shared the lessons

we learned by integrating an open-source software in a network simulator.

We have recently ported BGP++ into the GTNetS [59] network simulator [60]. GTNetS is an emerging network simulator with a number of attractive features, which make it a more suitable platform for conducting BGP++ simulations than ns-2. In particular, GTNetS uses actual IP addresses for identifying simulated routers and hosts, as opposed to ns-2's use of scalar identifies. Thus, GTNetS reflects more accurately the Internet addressing reality and it integrates better with Zebra bgpd, which inherently employs IP addresses for routing and addressing. Moreover, GTNetS is built with scalability in mind and thus can deliver more scalable simulations than ns-2.

CHAPTER 3

AS TOPOLOGY

3.1 Introduction

In this chapter, we switch our attention to retrieving vital Internet topology data for conducting realistic BGP simulations. Measuring the Internet topology is important for evaluating and optimizing routing functions that depend on the structure of the underlying topology. For example, it is well known that the performance of BGP's convergence process depends strongly on the underlying topology [61, 31]. For this reason, BGP convergence optimizations need to be tested using topologies that reflect the true interconnections between Internet ASs.

We focus on the AS-level topology of the Internet, i.e., the graph in which nodes reflect ASs and links reflect interconnections between ASs. Presently, BGP tables are the main source of AS-level topological data, since they provide a large set of AS paths that are used for routing. Thus, one can construct an AS topology by merging AS paths from multiple BGP tables. Publicly available BGP tables are currently available from RouteViews [49], RIPE [62], CERNET [63], and from looking glass servers [64]. An alternative technique to map the AS topology of the Internet is to convert measured IP-level topologies to their AS-level counterparts. This mapping requires an accurate IP address to AS number translation, which is a topic of active research [65, 66]. AS topologies derived using this technique are available from CAIDA [67]. Another method to map the AS topology of the Internet is to exploit information in Internet Routing Registries (IRRs) [68]. The IRRs constitute a distributed database containing records of ASs' routing policies, assigned IP prefixes, adjacencies, contact information, etc. One can build an AS topology by finding AS adjacencies registered in the database. Nevertheless, IRRs are maintained in a volunteer basis and contain significant portions of incomplete or obsolete records. For this reason, this technique is not widely-used.

BGP tables bear a set of limitations in providing accurate and complete AS topologies. First, a BGP table reflects a single vantage point from which we can only capture a limited set of topological data. To illustrate this inefficiency, consider the extreme scenario of using shortest paths from a single vantage point to map a full mesh topology of size n. It is evident that shortest paths to every destination would only reveal $n-1$ of the total $n(n-1)/2$ links. Moreover, BGP policies limit the export and import of routes between ASs. In particular, prefixes learned over p2p links are not re-advertised to any providers, which means that a p2p link is not by default seen (as part of an AS path) from any upstream ASs. Thus, BGP policies further limit the topological information that can be extracted from BGP tables. A third limitation is that BGP table collectors, like the ones used by RouteViews and RIPE, use BGP sessions to collect routing tables. These BGP sessions filter out a large number of AS paths, since routers are restricted to sending their best paths to the collectors and cannot share their back-up paths.

In this work we explore an alternative method to construct representative AS topologies of the Internet. We evaluate the utility of using AS paths in BGP update messages over the traditional method of using AS paths in BGP tables. Our experiments demonstrate that BGP update messages reveal many AS links that are not seen in BGP tables. Using BGP updates, we build an AS topology that has 61.5% more AS links than the largest AS topology we constructed from BGP tables, collected from the same source as the BGP updates. Thus, we demonstrate that BGP updates are a viable alternative or additional source for measuring the AS topology of the Internet. Then, we analyze the temporal properties of the AS links seen in BGP updates and we find that a significant number of AS links can only be seen for short time intervals during BGP turbulence. Thus, it is practically hard to capture these links in BGP tables, which are archived at coarse time intervals. Finally, we compare the topological properties of the AS topology we constructed from BGP updates with an AS topology constructed from BGP tables. We find that the new AS links lie between small and medium degree ASs and that they have small betweenness,

pointing to links at the “periphery” of the AS topology.

3.2 Related Work

Zhang [69] *et al.* independently and in parallel with this work [70] evaluated BGP updates as a source of AS-level topological data and discovered that BGP updates reveal new AS links with respect to BGP tables. Their work differs in that they also used other data sources, i.e., the IRRs and the looking glass servers, in an attempt to construct the largest possible AS topology. In contrast, our goal is to evaluate the utility of using BGP update messages alone over the commonly used BGP tables. In addition, we also evaluate the temporal and topological properties of the newly discovered AS links. The work by Chang *et al*. [71] explored several diverse data sources, i.e., BGP tables, looking glass servers, and the IRRs, to create a more complete AS topology that had 40% more connections than a corresponding table-derived topology, however they did not examine BGP updates. Andersen *et al*. explored temporal properties of BGP updates to create a correlation graph of IP prefixes and identify clusters [72] . The clusters imply some topological proximity, however their study is not concerned with the AS-level topology, but rather with clustering prefixes using temporal correlations of updates.

3.3 BGP instabilities as a source of topological data

BGP is known to have a slow converge that is coupled with significant routing churn. In the event of a routing change the so-called *path exploration* problem [7] results in superfluous BGP updates, which advertise distinct backup AS paths of increasing length. Labovitz *et al*. [7] showed that in theory there can be up to $O(n!)$ superfluous updates during BGP convergence, where n is the number of ASs in the system. In this work we first seek to answer if these superfluous updates reveal new AS links that are not seen in AS paths extracted from BGP tables.

For example, observe the simple update sequence in Table 2 that we found in our

Table 2. Example of a simple sequence of BGP updates that unveils a backup AS link. The AS link 2828-14815 is not observed in BGP routing tables, but it is observed in BGP updates.

Time	AS-path	Prefix
2003-09-20 12:13:25	(withdrawal)	205.162.1/24
2003-09-20 12:13:55	10876-1239-2828-14815-14815-14815-14815-14815	205.162.1/24
2003-09-20 12:21:50	10876-1239-14815	205.162.1/24

dataset. The updates are received from a RouteViews neighbor in AS10876 and pertain to the same prefix. The neighbor initially sends a withdrawal for the prefix 205.162.1/24 and shortly after an update, in which the AS number 14815 has been prepended multiple times. AS path prepending is a technique used by ASs to artificially increase the length of an AS path. Using this technique they make a particular path less preferable to upstream ASs, since the AS path length is one of the main criteria used in BGP route selection. Thus, the long AS path prepending reveals thats the path 10876-1239-2828-14815-14815-14815-14815-14815 is a back-up path. More importantly, we find that the AS link 2828-14815 in this path is not observed in any other AS path in BGP tables taken from the same RouteViews router. Shortly after, the router receives a third update with a shorter AS path, in which it converges. Thus, this process illustrates that during BGP convergence, BGP update messages expose short-lived AS path that reveal new topological information. The short-lived nature of these AS paths is the reason that they cannot be captured in BGP tables snapshots, which are taken at coarse time intervals.

To measure the extent to which BGP instability can be useful for topology mapping, we collect BGP updates between September 2003 and August 2004 from the RouteViews router `route-views2.oregon-ix.net`. This router has multihop BGP sessions with 44 other routers and saves all received updates in the Multi-threaded Routing Toolkit (MRT) format [49]. After converting the updates to ASCII format, we parse the set of AS paths and mark the time each AS link was first observed, ignoring AS sets and private AS numbers. There are more than 875 million announcements and withdrawals, which yield an

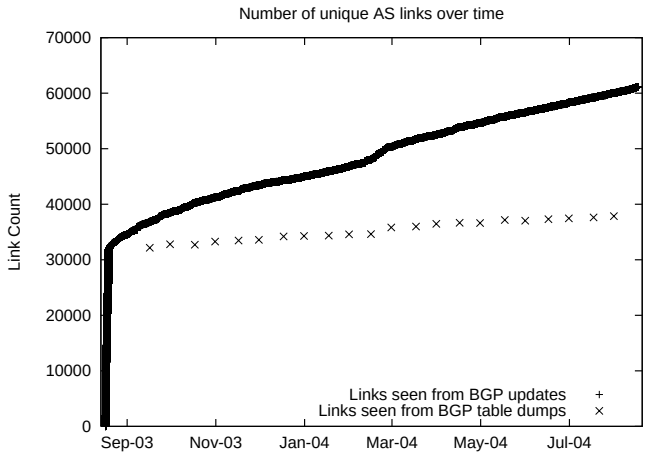


Figure 16. Number of unique AS links observed in BGP updates versus number of unique AS links observed in BGP tables.

AS topology of 61,134 AS links and 19,836 ASs. We denote this topology as G_u, where the subscript u marks that it was constructed from BGP updates. To quantify the extent of additional information gathered from updates, we collect BGP routing tables from the same router on the 1st and 15th of each month between September 2003 and August 2004. For each routing table dump we count the number of unique AS links, ignoring AS sets and private AS numbers for consistency. Figure 16 illustrates the comparison. The solid line plots the cumulative number of unique AS links over time seen in BGP updates. After an initial super-linear increase, the number of additional links grows linearly, much faster than the corresponding increase observed from BGP routing tables. At the end of the observation window, BGP updates have accumulated an AS topology that has 61.5% more links and 10.2% more nodes than the largest table-derived AS topology. The notable disparity suggests that the real Internet AS topology is different from what the widely-used table-derived AS topologies suggest. To evaluate the differences we conduct a detailed analysis of the temporal and topological properties of the observed connectivity.

3.4 Temporal Analysis

Identifying temporal properties of the AS connectivity observed from BGP updates is necessary to understand the interplay between the observation of AS links and BGP dynamics.

In addition, we want to compare the temporal properties of AS links present in BGP tables with AS links observed in BGP updates. To do so, we first introduce the concept of *visibility* of a link from a collector. We say that at any given point in time a link is visible if the collector has received at least one update announcing the link, and the link has not been withdrawn or replaced in a later update for the same prefix. A link stops being visible if all the prefix announcements carrying the link have been withdrawn or reannounced with new paths that do not use the link. We then define the following two metrics that capture temporal properties of AS links:

1. *Normalized Persistence* (NP) of a link is the cumulative time for which a link was visible in RouteViews, over the time period from the first time the link was seen to the end of the measurement period.

2. *Normalized Lifetime* (NL) of a link is the time period from the first time to the last time a link was seen, over the time period from the first time the link was seen to the end of the measurement period.

The NP statistic represents the cumulative time for which a link was visible in the collector, while the NL represents the span between the first and the last time a link was observed. Both are normalized over the time period from the first time a link was seen to the end of the measurement period to eliminate bias against links that were not seen from the beginning.

To calculate the NP and NL statistics, we replicate the dynamics of RouteViews routing table using the BGP updates dataset. We implement a simple BGP routing daemon that parses BGP updates and reconstructs the BGP routing table keeping per-peer and per-prefix state as needed. Then, for each link we create an array of time intervals for which the link was visible and calculate the NP and NL statistics. A limitation of the RouteViews data is that it does not include notification messages. As a results, we cannot explicitly pinpoint the event of a session reset between the RouteViews collector and one of its immediate

neighbors. Detecting such sessions resets is necessary for removing from the virtual table of our BGP daemon AS paths learned from the router with which the session was lost. To address this limitation we implement a heuristic to infer session resets.

The problem of detection of BGP session resets has been examined in previous research. Maennel *et al*. propose a heuristic to detect session resets between ASs in arbitrary Internet locations by monitoring BGP updates in RouteViews [73]. For our experiments, we are concerned with a less daunting task: we seek to detect session resets between a RouteViews collector and one of its immediate neighbors. Our inference heuristic is composed of two components. The first detects surges in the BGP updates received from a certain neighbor over a short time window of s seconds. In particular, if the number of prefixes updated from a neighbor in s is more than a significant portion of p percent of the previously known prefixes from the same neighbor, then a session reset is inferred.

The second component detects periods of significant inactivity when a threshold t is passed from otherwise active peers. We combine both approaches and set low thresholds ($t = 4mins$, $p = 80\%$, $s = 4secs$) to yield an aggressive session reset detection algorithm. Then, for testing we calculate NP and NL over a period of a month with and without aggressive session reset detection enabled. We find that the calculated statistics are virtually the same with less then 0.1% variation. This is because the short lifetime of session resets does not affect the span of the NP and NL statistics, which are calculated over a much longer period. For this reason, we omit the detection of session resets in further experiments since it barely affects the computed metrics.

We compute the NP and NL statistics over a 5-month period[1], from September 2003 to January 2004, and plot their distributions in Figures 17 and 18. Figure 17 demonstrates that NP identifies two strong modes in the visibility of AS links. At the lower end of the x axis, more than 5,000 thousand links have NP $\leqslant 0.2$, while at the upper end of the x axis,

[1]For the calculation of the NP and NL statistics we use a partial 5-month dataset instead of the 12-month dataset we used in Section 3.3. This is becasue it requires extensive time to parse the 12-month dataset of 875 million announcements.

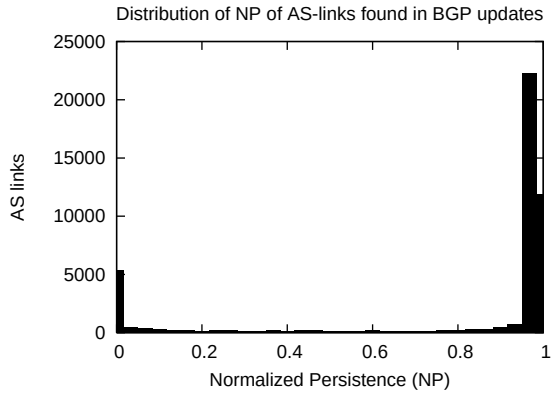


Figure 17. Distribution of Normalized Persistence of AS links seen between September 2003 and January 2004 in BGP updates.Normalized Persistence.

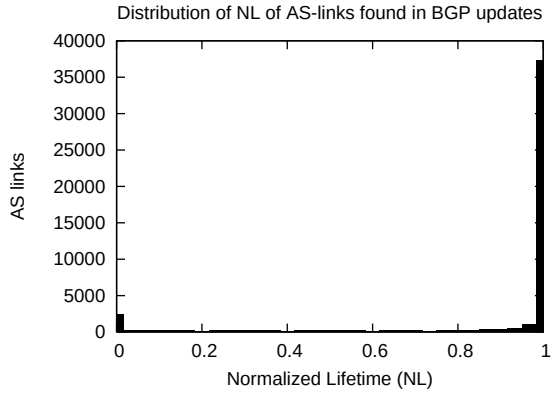


Figure 18. Distribution of Normalized Lifetime of AS links seen between September 2003 and January 2004 in BGP updates.Normalized Persistence.

almost 35,000 links have an NP close to one. These two modes demonstrate that while most links are visible for most of the time, a significant number of links appear *only* appear for short time intervals, i.e., during BGP convergence, and are typically not visible. The distribution in Figure 18 of the NL statistic is even more modal. Approximately 2,000 links have NL very close to zero, while all the remaining links have NL close to one. Since by definition the NL statistic of a particular link is greater than the NP statistic, we conclude that out of the 5,000 links with small NP, approximately 3,000 have NL close to one. This illustrates that more than half of the links that appear only during BGP convergence have a long lifetime. In other words, these links are consistently part of the AS topology even if we can rarely observe them.

Table 3. Normalized Persistence in G_1 and G_2.

	G_1	G_2
NP $\leqslant$ 0.2	65 (0.2%)	6891 (57.5%)
0.2 <NP< 0.8	1096 (3.2%)	1975 (16.5%)
NP $\geqslant$ 0.8	33141 (96.6%)	3119 (26.0%)

At the end of the 5-month period, BGP updates have accumulated a graph that we decompose into two parts. One subgraph, G_1, is the links in common with a topology extracted from a BGP table collected from RouteViews at the end of the 5-month period. The second subgraph G_2 is the graph remaining by removing the links in G_1. Table 3 shows the number of links with NP $\leqslant$ 0.2, 0.2 <NP< 0.8, and NP $\geqslant$ 0.8 in G_1 and G_2. Indeed, only 0.2% of the links in G_1 have NP $\leqslant$ 0.2, confirming that BGP tables capture only the AS connectivity seen at steady state. In contrast, most links in G_2 have NP $\leqslant$ 0.2, exhibiting that most additional links found in BGP updates appear only during BGP turbulence.

3.5 Topological Analysis

Ultimately, we want to know what new information we learn from BGP updates about the AS topology of the Internet, e.g., where the new links are located and how the properties of the graph change. A handful of graph-theoretic metrics have been used in the literature to evaluate the topological properties of the Internet. We choose to evaluate two representative metrics of important properties of the Internet topology:

1. *Joint Degree Distribution (JDD)*. The JDD $M(k_1, k_2)$ of a graph is another structural metric that gives the total number of links between k_1-and k_2-degree nodes. In other words, JDD describes the placement of links in a graph with respect to the degrees of their adjacent nodes.

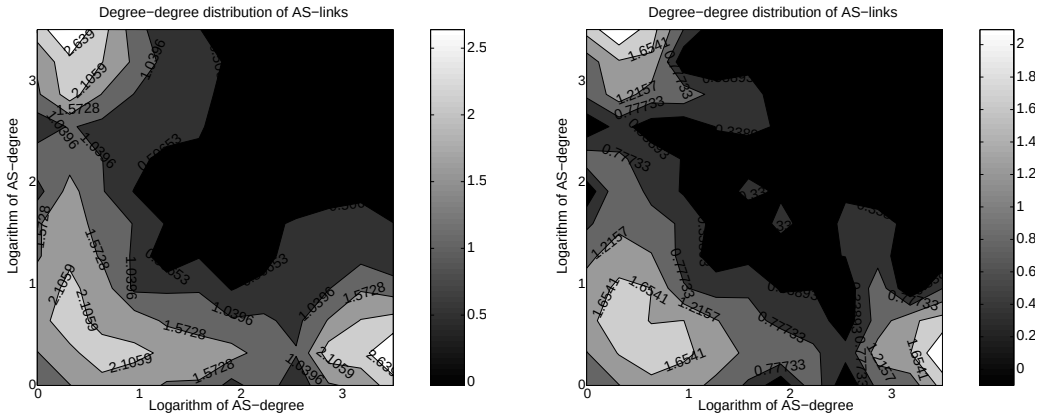
2. *Betweenness Distribution of AS links.* Given a graph $G(V, E)$, the betweenness $B(e)$ of link $e \in E$ is defined as:

$$B(e) = \sum_{ij \in V} \frac{\sigma_{ij}(e)}{\sigma_{ij}},$$

where $\sigma_{ij}(e)$ is the number of shortest paths between nodes i and j going through link e and σ_{ij} is the total number of shortest paths between i and j. With this definition, link betweenness is proportional to the traffic load on a given link under the assumptions of uniform traffic distribution and shortest-path routing.

We first examine the JDD distribution of the two graphs. Figure 19, compares the JDD of the links in G_u with the JDD of the links that are present in G_u but not in G_t. The overall structure of the two contourplots is similar, except for the differences in the areas of links connecting low-degree nodes to low-degree nodes and links connecting medium-degree nodes to medium-degree nodes (the bottom-left corner and the center of the contourplots). The absolute number of such links in $G_u - G_t$ is smaller than in G_u, since $G_u - G_t$ is a subgraph of G_u. However, the contours illustrate that the ratio of such links in $G_u - G_t$ to the total number of links in $G_u - G_t$ is higher than the corresponding ratio of links in G_u. Figure 20 depicts the contourplot of the ratio of the number of links in G_t over the number of links in G_u connecting ASs of corresponding degrees. The dark region between the 0.5 and 1.5 exponents, which correspond to AS degrees of 3 and 32 respectively, on the x and y axes highlights that BGP updates find more links between low and medium-degree ASs. Such ASs are typically small and reside in the "periphery" of the Internet AS topology. Thus, we find that the additional AS links we capture from BGP updates reside between small and medium size ASs.

Then we examine the link betweenness of the AS links. Figure 21 illustrates the betweenness distribution of G_u and G_t and reveals that our updates-constructed graph has more links with small betweenness. Links with small betweenness are links that do not carry substantial traffic in a uniform traffic communication model. These links are expected to have a low traffic load, which points to backup AS links and links used for local



(a) G_u (b) $G_u - G_t$

Figure 19. JDD. The x and y axes show the logarithms of the degrees of the nodes adjacent to a link. The color codes show the logarithm of the number of the links connecting ASs of corresponding degrees.

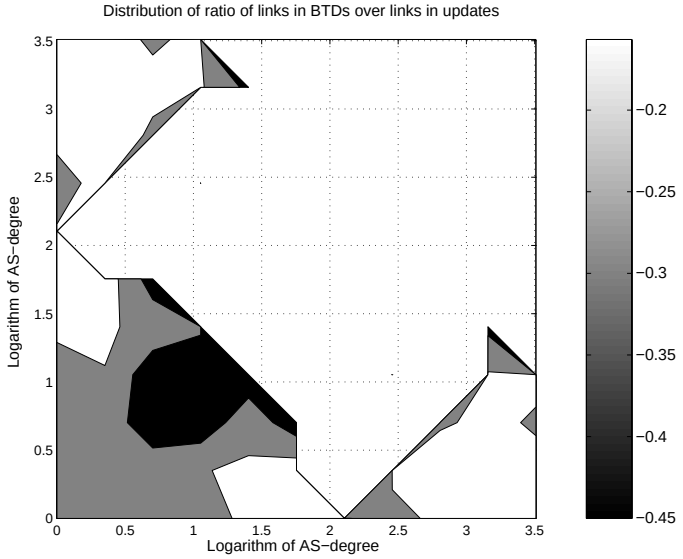


Figure 20. Distribution of the ratio of the number of links in G_t over the number of links in G_u connecting ASs of corresponding degrees. The x and y axes show the logarithms of the degrees of the nodes adjacent to a link. The color codes show the logarithm of the above ratio.

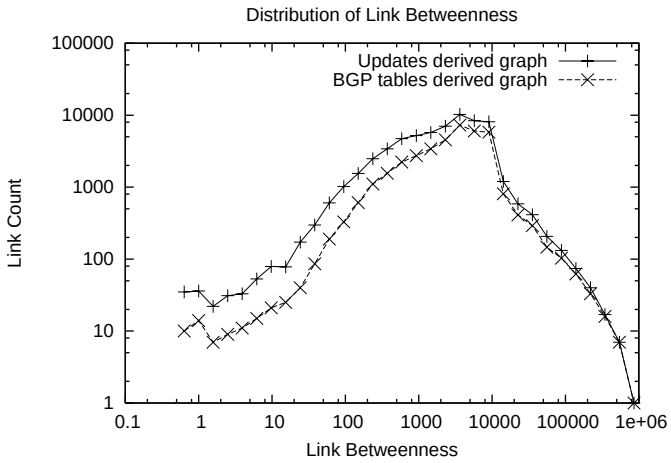


Figure 21. Distribution of the link betweenness of G_u compared to G_t.

communication in the periphery of the Internet.

Overall, our topological analysis shows that our more complete graph constructed from BGP updates has more links between low- and medium-degree nodes and more back-up links with respect to table-derived topologies.

3.6 Conclusion

In this work we exploited the previously unharnessed topological information that can be extracted from a well-known and easily accessible source of Internet routing data. We found that BGP updates reveal a significant portion of new topological data, which are not captured from the typically used BGP tables. We compared the temporal properties of AS links that appear in BGP tables with AS links that appear in BGP updates. We found that a large part of the AS links captured from BGP updates appear only for short time intervals during BGP convergence and that they are usually not captured from BGP tables. By accumulating BGP updates over a long time period we constructed an AS topology that is substantially larger than topologies constructed from BGP tables alone. We analyzed the more complete AS topology and showed that most of the new links discovered from BGP updates connect low- and medium-degree ASs, suggesting AS links used for backup purposes and local communication in the periphery of the Internet.

CHAPTER 4

AS RELATIONSHIPS

4.1 Introduction

The Internet is composed of thousands of ISPs that operate individual parts of the Internet infrastructure. ISPs engage in both formal and informal relationships to collectively and ubiquitously route traffic in the Internet. These relationships are usually realized in the form of business agreements that translate into engineering constraints on traffic flows within and across individual networks participating in the global Internet routing system. Accurate data on the structure of actual relationships among ASs is required for many research efforts concerned with performance, robustness, and evolution of the global Internet. Examples of both research and operational tasks that cannot neglect AS relationships include: 1) realistic simulations trying to model path inflation effects caused by routing policies; 2) understanding how packets are routed in the Internet and how to optimize Internet paths by analyzing existing deficiencies; 3) development of more scalable interdomain routing protocols and architectures, like HLP [74], that take into account the structure of AS relationships to optimize their performance; 4) evaluating how AS relationships affect the evolution of the Internet infrastructure and constructing economy-based models of the global Internet growth [75, 76]; 5) analysis of the spectrum of BGP configuration scenarios in order to develop more expressive routing protocols and configuration languages; 6) inference of AS paths in the Internet [77]; 7) modeling the structure of routing tables and developing synthetic routing tables needed for simulations of routing table lookup algorithms [78]; 8) development of better topology generators that account for the topological idiosyncrasies associated with AS relationships [79]; 9) selection of data centers for server replicas by measuring the origin of traffic to existing servers and evaluating connectivity and AS relationships of candidate data centers [80]; 10) selection of peers or upstream providers based on connectivity and AS relationships of candidate ISPs.

The first contribution of this work is development of new heuristics to infer AS relationships in the Internet using data from: 1) Border Gateway Protocol (BGP) routing tables, and 2) the Internet Routing Registries (IRRs) [68]. Our heuristics stem from identification and resolution of limitations in previous AS relationship inference efforts. The problem of inferring AS relationships was first introduced by Gao [4], who proposed an inference heuristic (LG) that identified top providers and p2p links based on AS degrees; and c2p and s2s links by assuming that every BGP path must comply with the valley-free model. This work was followed by Subramanian *et al*. (SARK)[1] [81], Di Battista *et al*. (DPP) [82], and Erlebach *et al*. (EHS) [83], who developed a consistent and elegant mathematical formulation of the inference problem (SARK) and found rigorous solutions (DPP, EHS) to the problem formulated by SARK. The latter solutions (DPP, EHS) both retreated to inferring only c2p relationships after demonstrating that it was in fact impossible to rigorously infer p2p relationships within the SARK problem formulation framework. In our work, we show that the DPP and EHS solutions frequently infer obviously incorrect relationships, e.g., ones with well-known tier-1 ISPs identified as customers of small ASs. We analyze and discuss the root causes of these limitations. Based on our findings, we propose new, improved heuristics that mitigate these problems and infer not only c2p relationships, but also p2p and s2s.

The second contribution of this work is a survey with organizations operating ASs and retrieval of valuable information on the types of AS relationships they have with other networks. We use this information to validate the AS relationships we infer. We find that our heuristics are accurate in inferring c2p (96.5% accuracy), p2p (82.8% accuracy), and s2s (90.3% accuracy) relationships. We also analyze the actual AS relationships that we learned from our survey and find that AS topologies constructed from BGP tables miss up to 86.2% of the adjacencies of the surveyed ASs. Moreover, BGP tables contain only 38.7% of the p2p adjacencies from the survey, while they capture 86.7% of the c2p adjacencies. To

[1]The acronyms indicated in parathesis are used to refer to the heuristic introduced in the corresponding study.

the best of our knowledge, our study is so far the most exhaustive AS relationship validation effort, and it is the first to use company-verified data to measure the completeness of AS topologies. Our findings should induce caution in the popular treatment of BGP-derived AS topologies as substantially complete or representative of the inter-AS connectivity structure of the Internet.

Finally, in pursuit of deeper understanding of real-world dynamics of the macroscopic Internet topology, we automate our heuristics and introduce an AS relationship repository where we archive inferred AS relationships on a weekly basis. We utilize our inferences to compute ranking of ASs based on inferred AS relationship hierarchies. We develop a web site [84] where we make our AS ranking and relationships publicly available.

In the next section, we introduce our heuristics. In section 4.3 we apply them to Internet data and compute the types of relationships in a snapshot of the AS topology. In section 4.4, we describe the results of our survey, validate our heuristics, and analyze the true AS relationships that we learned from the participating ASs. In section 4.5, we briefly discuss our ranking of ASs based on inferred AS relationship hierarchies, and introduce the web site where we make our ranking and relationship data publicly available. In section 4.6, we discuss previous approaches to inferring AS relationships and highlight our improvements. We conclude in section 4.7.

4.2 Inference Heuristics

4.2.1 [Mis-]inferring c2p relationships

SARK [81] cast the inference of AS relationships into the Type-of-Relationships (ToR) combinatorial optimization problem: given an undirected graph $G(V, E)$ derived from a set of BGP paths P, assign the edge type (c2p or p2p; s2s relationships are ignored) to every edge in G such that the total number of valid paths in P is maximized. The ToR objective of maximizing the number of valid paths implies that not all the paths in P are necessarily valid. Invalid paths may result from BGP misconfigurations or from BGP policies that are more complex and do not fall into the c2p/p2p/s2s classification. We call these paths

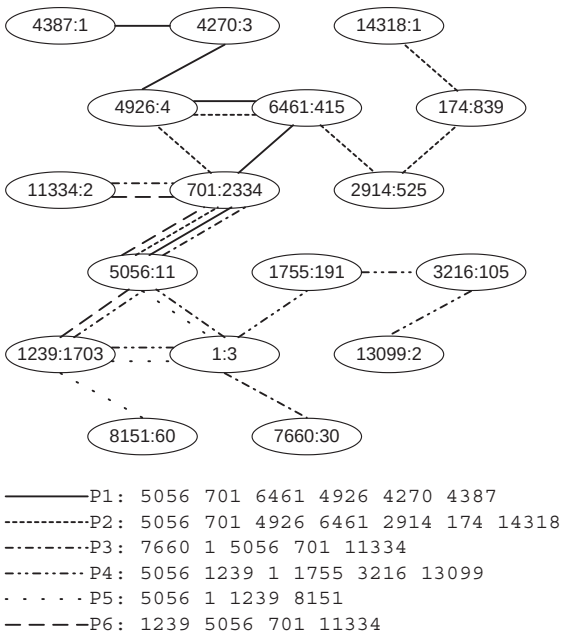


Figure 22. Instance of ToR problem that does not admit a solution. For each AS, we also note its AS degree as seen in our AS topology described in section 4.3.

anomalous. SARK speculated that ToR is NP-complete and developed a heuristic solution, later improved by DPP [82] and independently by EHS [83]. DPP and EHS proved that ToR is NP-complete, developed rigorous approximate solutions to the problem and proved that it is not possible to infer p2p relationships in the ToR formulation. For this reason, their solutions annotate all the edges in G as c2p, ignoring p2p and s2s relationships.

Although the ToR problem has been studied substantially, we find that it is subject to erroneous inferences. We demonstrate these limitations in the following three examples.

Example 1. In cases when there are multiple solutions with the same number of valid paths, ToR has no means to deterministically select the most realistic solution. Instead, it randomly selects one of the available solutions. Consider the real-world instance of the ToR problem in Figure 22, which was used in [82] to introduce ToR. We represent a possible solution to the ToR problem by introducing *orientation* to every edge *from* the customer AS *to* the provider AS. We can check that in the considered example, there are four distinct orientations of all edges, each maximizing the number of valid paths, but rendering one of

P1, P2, P4, or P5 as invalid. ToR has thus to arbitrarily select one of four possible solutions. Consider another example of the same type. Let path $p \in P$ be a sequence of edges $i_1, i_2, \ldots, i_{|p|-1}, j \in E$, i.e., $p = \{i_1 i_2 \ldots i_{|p|-1} j\}$. Suppose that the last edge j appears only in this one path p and that it is from a large provider (such as UUNET) to a small customer. Suppose that other edges $i_1, i_2, \ldots, i_{|p|-1}$ appear in several other paths and that they are correctly inferred as c2p. In this scenario both orientations of edge j (i.e. correct and incorrect: p2c and c2p) render path p valid. Edge j cannot thus receive a deterministic direction from ToR. We find many incorrect inferences of this type in our experiments. Similarly, in [85], several well-known large providers such as UUNET, AT&T, Sprint, Level3, are inferred as customers of smaller ASs such as AS1 (degree 3), AS2685 (2), AS8043 (1) and AS13649 (7), respectively.

Example 2. The AS degree sequences in the paths shown in Figure 22 reveal that path P6 is anomalous, since there is a small degree AS that appears to transit traffic between two large degree ASs. (AS 5056 with degree 11 appears to transit traffic between ASs 701, UUNET, and 1239, Sprint.) The ToR solution, however, will only have a single invalid path, one of P1, P2, P4, or P5, treating path P6 as valid. To make P6 valid, AS 5056 must be inferred as a provider of either UUNET or Sprint, or a provider of both, all of which is known to be incorrect. The key point is that while it is reasonable to assume that most AS paths in the Internet have a hierarchical structure, and that non-hierarchical AS paths are uncommon (and often due to misconfiguration), it is quite erroneous to believe that a solution maximizing the number of valid paths is correct.

Example 3. Another problem with SARK/DPP/EHS is that they ignore s2s relationships, a simplification that results in proliferation of erroneous inferences. Consider a path $p \in P$, $p = \{ij\}$, $i, j \in E$, and suppose that in reality i is a sibling edge that appears in multiple paths and that j is a c2p edge that appears only in path p. ToR solutions can classify edge i either as c2p or p2c depending on the structure of other paths containing it. If this structure results in directing i as p2c, then to make path p valid, the algorithm

erroneously directs edge j as p2c, too.

The above examples illustrate that: 1) trying to simply maximize the number of valid AS paths results in incorrect AS relationship inferences; 2) one should not treat all paths equally, regardless of the AS degree sequence in a path; 3) it is necessary to account for s2s relationships.

4.2.2 Improving the integrity of c2p inferences

In this subsection we show how we address these limitations and identify c2p relationships more accurately. Instead of solely relying on the valley-free model, we develop a heuristic that also exploits the structure of AS paths reflected by the degrees of the ASs.

More specifically, for every edge i in G we introduce a weight $f(d_i^-, d_i^+)$ that is a function of the degrees d_i^- and d_i^+ ($d_i^- \leqslant d_i^+$) of the ASs adjacent to the edge i. The weight f is large when there is a significant degree difference ($d_i^- \ll d_i^+$) between these neighboring ASs, and small otherwise. In directing the edges of the graph, we use f in the following way: when an edge i is directed from a small degree AS to a large degree AS, it earns a *bonus* b_i equal to $f(d_i^-, d_i^+)$, otherwise $b_i = 0$. We then formulate the inference problem as the following multiobjective optimization problem:

O_1 Maximize the number of valid paths in P;

O_2 Maximize the sum $\sum_{i \in E} b_i$.

This formulation has two features that build upon the strengths of previous work: 1) it respects the valley-free model and tries to maximize the number of valid paths in the input path set P, similar to SARK, DPP and EHS; 2) it uses the implicit knowledge embedded in AS degree information, similar to LG, to assign directions to edges along the node degree gradient, meaning that edge orientations collinear with this gradient are given certain weighted preference. In other words, the main idea behind our formulation is that while we still try to maximize the number of valid paths, we also use the AS degree information to detect paths that are anomalous and that we should not try to make valid. In contrast, the

ToR formulation solely tries to maximize the number of valid paths, treating all AS paths as equivalent, regardless of the AS degree sequences in the paths.

These two methodological objectives can be conflicting. The reason is that it is erroneous to assume that all or a maximal number of AS paths follow the valley-free model. To illustrate, consider the example in Figure 22. According to objective O_1, at least one of the edges 1239-5056 or 5056-701 in path P6 is erroneously directed against the node degree gradient in order to render P6 valid. By introducing O_2, we relax the first objective's requirement for a maximal number of valid paths. We can thus accept an invalid orientation for P6 based on the strong degree-gradient indication (O_2) that neither 1239 nor 701 are customers of 5056. In simpler words, comparing to the original ToR formulation, we relax the requirement to have a maximal number of valid paths and invoke additional information implicitly present in observed AS degrees in order to infer AS relationships more realistically.

To explore the tradeoff between the objectives O_1 and O_2, we introduce a parameter α and we weight O_1 by α and O_2 by $1-\alpha$. Then, by varying α in the region between 0 and 1, we control the relative preference of the two objectives.[2]

To solve the newly formulated problem, we map the c2p or p2c relationship of edge i to boolean variable x_i as follows: assuming an arbitrary initial direction of i, assignment of *true* to x_i means that edge i keeps its original direction, while assignment of *false* to x_i reverses the direction of i. We find assignments to variables x_i by reducing the multiobjective optimization problem to the MAX2SAT problem.

MAX2SAT is a boolean algebra problem: given a set of clauses with two boolean variables per clause $l_i \vee l_j$, find an assignment of values to variables maximizing the number of simultaneously satisfied clauses [87]. If the clauses are weighted, the problem is to maximize the sum of weights of the simultaneously satisfied clauses. MAX2SAT is

[2]In the terminology of multiobjective optimization [86], we consider the simplest scalar method of weighted sums.

NP-complete, however there exists a good approximation [88] that uses semidefinite programming (SDP) to deliver an approximation ratio of 0.94.

If we are concerned only with O_1, then we can use DPP or EHS'es reduction of ToR to MAX2SAT.[3] Using this reduction, we get a set of $x_i \vee x_j$ clauses, where $i, j \in E$.

To reduce O_2 to MAX2SAT, we introduce clause $x_i \vee x_i$ for every edge $i \in E$ that has an initial direction along the node degree gradient, and a clause $\bar{x_i} \vee \bar{x_i}$ for every edge with an initial direction against the node degree gradient. We thus ensure that if an edge is directed along the node degree gradient, then the corresponding clause is satisfied. To make our MAX2SAT instance equivalent to O_2, we weight every clause by $b_i = f(d_i^-, d_i^+)$.

We then reduce the resulting multiobjective optimization problem to MAX2SAT by refining the weights of the clauses. We weight the O_1 clauses by $w_{ij}(\alpha) = c_1\alpha$. The normalization coefficient c_1 is determined from the condition $\sum_{i \neq j} w_{ij}(\alpha) = \alpha \Rightarrow c_1 = 1/m_1$, where m_1 is the number of O_1 clauses. For the O_2 clauses, we use $w_{ii}(\alpha) = c_2(1 - \alpha) f(d_i^-, d_i^+)$, where the normalization coefficient c_2 is determined from the condition $\sum_i w_{ii}(\alpha) = 1 - \alpha$. In our notation, we fix the following weights:

$$w_{ij}(\alpha) = \begin{cases} c_1\alpha & \text{for } O_1 \text{ clauses,} \\ c_2(1-\alpha) f(d_i^+, d_i^-) & \text{for } O_2 \text{ clauses.} \end{cases} \tag{3}$$

With this selection of w_{ij}, we can vary α to explore the tradeoff between the two objectives and adjust it to the region or the point that results in the most accurate AS relationship inferences. We discuss the optimal value of α in section 4.3.3.

Function f injects dependence on AS degrees into our inference process. This function should take large values when its two degree arguments differ significantly, otherwise its values should be small. We note that a given difference in AS degrees is of different importance for small ASs and for large ASs. For example, a degree difference of 50 says more about the relative size of two ASs of degrees 1 and 51, than of 3000 and 3050. For

[3] For brevity, we do not discuss reduction of O_1 to MAX2SAT. See [82, 83] for details.

this reason, we normalize the degree difference in f to the *relative* node degree gradient $(d_i^+ - d_i^-)/(d_i^+ + d_i^-)$. In addition, topology graphs derived from BGP data provide only approximations of the true AS degrees and tend to underestimate degrees of small ASs. In contrast, they yield more accurate degree approximations for larger ASs [89]. To model this effect, we multiply the relative node degree gradient by a logarithmic factor reflecting our stronger confidence in accuracy of large AS degrees, compared to small ones. We thus fix f to:

$$f(d_i^+, d_i^-) = \frac{d_i^+ - d_i^-}{d_i^+ + d_i^-} \log(d_i^+ + d_i^-). \tag{4}$$

In summary, our formulation of the c2p relationship inference problem exploits the structure of the AS paths to address the limitations that we illustrated in Examples 1 and 2 of section 4.2.1. We resolve the limitation of Example 3 by independently inferring s2s relationships, as we explain in section 4.2.4. We then remove s2s edges from both graph G and path set P to avoid proliferation of incorrect c2p inferences. We solve the multiobjective optimization problem and provide a comprehensive evaluation of our c2p solution in section 4.3.3.

4.2.3 Inferring p2p relationships

The inference of p2p relationships is more challenging than the inference of c2p relationships. We first need to emphasize that as both DPP and EHS show, it is impossible to infer p2p relationships within the ToR problem formulation. Indeed, a valid path can have only one p2p link adjacent to the top provider in the path. If we replace this p2p link with a c2p or p2c link, the path remains valid, as it still has a valley-free, hierarchical structure. It follows, that by solving ToR, i.e., maximizing the number of valid paths, we cannot deterministically infer any p2p relationships at all. Xia and Gao [90] confirm this finding that LG and SARK's p2p inference heuristics yield low accuracy of, respectively, 49.08% and 24.63% of correct p2p inferences.

For the inference of p2p relationships in this work, we partially rely on LG and DPP

to develop an improved heuristic that combines LG and DPP's strengths. We start from a set of BGP paths P and extract graph G from it. Initially, we preprocess P to identify links that are not of p2p type (non-p2p). According to the valley-free model, any path can have at most one p2p link and this link must be adjacent to the top provider of the path. We thus parse all paths in P and deem all links that are not adjacent to the top provider in a path as non-p2p. We determine the top provider in a path as the highest degree AS in the path.

LG uses a similar but not identical approach. LG assumes that a p2p link can lie only between the highest degree AS and its highest degree neighbor. LG deems such links as p2p if the degree ratio between the two ASs is smaller than an external parameter, taken to have values of 60 or ∞. This method is aggressive in excluding non-p2p links. To illustrate, we consider an AS path A-B-C-D, where AS degrees are $d_A = 10$, $d_B = 500$, $d_C = 1000$, and $d_D = 501$. LG allows only link C-D to be a p2p link and deems the others as non-p2p. However, the degree difference between B and D is too small to make this judgment reliably. Our heuristic addresses this shortcoming by considering both B-C and C-D as candidate p2p links.

We denote by R the set of possible p2p edges constructed this way. We then introduce a weight $g(d_i^-, d_i^+)$ for every edge $i \in R$. Weight g is large when the ASs adjacent to edge i have similar degrees, and small otherwise. We thus encode our higher (lower) confidence that a couple of neighboring ASs are peers when their degrees are similar (dissimilar). We select weight g to complement weight f we used for the inference of c2p links:

$$g(d_i^-, d_i^+) = 1 - c_3 f(d_i^-, d_i^+), \tag{5}$$

where $c_3 = 1/\max_{i \in E} f(d_i^-, d_i^+)$ is a normalization coefficient.

Next, we remove from R any links left from the previous step that connect ASs with too large degree differences $d_i^- \ll d_i^+$. More specifically, we introduce threshold $w_e \in [0, 1]$ and remove every edge i with $g(d_i^-, d_i^+) < w_e$. The LG heuristic uses a similar threshold and empirically selects its value of 60 or ∞. Instead, to choose a proper value for w_e, we use information we learn from our survey. For each true p2p and c2p link present both in our

survey results and in R, we examine what selection of w_e leads to: 1) erroneously excluding a true p2p link from the set of possible p2p links R, meaning that $g(d_i^-, d_i^+) < w_e$ for a true p2p link; and 2) erroneously including a true c2p link in the set of possible p2p links R, meaning that $g(d_i^-, d_i^+) > w_e$ for a true c2p link. We find that the value of w_e that minimizes errors is $g(3, 545)$. The need for external threshold w_e is unfortunate, but the large degree difference between $d^- = 3$ and $d^+ = 545$ indicates that this threshold simply cleans R of links that are unlikely to be of p2p type.

At the last step of our p2p inference process, we examine paths in P with more than one edge from R. Such paths violate the valley-free model, and we select as the final set of p2p links a subset of R that does not contain links violating the valley-free model. DPP showed that the problem of finding a maximal set of p2p links that do not introduce invalid paths in P is equivalent to the Maximum Independent Set (MIS) problem. In the MIS problem, we are given a graph with nodes in N and arcs in A and we need to find the maximum subset of N such that no two nodes of the subset are joined by an arc in A. To make a more informed selection of p2p links, we utilize our weights g and turn the MIS problem into the Maximum Weight Independent Set (MWIS) problem. By solving MWIS, we find a maximal weight subset of R that does not introduce invalid paths into P. Selecting a maximal weight subset of R, we give preference to edges with large weights g because these edges are more likely to be of p2p type. We solve the NP-complete MWIS problem by means of a polynomial time approximation from [91]. Using the described approach, we remove from set R links that introduce invalid paths and obtain our final set F of p2p links.

4.2.4 Inferring s2s relationships

Sibling links connect ASs belonging to the same organization. Communication between s2s ASs is not subject to the export restrictions found in c2p and p2p relationships, e.g., rules such as “prefixes learned from a peer cannot be announced to other peers” do not apply for s2s ASs. The affiliated nature of such ASs enables them to implement much

more flexible and diverse policies. For this reason, it is difficult to infer s2s relationships from BGP data. Instead, we utilize the IRR databases to infer s2s links. Specifically, we track the organization to which each AS is registered in the databases and create groups of sibling ASs registered to the same organization. In several cases sibling ASs are registered to syntactically different organization names which still represent the same organization by other measures. For example, ASs 7018 and 3339 are registered to "AT&T WorldNet Services" and "AT&T Israel", respectively. To find such cases, we examine the organization names and create a dictionary of organization name synonyms. Then, we infer as s2s the ASs that are registered to the same organization name or to synonymous organization names. The strength of this approach is that it takes advantage of explicit information contained in the IRR databases. Although we realize the information of these databases is not always up-to-date or perfectly accurate, the organization names change less frequently than BGP policies and other more dynamic attributes. We can therefore treat the IRR databases as a source of publicly available information, which is reasonably accurate for the inference of s2s relationships.

4.2.5 Summary of inference heuristics

In summary, our inference heuristics take as input a set of BGP paths P and a corresponding graph $G(V, E)$ and proceed with the following three consecutive steps:

1. Use IRRs to infer s2s relationships and create set $S \subset E$ of s2s links;
2. Feed P, G, and S into the c2p heuristic assigning c2p/p2c relationships to the links in $E \setminus S$;
3. Use P and G to infer p2p relationships and to create set $F \subset E$ of p2p links.

The final answer is set S of s2s links, set $F \setminus S$ of p2p links, and set $E \setminus F \setminus S$ of c2p links.

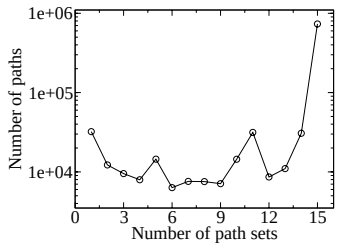


Figure 23. Persistency distribution of paths in $\cup_{k=1...15} P_k$.

Table 4. Number of paths, links and ASs in the stable path set versus averages of corresponding numbers over P_k, $k = 1 \ldots 15$

	Stable paths P (decrease)	Average over P_k
Paths	733,931(-12.49%)	838,734
Links	38,282(-4.34%)	40,023
ASs	18,943(-1.87%)	19,305

4.3 Applying Heuristics to the Data

4.3.1 Collecting and sanitizing the data

We first construct the input BGP path set P and the corresponding graph $G(V, E)$. Anomalous paths caused by BGP misconfigurations occur quite often and affect 200-1200 BGP table entries daily [5]. For this reason, we first sanitize the input data to mitigate the impact of misconfigurations.

We collected BGP tables from RouteViews [49], at 8-hour intervals, over the period from 03/01/2005 to 03/05/2005, for a total of 15 BGP table instances. After cleaning out AS prepending and AS sets, each BGP table yields a path set P_k. We define the *persistency* of a path $p \in \cup_{k=1}^{15} P_k$ as the number of sets P_k containing p. Figure 23 illustrates the persistency distribution of the the paths in $\cup_{k=1}^{15} P_k$. The distribution (note the log scale on y-axis) shows that although the majority of the paths appear in most of the 15 sets, a significant number of paths appear only in a few of the sets. Since BGP misconfigurations are temporal events, we select as input P to our algorithm the paths that appear in all of the 15 sets P_k. We call P the *stable path* set and the paths that are not selected the *unstable path* set.

In Figure 4 we compare the number of paths, links and ASs in the stable paths set with

Table 5. Number of unique degree-valleys and total number of degree-valleys found in stable and unstable path sets.

window w	Unique degree-valleys		Total degree-valleys	
	Stable paths	Unstable paths	Stable paths	Unstable paths
5	206	241 (+17%)	1290	2368 (+83.6%)
10	167	208 (+24.6%)	1178	2290 (+94.4%)
15	150	190 (+26.6%)	1135	2135 (+88.1%)
20	141	171 (+21.2%)	1119	1609 (+43.8%)

the corresponding averages over P_k. Even though P is 12.49% smaller than the average size of P_k, our filtering of unstable paths does not entail significant information loss in terms of number of links (4.34% reduction) and ASs (1.87% reduction). Next, we verify that unstable paths include more misconfigurations than stable paths. We call as *degree-valley* any AS sequence A-B-C with degrees d_A, d_B, and d_C, such that both d_A and d_C are larger than d_B summed with a small window constant w ($d_A, d_C > d_B + w$). The small window constant w helps to filter out trivial differences between d_B and d_A, d_C. Then, for both the stable and unstable path sets, we randomly select 10,000 paths and count the number of degree-valleys for different w. We repeat the random selection 100 times and show in Table 5 the average number of unique degree-valleys and the average of the total number of degree-valleys in the selected paths. The table illustrates that the number of degree-valleys in the unstable paths is up to 94.4% larger than the number of degree-valleys in the stable paths. This sharp increase of degree-valleys in the unstable paths indicates that unstable paths contain significantly more misconfigurations.

4.3.2 Inferring s2s relationships

For the inference of s2s relationships we use the RIPE, ARIN, and APNIC whois databases, collected on 06/10/2004.[4] We analyze the databases according to the methodology outlined in section 4.2.4 and find 1,943 organizations that own multiple AS numbers. We then examine the input graph G and discover 177 edges between ASs that belong to the same

[4]Since we extract from these databases the information that changes slowly with time, the date of the database dump is not critically important.

organization ($|S| = 177$).

4.3.3 Inferring c2p relationships

We implement our technique detailed in section 4.2.1, reusing parts of the code from [83], utilizing the LEDA v4.5 software library [92], and a publicly available SDP solver DSDP [93]. We remove edges inferred as s2s from E and compute orientations of the remaining edges in E for different values of α. We vary α, sampling densely the interval of its values between 0 and 1.

To evaluate the computed orientations, we introduce a metric called *reachability*. We define reachability of AS X as the number of ASs one can reach from AS X traversing only p2c edges. We then sort all ASs by their reachability, and group ASs with the same reachability into *levels*. ASs at the highest level have the largest trees of customer ASs. ASs at the lowest level have the smallest reachability. We then define the position *depth* of AS X as the number of ASs at levels above the level of AS X. We call the position *width* of AS X the number of ASs at the same level as AS X. The reachability of an AS has the following two properties: 1) it is determined entirely from the inferred c2p relationships; and 2) it induces a natural hierarchy of ASs based on the size of their customer trees. These two properties enable us to perform initial validation of the inferred c2p relationships by matching the top ASs in the hierarchy against the empirically known largest ISPs in the Internet.

In Table 6 we examine the top five ASs in the hierarchies calculated for $\alpha = 0$ and $\alpha = 1$. When $\alpha = 0$, the top five positions in the hierarchy are occupied by the well-known ISPs: UUNET, AT&T, Sprint, Level 3, and Qwest. On the other hand, when $\alpha = 1$, these positions are taken by ASs of very small degrees, e.g., AS13987 of degree 3. The columns in the table track the position of these ASs in the hierarchies induced for α equal to 0, 0.01, 0.05, 0.1, 0.5, and 1. We observe that as α gets closer to 1, the well-known ASs get away from the top of the hierarchies, highlighting an increasingly stronger deviation from reality. This deviation is maximized when $\alpha = 1$. Recall that when $\alpha = 1$, our problem formulation

Table 6. The reachability-based hierarchy of ASs and percentage of invalid paths as functions of α. For different values of α, we show the position *depth* (the number of AS at the levels above) and *width* (the number of ASs at the same level) for the ten ASs that occupy the top five positions when α takes its two extreme values: $\alpha = 0$ and $\alpha = 1$. The AS numbers are matched to AS names using the WHOIS databases.

				$\alpha = 0.00$		$\alpha = 0.01$		$\alpha = 0.05$		$\alpha = 0.10$		$\alpha = 0.50$		$\alpha = 1.00$	
Percentage of invalid paths															
				12.75%		1.79%		0.69%		0.46%		0.36%		0.33%	
Top of reachability based hierarchy															
	AS #	name	degree	dep.	wid.	dep.	wid.	dep.	wid.	dep.	wid.	dep.	wid.	dep.	wid.
$\alpha = 0$	701	UUNET	2334	0	1	1	1	0	105	0	120	2	201	11	319
	7018	AT&T	1911	1	1	2	1	0	105	0	120	2	201	11	319
	1239	Sprint	1703	2	1	0	1	0	105	0	120	2	201	11	319
	3356	Level 3	1228	3	1	3	1	0	105	0	120	2	201	11	319
	209	Qwest	1105	4	1	4	1	0	105	0	120	2	201	11	319
$\alpha = 1$	14551	UUNET	35	128	1	137	2	138	1	151	1	260	2	0	1
	13987	IBASIS Inc.	3	1792	955	1802	963	1830	976	1847	971	1885	966	1	2
	8631	Routing Arbiter	48	108	1	123	1	122	2	0	120	0	1	1	2
	23649	Hong Kong Teleport	4	1792	955	1802	963	899	121	916	121	967	119	3	8
	4474	Village Communications	2	2747	16136	2765	16118	2806	16077	2818	16065	2	201	3	8

is equivalent to the original ToR formulation, signifying the limitations of AS relationship inference based solely on maximization of the number of valid paths.

The induced hierarchies suggest that solutions with values of α close to 1 are incorrect since they propel small ASs to the top of the hierarchy, while well-known ISPs sink to lower positions. On the other hand, the percentage of invalid paths, listed at the top of Table 6, attains its maximum of 12.75% when $\alpha = 0$. The latter observation suggests that the solution with $\alpha = 0$ is also incorrect since it conflicts with the valley-free routing model. Taken together, these two observations indicate that correct solutions to our multiobjective optimization correspond to intermediate values of α that result both in realistic hierarchies and in small numbers of invalid paths. We have to emphasize that there is no oracle, intrinsic to the multiobjective optimization problem formulation, that would reveal to us the value of α reflecting the "right" balance between the two objectives. As always with multiobjective optimization [86], we have to exercise our external expert knowledge of data specifics to sift out the most realistic relative weight of the objectives. For our problem, we formalize this expert insight as follows: we search for the value of α corresponding to the smallest percentage of invalid paths among all the solutions that have only well-known ISPs at the top of the hierarchy. In our experiments, this most realistic value of α is 0.01, cf. Table 6.

4.3.4 Inferring p2p relationships

We implement our p2p heuristic detailed in section 4.2.3, using the QUALEX [91] solver to approximate the MWIS problem. We then infer p2p relationships in the AS topology G and construct the set F of p2p links. After removing from F the set S of s2s links, we obtain our final answer that contains $3,553$ p2p links ($|F \setminus S| = 3,553$).

4.3.5 Summary of experiments

In Table 7, we show the summary statistics of our experiments. Out of $38,282$ links in the original AS topology $G(V, E)$, we infer a majority of $34,552$ (90.26%) links as c2p, $3,553$ (9.28%) links as p2p, and only 177 (0.46%) links as s2s.

Table 7. Summary statistics of the inferred relationships.

	Total $\|E\|$	c2p links $\|E \setminus F \setminus S\|$	p2p links $\|F \setminus S\|$	s2s links $\|S\|$
number of links	38,282	34,552	3,553	177
percentage	100%	90.26%	9.28%	0.46%

4.4 Survey and Validation

Measuring, understanding, and modeling AS relationships in the Internet are challenging tasks hampered by the fact that these relationships are sensitive business information and generally considered private by ISPs. Nevertheless, without validation against truth, we have no way of evaluating the integrity of a model. For this reason, we validated our inferences via private communications with engineers from the ASs under observation.

We contacted several ASs ranging from large continental or national ISPs, to content providers, and university networks. We sent the list of AS relationships that we inferred for a given AS to this AS's network administrator, peering negotiator, informed engineer, or researcher. We included three questions in our email inquiry:[5]

Q1: For the listed inferred AS relationships, specify how many are incorrect, and what are the correct types of the relationships that we mis-inferred?

Q2: What fraction of the total number of your AS neighbors is included in our list?

Q3: Can you describe any AS relationships, more complex than c2p, p2p, or s2s, that are used in your networks?

We performed the survey in the period between 06/07/05 and 06/30/05. We received answers from 38 out of the 78 ASs we contacted. Among these, 5 were tier-1 ISPs, 13 were smaller ISPs, 19 were universities, and 1 was a content provider. Based on the replies we collected from the surveyed ASs, we learned the true relationship types for 3,724 of

[5]We also offered to sign a non-disclosure agreement (NDA) that protected peering information from being released to the public and regulated our data analysis to anonymizing the participating ASs. Only one organization (a government agency) required an NDA and two commercial ISPs did not have a policy in place (or the policy was not) to deal with such requests. They still helpfully provided us with general answers regarding what percent of relationships were mis-inferred.

Table 8. Validation of the inference results using the survey data. Each row shows the total number, number of correct, and percentage of correct inferred AS relationships.

	links	inferred c2p links	inferred p2p links	inferred s2s links
total number of	3,724	3,070	623	31
number of correct	3,508	2,964	516	28
percentage of correct	94.2%	96.5%	82.8%	90.3%

our inferred AS relationships, which corresponded to 9.7% of the total number of links in our graph. We learned only 54 true relationships from the universities, whereas all the remaining relationships came from the ISPs and the content provider. Among the 3,724 verified AS relationships, 82.6% were c2p, 16.1% were p2p, and 1.2% were s2s.

4.4.1 Validation of inferred AS relationships

We validate our heuristics by counting the number of correctly inferred AS relationships. Our validation results in Table 8 highlight that our model correctly infers 96.5% of c2p, 82.8% of p2p, and 90.3% of s2s relationships. The total percentage of correctly inferred AS relationships is 94.2%. This accuracy level substantiates that our heuristics produce reliable and veracious inferences of the true types of AS relationships in the Internet.

4.4.2 Missing AS links

In this section we leap forward to analyze the relationships of the full set of adjacencies of the participating ASs, including the links that we do not see in BGP tables and, consequently, in our graph. The second question in our survey asks ASs for the ratio of the number of their neighbors in our AS topology data to the total number of AS neighbors they actually have. Out of the 38 ASs, 27 (3 of which were tier-1 ISPs) provided us not only with this ratio, but also with the types of relationships their ASs have with the missing neighbors. We analyze the 1,114 true adjacencies of these ASs and compared them with the BGP table data.

In Figure 24 we plot the per-relationship breakdown of the true and observed adjacencies of the ASs. Out of a total of 1,114 adjacencies, BGP tables contain only 552. The p2p,

c2p, and s2s bars in Figure 24 lead us to the following observations.

1) We only see 38.7% out of the 865 p2p links, whereas we see 86.7% out of the 218 c2p links, and 93.3% of the 30 s2s links. This gap demonstrates that BGP-derived AS topologies miss a notable percentage of p2p links. The reasons these links are missed stem from the intrinsic nature of p2p relationships. In a p2p relationship, prefixes learned from a peer AS are not advertised to any providers. Consequently, a link between two p2p ASs is not by default seen (as a part of some AS path) at any upstream ASs. It follows that we can only observe a p2p link in the BGP tables of the customers or siblings of the p2p ASs. The periphery of the Internet has many small interconnecting customer and sibling ASs. Thus, in order to observe a significant portion of true p2p links in the periphery, we should have great numbers and variety of BGP tables from these small ASs. Therefore, it becomes evident that it is practically hard to capture p2p links, or even some representative statistics of p2p links, using solely BGP tables with small number of data feeds.

2) The majority of the 1,114 true adjacencies are in reality p2p: 865 (77.6%) are p2p, while only 218 (19.6%) and 30 (2.7%) are c2p and s2s, respectively. We thus face a surprisingly large number of p2p relationships. These relationships appear to be popular among small and medium size ASs. More surprisingly, we find that p2p links are also very common for tier-1 ISPs. Some tier-1 ISPs have several dozens or even *hundreds* of p2p relationships, frequently with ASs of smaller size. This finding contradicts with the common belief that tier-1 ASs form a rich club and have p2p relationships only with other tier-1 ASs in the rich club. In a sharp contrast with this belief, we observe that tier-1 ISPs can have more open policies and more sophisticated incentives to establish peering relationships with smaller ISPs.

Next, we seek to evaluate how representative the BGP-derived AS degrees are of the true AS degrees. In Figure 25 we plot the number of true AS adjacencies of the surveyed ASs versus the number of AS adjacencies we observe in our BGP data. For few selected ASs that exhibit the highest percentage of missed adjacencies, we mark this percentage in

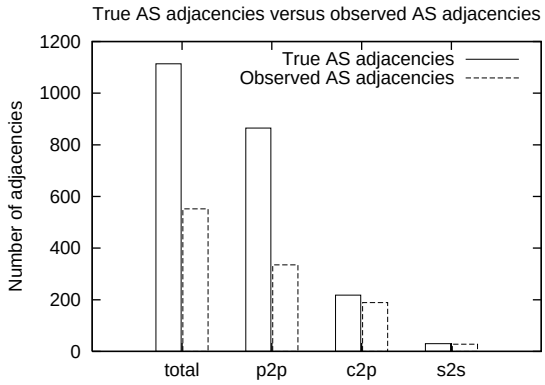


Figure 24. Number of true and observed AS adjacencies for different types of AS relationships.

the figure. We make the following observations.

1) At the bottom-left corner of Figure 25, 20 ASs[6] that are mainly university networks, have their true numbers of adjacencies close or identical to the measured numbers of adjacencies. We find that most of the adjacencies of these small ASs are c2p links. As we have seen above, our AS topology captures c2p links relatively well.

2) Examining the rest of the diagram, we first observe that the percentage of missed adjacencies can be as large as 86.2%. Degrees of a majority of the highly connected ASs are under-sampled, half of them missing more than 70.5% links. Further examination of the missed AS links reveals that most of them are of p2p type, which is consistent with Figure 24, and with our understanding that extensive peering not amenable to capture from few BGP feeds can render BGP table-derived AS degrees a poor estimate of true AS degrees.

4.4.3 Complex AS relationships

The last question of our survey asks about more complex configuration scenarios the AS may be using. From the responses we learn that although the majority of AS relationships are simply c2p or p2p, in some cases their configurations are either more specialized versions of the basic c2p or p2p types, or a hybrid of c2p and p2p (c2p/p2p).

For example, the backup provider relationship is a specialized variant of the basic c2p relationship. In this relationship, a customer AS has a c2p relationship with a provider AS,

[6]Note that points (1,1) and (2,2) in the figure correspond to more than one AS.

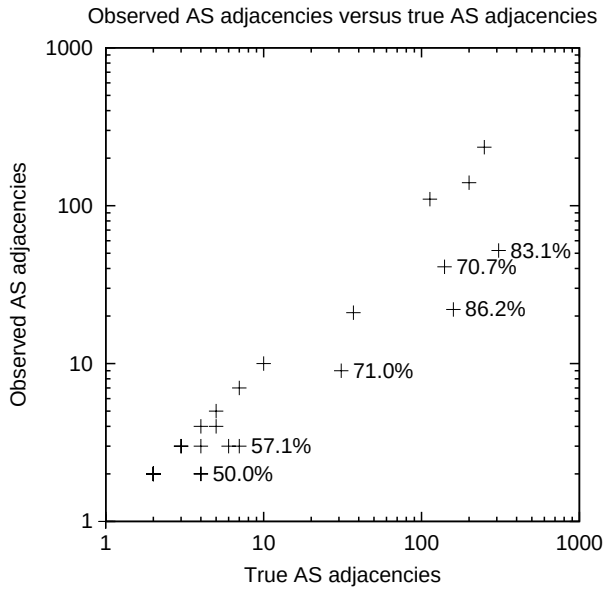


Figure 25. Number of true adjacencies versus number of observed adjacencies for the surveyed ASs. For the ASs that exhibit the highest percentage of missed adjacencies, we mark this percentage on the figure.

but this relationship is active, i.e., allows traffic to flow, only during an emergency situation such as a disruption of connectivity to the main upstream provider of the customer AS. In other words, the backup provider relationship is a temporally conditioned version of the c2p relationship.

An example of a hybrid c2p/p2p relationship is when two ISPs interconnect at multiple peering points and have different types of relationships at these points. For instance, two ISPs can have a p2p relationship at a peering point in Europe and a c2p relationship at a peering point in the U.S. Another flavor of a hybrid c2p/p2p relationship is when two ASs have different types of relationships for different IP prefixes. In this case the ISPs may have a p2p relationship for one set of IP prefixes and a c2p relationship for another set of IP prefixes. The last two hybrid c2p/p2p relationships illustrate that there also exist spatial and prefix-based aspects to AS relationships, conditional on either geographic location or on a specific segment (IP prefix) of the address space.

In other words, based on the configuration descriptions we collected in our survey, we conclude that AS relationships vary across the following three dimensions: space, time,

and prefix. Therefore, to fully characterize a relationship between a pair of ASs, one has to gain access to information identifying the ASs' policy configurations per peering location(s), per time, and per prefix. Such details would enable one to infer more complex relationships, e.g., the hybrid c2p/p2p or specialized c2p relationships discussed above. Although collecting macroscopic statistics on complex AS relationships is a daunting task, the increasing availability of Internet routing data might make this task feasible in the future.

4.4.4 Lessons learned

Overall, analyzing the valuable data we collected in our survey, we make the following observations:

1. Our inference heuristics perform exceptionally well in inferring AS relationships of AS links observed from BGP tables. The overall accuracy is 94.2%.

2. p2p is the dominant type of AS relationships in the examined ASs, 77.6% of their actual adjacencies being p2p.

3. AS topologies extracted from BGP tables substantially under-sample the number of p2p links. In our data we see only 38.7% of the true p2p links present in the survey data. The percentage of missing links per AS, i.e., the AS degree underestimate, reaches 86.2%.

4. AS relationships vary across the following three dimensions: space, time, and prefix. To completely characterize a relationship between two ASs, one needs to know their per peering point, per time, and per prefix policy configuration details.

4.4.5 Limitations

We note that the data in our survey bears certain limitations and that our results should be interpreted accordingly.

First, the self-selection aspect of the sampling of ASs may induce biases into the validation, and the 3,724 links we used for validation corresponds only to 9.7% of the total

number of links in our data. Although we acknowledge these limitations, we note that providing rigorous validation of inferred AS relationships is an extremely challenging task because of the difficulty in obtaining ground-truth data against which we can match our inferences. In comparison with previous studies that mostly did implicit validation, we go a step further: besides implicit validation in section 4.3.3, we also make an effort to collect samples of the ground truth to perform explicit validation.

Second, our findings suggesting dominance of p2p links and high inaccuracy of BGP-derived AS degrees are based on a limited set of data. For this reason, our results cannot be interpreted as accurately representing the AS structure of the whole Internet. Indeed, our research yields cases that contradict the common belief that BGP-derived topologies are reasonably accurate and that the number of p2p links is much smaller than the number of c2p links. We hope our findings will inspire further pursuit of representative statistics on the number of p2p links in the Internet.

4.5 Repository of AS Relationships and AS Rank

To make our results easily accessible and practically useful for the community, we develop an AS relationship repository where we archive the inferred AS relationships on a weekly basis and make them publicly available. We also apply our inferences to rank ASs based on ASs' prominence in the hierarchy induced by the inferred relationships. We seek to measure this prominence by the relative size of the Internet that a given AS can reach "for free", that is, following only p2c links. The prominence of an AS is then the cumulative sum of prominence of the ASs in its customer tree, cf. Table 6. Users of our AS ranking service have many options to tune their settings including options to: 1) measure prominence of an individual AS by: a) the total size of the IP address space advertised by the AS, b) the total number of the advertised prefixes, or c) just 1, meaning that all ASs are treated equal; and 2) to group multiple sibling ASs into sibling groups defined by: a) the data in the IRRs, or b) user-provided sibling lists. AS ranking is valuable not only for conceptual

understanding of relative importance of Internet players, but also for network vendors and operators in prioritizing their customer lists and in solving other practical tasks. We develop a website [84] where we make both the AS ranking and the raw AS relationship data freely available to the public.

4.6 Comparing with Previous Works

Gao's seminal work [4] (LG) was the first to formulate and systematically study the AS relationships inference problem. Gao proposed to infer c2p and s2s relationships using a heuristic that treated every AS path as a hint of true types of links in the path. More specifically, this approach takes a set of AS paths as input, directs every link toward the highest degree AS in the path, and after parsing all paths, counts the directions each link has accumulated. If a link has received consistent directions throughout the process, it is marked as c2p with the provider being at the top of the directed link. Otherwise, the link is marked as s2s. This approach is sensitive to path anomalies. In the presence of anomalies, c2p edges may receive inconsistent directions leading to mis-inference of true c2p relationships as s2s. Our c2p heuristic is resilient to anomalous paths since we employ the node degree-gradient to detect, accept, and intelligently process invalid orientations. Moreover, our s2s heuristic exploits explicit information in the IRR databases and does not rely on BGP tables. We believe it is quite hard to reliably infer s2s relationships using solely BGP tables.

In inferring p2p links LG constructs the candidate lists consisting of links located only between the top provider in a path and its highest degree neighbor. The algorithm then marks the candidate links as p2p if the degree ratio of their adjacent ASs is smaller than 60 or ∞. Our p2p heuristic borrows several ideas from this approach and improves it in the following ways: 1) we do not force candidates to lie just between the top provider and its highest degree neighbor; 2) we use a more flexible weighting strategy than just a degree ratio cut-off; 3) we exercise our MWIS-based cleaning process to filter out from the candidate

lists those links violating the valley-free model.

The SARK [81] work translates the AS relationship inference task into an elegant combinatorial optimization problem, ToR, and proposes a heuristic to solve it. This heuristic takes as input the BGP tables collected at different vantage points and computes a *rank* for every AS. This rank is a measure of how close to the graph core an AS lies and it is equivalent to node *coreness* [94]. The heuristic then infers AS relationships by comparing ranks of adjacent ASs. If the ranks are similar, the algorithm classifies the link as p2p, otherwise it is c2p. We believe that node coreness is not an appropriate metric for inferring AS relationships, since it is unclear how ASs' topological positions in AS graphs are related to AS relationships. Instead of measuring how close to the graph core ASs are positioned, we exploit the policy encodings in the AS paths to infer AS relationships.

The DPP [82] and EHS [83] studies provide a mathematically rigorous solution to the ToR problem. They first demonstrate that the ToR framework does not have any mechanism allowing one to rigorously infer p2p relationships. Both DPP and EHS retreat therefore to inferring c2p relationships only. Although DPP and EHS deliver a formally best solution to ToR, the purpose of our exhaustive discussion in Section 4.2.1 is to show that ToR is not an appropriate problem formulation since it quickly leads to too many incorrect inferences.

The work by Xia *et al*. [90] utilizes the IRR databases to extract relationships among a subset of ASs. They also propose a variation of the LG heuristic that takes this subset as an input to infer other AS relationships. Their major contribution is dealing with accurate and current IRR databases providing explicit information on AS relationships. On the other hand, using IRR data has its own methodological problems: 1) it is much harder to automate: in particular, extracted data requires deciphering non-standardized representations and manual searching ASs' websites to determine their use of BGP communities; 2) the data is not always accurate and its accuracy level is hard to estimate; 3) not all ASs are registered. We also use the IRR data in our work but only for s2s relationship inference. Instead of policy records, we process only the organization description records that

are relatively stable over time.

Mao *et al.* [77] propose an AS relationship inference technique based on the assumption that ASs prefer shorter AS paths over longer AS paths. In reality, ASs often use routing policies selecting next-hop ASs based solely on next-hop AS rankings, meaning that they make their route selections regardless of the AS path lengths. It is thus unclear if an AS path length formulation of the problem can lead to correct inferences.

Most of the previous works resorted to implicit validation. However, such methodology does not always yields reliable results. For example, the techniques in [81, 82, 83, 77] all used the number of valid paths as an indicator of the accuracy of the inferred relationships. As we discussed in section 4.2.1, a large number of valid paths does not necessarily correspond to a large number of correctly inferred AS relationships. In contrast with previous works, we base our validation both on implicit metrics, e.g., reachability from section 4.3.3, and on the explicit data we collected in our survey.

4.7 Conclusion

The relationships among ASs in the Internet represent the outcome of policy decisions governed by technical and business factors of the global Internet economy. Precise knowledge of these relationships is therefore an essential building block needed for any reliable and effective analysis of technical and economic aspects of the global Internet, its structure, and its growth.

In this work we introduced novel heuristics that accurately infer c2p, p2p, and s2s relationships and that effectively resolve the limitations we identified with previous approaches. Our heuristics significantly improve the state-of-the-art in inferring c2p relationships and carefully address the particularly difficult problems of inferring p2p and s2s relationships from currently available data.

Another significant contribution of our work is that we tackled the challenging task of

validating inferred AS relationships. We conducted a survey with AS network administrators to gather information on the actual relationships they have with their neighbors. Using the data in our survey, we validated our heuristics and demonstrated that they achieve high accuracy of 96.5% (c2p), 82.8% (p2p), and 90.3% (s2s) of correctly inferred relationships.

The data we collected in our survey allowed us to gain insight into the complete structure of the actual relationships of the surveyed ASs. To our great surprise, we discovered that p2p relationships are much more dominant than we could expect: 77.6% of the validated adjacencies were p2p. We also found that the currently available BGP data underestimates substantially the number of the p2p relationships existing in reality. In particular, the AS topologies extracted from the BGP data miss up to 86.2% of the true links adjacent to an AS.

Easy access to accurate AS relationship data is essential to a variety of studies dealing with aspects of Internet architecture and policy. To support the research community with as objective data as possible, we have automated our heuristics in order to calculate and archive AS relationships on a weekly basis. We also use the inferred relationships to provide a ranking of ASs; we post both the data and the ranking on the web [84]. We believe our work will significantly benefit Internet research that strives to build more realistic models validated against reliable and accurate AS topology data.

CHAPTER 5

AS TAXONOMY

5.1 Introduction

In this chapter we turn our attention to the ASs, the individual entities that operate the Internet. The AS topology of the Internet is an intermix of networks owned and operated by many different organizations, e.g., backbone providers, regional providers, access providers, universities, and private companies. Although the AS topology of the Internet has been studied extensively, statistics on the different types of ASs in the Internet are not presently available. Statistical information that faithfully characterizes different ASs is on the critical path toward understanding the structure of the Internet, as well as for modeling its topology and growth. In this work, we propose an AS taxonomy, i.e., a set of AS types that characterize ASs, and we seek to derive macroscopic statistics on the different types of ASs in the Internet.

In topology modeling, knowledge of AS types is mandatory for augmenting synthetically constructed or measured AS topologies with realistic intra-AS and inter-AS router-level topologies. For example, we expect the network of a dual-homed university to be drastically different from that of a dual-homed small company. The university will likely contain dozens of internal routers, thousands of hosts, and many other network elements (switches, servers, firewalls). On the other hand, the small company will most probably have a single router and a simple network topology. Since there is such a diversity among different network types, we cannot accurately augment the AS-level topology with appropriate router-level topologies if we cannot characterize the composing ASs.

Moreover, annotating the ASs in the AS topology with their types is a prerequisite for modeling the evolution of the Internet, since different types of ASs exhibit different growth patterns. For example, Internet Service Providers (ISP) grow by attracting new customers

and by engaging in business agreements with other ISPs. On the other hand, small companies that connect to the Internet through one or few ISPs do not grow significantly over time. Thus, categorizing different types of ASs in the Internet is necessary to identify network evolution patterns and develop accurate evolution models.

An AS taxonomy is also necessary for mapping IP addresses to different types of users. For example, in traffic analysis studies its often required to distinguish between packets that come from home and business users. Given an AS taxonomy, its possible to realize this goal by checking the type of the AS that originates the prefix in which an IP address lies.

In this chapter, we first propose a set of classes representing Internet ASs: large ISPs, small ISPs, customer ASs, IXPs, NICs, and universities. Then, for each AS we collect the following AS attributes: organization description name; number of inferred customers, providers, and peers; number of advertised IP prefixes; and total advertised IP space. We construct a training set of ASs, for which we manually determine their true AS type. Then we apply a machine-learning algorithm to the attributes of the training ASs and extract patterns associated with different AS types. We express this patterns into a set of classification rules, which we apply to the full set of Internet ASs and derive macroscopic statistics of the different types of ASs in the Internet. Next, we validate our inferences and find that our classification algorithm achieves a high accuracy: 78.1% of the examined classifications were correct. To promote further research and understanding of the Internet's structure, we develop a website [30] in which we make the collected AS attributes and the inferred AS types publicly available.

In Section 5.2 we start with a brief discussion of related work. Section 5.3 describes the data we used, and in Section 5.4 we specify the set of AS classes we use in our experiments. Section 5.5 introduces our classification approach and results. We validate them in Section 5.6 and conclude in Section 5.7.

5.2 Related Work

Several works have developed techniques decomposing the AS topology into different levels or tiers based on connectivity properties of BGP-derived AS graphs. Govindan and Reddy [95] propose a classification of ASs into four levels based on their AS degree. Ge *et al*. [96] classify ASs into seven tiers based on inferred customer-to-provider relationships. Their classification exploits the idea that provider ASs should be in higher tiers than their customers. Subramanian *et al*. [81] classify ASs into five tiers based on inferred customer-to-provider as well as peer-to-peer relationships.

Our work differs from previous approaches in the following ways:

1. We use a novel machine learning algorithm to identify intrinsic features distinguishing different AS types. In contrast to the discussed previous approaches, we do not employ heuristics or ad-hoc thresholds to classify ASs. Our classifier uses rules and associated thresholds, which are derived rigorously so as to maximize the expected accuracy of the classification.

2. We use an extensive set of diverse data including IRR records, inferred AS relationships, IP prefixes, and AS graphs, to classify ASs. In contrast to previous approaches, we do not rely exclusively on AS graphs, which often miss a substantial fraction of the true AS links in the Internet, resulting in incomplete AS topologies.

3. We propose a representative set of AS classes that reflect ASs with different network properties. Previous approaches, classify ASs into hierarchies of levels or tiers extracted from AS graphs; these methods tend to mix ASs with substantially different network properties into a single AS group. According to our analysis, small regional providers often have small AS degrees, as low as 1 or 2. The previous heuristics thus tend to assign these ASs to the lowest levels, where small companies and multihomed customers naturally reside.

5.3 Data Sources and AS Attributes

To construct the set of AS attributes that we use in our AS classification, we collect data from the following databases and measurement projects:

1) **IRRs** [68]. The IRRs constitute a distributed database containing records on ASs' routing policies, assigned IP prefixes, contact information, etc. A natural approach to identifying the type of an AS, given its AS number, is to lookup the AS number in the IRRs and examine its organization description record. In the RPSL [97] terminology, this record is the `descr` attribute of the RPSL class `aut-num`. It contains the name or a short description of the organization that owns the AS number. For example, the following are entries for the `descr` attribute found in the IRRs: "Intervivo Networks, a broadband Internet access provider" and "Auckland Peering Exchange". The `descr` attribute does not have a standard representation. It usually consists of a short description as in the examples above, but in some cases it only contains an acronym, e.g., "KPMG LLP", "LTI". For the purposes of this work, the `descr` record is our first AS attribute. We use text classification techniques (outlined in section 5.5) to analyze the `descr` records and to find keywords, like "university" or "ISP", that reflect different AS types.

We downloaded the mapping of AS numbers to organization description records on 04/08/2005 from the CIDR Report [98], which provides on a daily basis mappings of AS numbers to organization description records extracted from ARIN, RIPE NCC, LAPNIC, APNIC, KRNIC, TWNIC, and JPNIC databases. We preprocess the organization records by removing stop words, i.e., words with little semantic meaning, such as "of" or "the", using the stop word list [99]. Then, using the Porter stemming algorithm [100], we replace words with their stem.

We note that IRRs contain significant portions of incomplete or obsolete records, which is not a serious problem for this study since we are only concerned with the `descr` attribute, which changes relatively rarely.

2) **RouteViews** [49]. We download all 12 BGP table snapshots archived from the collector `route-views2.oregon-ix.net` on 07/18/2005. For each table snapshot we extract AS paths and remove AS sets and private AS numbers. Then, we merge the extracted AS paths into an AS topology and use the AS relationship inference heuristics described in chapter 4 to annotate the AS links with customer-to-provider and peer-to-peer relationships. We ignore sibling-to-sibling relationships, since they only account for 0.46% of the total number of links in the graph. After inferring AS relationships, we calculate the following three attributes for every AS: the number of providers, the number of customers, and the number of peers a given AS is connected to. Large ISPs typically have a large number of customers, zero providers, and a small number of peers, while small ISPs typically have few providers, a small number of customers and a large number of peers. Stub university or company networks typically have zero customers, zero or few peers and a small number of providers.

From the RouteViews data, we also extract information on IP prefixes. We use the chronologically first table snapshot from the same snapshot set to construct a mapping of AS numbers to the IP prefixes they advertise. Then, for each AS we count the number of advertised prefixes and use this number as another AS attribute. Small ASs, with tiny portions of IP address space allocated to them, as well as older ASs with large IP blocks, tend to advertise few prefixes. On the contrary, large ASs with relatively high IP address utilization and diversified routing policies tend to advertise many prefixes of various lengths.

One problem with this attribute is that IP prefixes are of drastically different sizes. Advertised IPv4 prefixes range in size from a /8, covering 2^{24} IP addresses, down to a /32, covering just one address. The prefix length of 24 bits is generally the smallest IPv4 prefix size that is globally routed, which suggests our last AS attribute to be the number of unique /24 prefixes found within the union of address space advertised by the AS. This attribute is likely to have small values for smaller ASs that use and advertise smaller portions of IP address space, while it is likely to be at its maximum for large or old ASs (those that

appeared in the Internet early, e.g., some military and academic networks) since they often have huge chunks of assigned IP address space that they utilize scarcely.

In summary, we collect data from the IRRs and RouteViews. We find 19,537 ASs. Using the collected data, we annotate every AS with the following six attributes: 1) the organization description record (the *description* attribute), 2) the number of inferred customers (the *customer* attribute), 3) the number of inferred providers (the *provider* attribute), 4) the number of inferred peers (the *peer* attribute), 5) the number of advertised IP prefixes (the *prefix* attribute), and 6) the equivalent number of /24 prefixes covering all the advertised IP space (the *space* attribute).

5.4 The AS Class Set

In this work, we focus on network properties of an AS as the main criterion determining the set of AS classes. In other words, to construct the AS class set, we use the rule that ASs in the same class should have similar network properties, while ASs in different classes should have different network properties. ASs in the same class may still have significant network differences, however these differences should be small compared with differences between networks of different classes. For example, a small university and a large university may have quite different networks, however these differences are less significant compared to the differences between a typical university network and a typical ISP network.

Besides employing our *de facto* empirical knowledge, we perform the following experiment to specify the set of AS classes. We *randomly* select 1200 ASs and then, for each AS, we examine its attributes, visit its website (if possible), search for references to its organization name and determine its business profile. After examining the spectrum of these 1200 ASs, we construct our AS class set:

1. *Large ISPs*: Large backbone providers, tier-1 ISPs, with intercontinental networks.

2. *Small ISPs*: Regional and access providers with small metropolitan or larger regional networks.

3. *Customer ASs*: Companies or organizations that run their own networks but as opposed to members of the previous two classes do not provide Internet connectivity services. We find a wide range of ASs in this class, like web hosting companies, technology companies, consulting companies, hospitals, banks, military networks, government networks, etc.

4. *Universities*: University or college networks. We distinguish these networks from members of the Customer AS class, since they typically have substantially larger networks that serve hundreds or thousands of end hosts.

5. *IXPs*: Small networks serving as interconnection points for the members of the first two classes.

6. *NICs*: Networks that host important network infrastructure, such as root or TLD servers.

5.5 AS Classification: Algorithms and Results

We build our classification algorithm using *AdaBoost* [101], a powerful machine learning technique. The main idea behind AdaBoost is to combine multiple simple classification rules into a efficient composite classifier. These simple classification rules are called *weak hypotheses* and by definition are only required to perform slightly better than random guessing. Intuitively, weak hypotheses reflect simple "rules of thumb" that are usually much easier to construct than a complex classifier. AdaBoost works iteratively over a set of training examples; at each iteration it finds a weak hypothesis that performs well on the examples which the weak hypotheses of previous iterations erroneously classified. One constructs a weak hypothesis by means of a *weak learning algorithm* or simply *weak learner*. The power of AdaBoost lies in a well-developed theoretical framework that intelligently combines the weak hypotheses into a composite classifier.

Let X denote the set of ASs that we want to classify and Y be the set of possible classes,

Algorithm 1: *AdaBoost.MH* pseudocode

Input: $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$
Initialize: $D_1(x, y) = 1/mk$; `// k is the total number of classes`
for $t = 1$ **to** T **do**
  Pass distribution D_t and examples S to weak learner
  Get weak hypothesis $h_t : X \times Y \to \mathbb{R}$
  Update distribution

$$D_{t+1}(x, y) = \frac{D_t(x, y) exp(-P(x, y) h_t(x, y))}{Z_t}$$

  `// where` Z_t `is a normalization coefficient`
  `// chosen so that` D_{t+1} `will be a distribution`
end
Output: $f(x, y) = \sum_{t=1}^{T} h_t(x, y)$

such that each AS $x \in X$ belongs to a unique class in Y. If $y \in Y$ is the correct class for AS x, than let $P(x, y) = 1$, otherwise $P(x, y) = -1$. The goal is to find a classifier that for each AS produces a *ranking* of all the possible classes. More formally, we will compute a ranking function $f : X \times Y \to \mathbb{R}$: for each AS $x \in X$, the classes in Y will be ordered according to $f(x, \cdot)$—the higher the value of $f(x, y)$, the more likely x belongs to y.

In Algorithm 1, we illustrate the AdaBoost.MH algorithm [102], a special member of the AdaBoost algorithm family that is suited for solving multiclass problems. Let $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ be the set of training examples. In our case, we construct S by manually determining the correct class of 1220 ASs: 1200 are randomly selected; and 20 are well-known large ISPs and IXPs, which we use to increase the number of these two types of ASs in the initial random sample. Let T be the total number of iterations. For each iteration $t = 1 \ldots T$, we maintain a distribution D_t of weights over the set of examples and classes $D_t : X \times Y \to \mathbb{R}$. At the first iteration, we initialize D_t to the uniform distribution, i.e., D_1 is constant for all (x_i, y_i), $1 \leqslant i \leqslant m$. At each subsequent iteration, we pass the distribution D_t and the training examples S to a weak learner that computes a weak hypothesis $h_t : X \times Y \to \mathbb{R}$. A positive (negative) sign of the weak hypothesis $h_t(x, y)$ corresponds to the prediction that AS x is (is not) a member of class y. The value $|h_t(x, y)|$

of the weak hypothesis reflects the confidence level of the prediction. Then, we update the distribution D_t so that the x, y pairs that were erroneously predicted, i.e., the signs of $h(x, y)$ and $P(x, y)$ differ, receive a exponentially higher weight. By assigning higher weight to incorrect predictions, we force the algorithm to focus on these difficult examples in the next round. The final classifier f is the sum of votes of the weak hypotheses in all rounds h_t, $t = 1 \ldots T$.

A weak hypothesis is equivalent to a one-level decision tree that checks a single AS attribute. For the description AS attribute, a weak hypothesis searches for the presence of a term or a sequence of terms in a given record, and if a match occurs, it outputs a confidence value for each of the classes. For example, upon finding the term "university" in the record "Seoul National University of Education" the weak hypothesis will likely output a high positive confidence value for the University AS class and a negative confidence value for the other AS classes. For scalar attributes, a weak hypothesis asks if a given attribute value is above or below a certain threshold. Depending on the outcome, the hypothesis outputs different confidence values.

The weak learner builds a weak hypothesis by exhaustively evaluating the attributes in the given weighted training examples. For a text attribute, it builds a candidate weak hypothesis by evaluating all possible terms and sequences of terms. For each term or sequence of terms, it calculates the appropriate confidence values by minimizing the Hamming loss, which is the fraction of examples x and classes y, for which the sign of the final classifier $f(x, y)$ differs from $P(x, y)$. Similarly, for each scalar attribute, the weak learner builds a candidate weak hypothesis by exhaustively searching the threshold and confidence values minimizing the Hamming loss. On its output, the learner returns the weak hypothesis attaining the minimum Hamming loss.[1]

To realize our classification algorithm, we use BoosTexter [102], a publicly available implementation of AdaBoost. In Table 9 we depict the weak hypotheses that our algorithm

[1] See [102] for the analytic expression of the Hamming loss.

Table 9. The list of weak hypotheses computed by AdaBoost.MH in the first six iterations. The first column is the iteration number; the second is the AS attribute that the weak hypothesis is checking; the third is the term or threshold of the weak hypothesis; and the remaining columns depict the computed negative or positive confidence values for each of the AS classes.

Round	Attribute	Term/Threshold	L.ISP	S.ISP	Cusmr	Uni	IXP	NIC
1	space	< 8.5						
		> 8.5						
2	description	"network inform"						
3	customer	< 1.5						
		> 1.5						
4	description	"exchang"						
5	description	"univers"						
6	customer	< 97						
		> 97						

discovered during its first six iterations. For each weak hypothesis, we illustrate the selected AS attribute, the term or threshold that is looked for, and the computed confidence values. The first weak hypothesis deals with the space attribute. If the IP address space advertised by an AS is less than 8.5 (equivalent) /24 prefixes then, the hypothesis assigns a positive confidence value to the Customer AS class and negative confidence values to the other classes. If the value of the space attribute is above 8.5, the hypothesis assigns negative or very close to zero confidence values to all the classes, which means that in this case it cannot make a confident positive prediction. The second weak hypothesis checks the description AS attribute for the presence of term "network inform".[2] If it finds one, it assigns a high positive confidence to the NIC AS class and negative confidence values to the other classes. Note, that in some cases the weak hypothesis assigns zero confidence values, meaning that it abstains from making any prediction.

We experimentally fix the number of rounds T to 28, since subsequent iterations lead to overfitting. Overfitting is a common problem in machine learning. It is a consequence of too extensive training of an algorithm on one dataset. The undesired effect is that the algorithm is memorizing the training examples instead of extracting concepts from them. Fortunately, we can easily detect if the algorithm tends to overfit by examining the produced

[2]Recall that words have been stemmed.

Table 10. Numbers of ASs in each AS class.

	Large ISPs	Small ISPs	Customer ASs	Universities	IXPs	NICs
ASs	44	5,599	11,729	877	33	332
%	0.2	30.1	63.0	4.7	0.2	1.8

classification rules.

Having the number of iterations fixed, we finally apply our classifier to the set of 19,537 ASs and calculate the ranking of the AS classes for them. For each AS we consider the highest ranked class as its predicted class. If the class with the highest rank value for an AS has the confidence value less than or equal to zero, then the classifier abstains from making a prediction since the given information is not sufficient to produce a reliable assignment. Overall, the classifier abstains from making a prediction for 923 ASs, which accounts for 4.7% of the total number of ASs in our dataset. In Table 10 we show the per category classification statistics. Among the classified ASs, 63.0% are Customers, 30.1% are small ISPs, 4.7% are Universities, 1.8% are NICs, 0.2% are ISPs, and 0.2% are large IXPs.

5.6 Validation

To validate our results, we employ a standard machine learning technique called *cross-validation*. Cross-validation is the process of splitting the training examples into two subsets. One then uses the first subset to train a new classifier and the second subset to validate the results of this new classifier.

From our main set of 1200 training examples, we randomly extract 1100 ASs and use varying size subsets of these 1100 ASs to train new classifiers. We validate the predictions of the new classifiers against the remaining 100 examples. We repeat the random selection process 400 times and for each iteration we compute the following evaluation metrics: 1) *accuracy*, which we define as the percentage of ASs for which the AS class with the highest rank value is their correct class. The disadvantage of this metric is that it checks only the top of the ranking, ignoring the remaining positions. To address this problem, we

use 2) *coverage*, which we define as the average position number of the correct class of an AS. For each AS, we number AS class positions incrementally starting from zero for the class with the highest positive confidence value. Thus, if all the predictions are correct the coverage is zero.

In Figures 5.6 and 5.6 we plot the average accuracy and coverage versus the size of the training set. As the size of the training set increases, the accuracy increases and the coverage decreases. For $|S| = 1100$ the accuracy reaches 0.781, e.g., 78.1%, and the coverage 0.251. The increasing accuracy trend suggests that for the training size of 1200 that we use in our final classification, the expected accuracy must be even higher. The low value of the coverage indicates that when the correct class is not of the top rank value, it is close to the top. More specifically, we find that for 97.7% of the predictions the correct class is in the top two positions of the ranking.

We next analyze the per class percentage of correct predictions. We find that for $|S| = 1100$ the percentage of correct predictions is on average: 100% for large ISPs, 100% for NICs, 100% for IXPs, 92.8% for Universities, 79.2% for Customer ASs and 72.1% for small ISPs. The actual distribution of ASs among the classes in our training set is: 684 Customer ASs, 401 small ISPs, 66 Universities, 36 NICs, 11 IXPs, and 2 large ISPs. The lower accuracy for customer ASs and small ISPs illustrates that these are the hardest classes to identify. We explain this effect by similarities between the characteristics of these two AS classes: 1) more than a half of the small ISPs appear to have the AS degree of 1 or 2, which is also typical for customer ASs; 2) some customer ASs, especially web hosting companies, advertise large numbers of different IP prefixes or large chunks of address space, which is also typical for ISPs.

In summary, we find that in the examined examples our classifier almost perfectly identifies large ISPs, NICs, IXPs and universities, while it also produces accurate predictions for customer ASs and small ISPs, which are the hardest to classify.

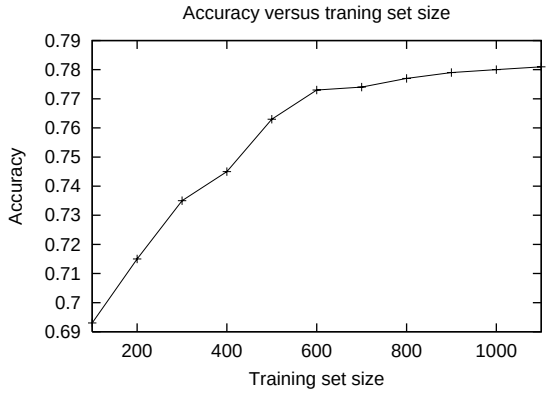


Figure 26. Accuracy of computed predictions versus the size of the training set.

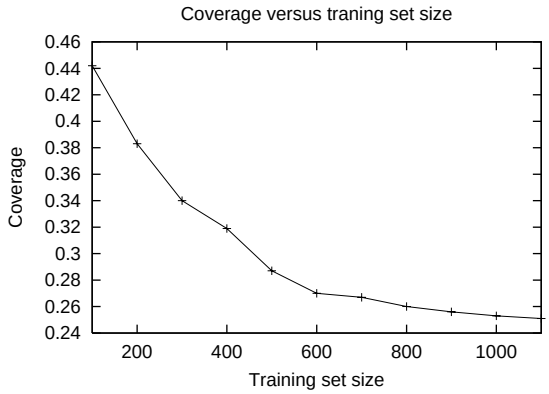


Figure 27. Coverage of computed predictions versus the size of the training set.

5.7 Conclusion

In this chapter, we established a classificfation of ASs required for expanding our understanding of the Internet infrastructure and for creating realistic models of its topology and evolution. We developed a novel classification methodology that we apply to an exhaustive set of AS data to obtain statistics on the different AS classes in the Internet. We validated our results and demonstrate that our classifier achieves accuracy of 78.1% in the examined data. Finally, to promote further analysis and to inspire development of better topology models, we built a website [30], where we make publicly available the AS attributes we used for our classification along with the AS class predictions.

CHAPTER 6

TOPOLOGY GENERATION

6.1 Introduction

Besides measured topologies, synthetically generated graphs are also important in conducting realistic simulations. Synthetic topologies are complementary to measured topologies and may be preferred when measured topologies are not 1) available, 2) of the desired size, or 3) complete enough. For example, a researcher may want to evaluate how the performance of a new routing protocol will scale with the size of the AS topology. In this case, it will use a topology generator to construct synthetic topologies of the desired size and will evaluate the performance of the new routing protocol on the synthetic topologies.

There are two main approaches to topology generation: the *macroscopic* and *microscopic* approaches. Macroscopic approaches identify important properties of measured topologies and try to reproduce these properties in synthetic graphs. For example, Power-Law Random Graphs (PLRG) [103] reproduce the power-law degree distribution observed in Internet topologies [104]. Microscopic approaches model the growth of a topology by accounting for local optimization criteria that shape the connectivity choices of individual nodes. For example, in the model by Fabrikant *et al*. [105], nodes optimize their connections to minimize the "last mile" connection cost and the transmission delays measured in hops. Both of these approaches have advantages and disadvantages. Macroscopic approaches overlook network design principles and thus they are not explanatory, i.e., they do not provide insight on why a topology has a particular structure. On the other hand, microscopic approaches try to model network design decisions, which is a challenging task since there do not always exist well-defined network design principles [106].

In this work, we introduce a macroscopic topology construction technique that models network topologies as annotated graphs, i.e., graphs with different link annotations. Different link annotations or equivalently link types are an inherent characteristic of many real

networks. For example, links in AS topologies represent different types of business relationships, like c2p, p2p, and s2s relationships. Links in router topologies have different transmission speed, e.g., as OC12, OC48, etc. Our topology generation approach identifies important properties of annotated graphs and reproduces these properties in synthetic annotated graphs. In contrast to previous macroscopic approaches, our technique 1) models annotation-aware graph properties[1], and 2) generates synthetic annotated graphs, i.e., synthetic graphs with realistic link annotations.

The main application that motivates this work is determining routing AS paths in synthetic AS topologies. Routing paths between ASs are determined by AS relationships. AS relationships result in the valley-free routing model, which states that every AS path has a hierarchical structure [4]. Given an AS topology annotated with inferred AS relationships, we can compute the policy-compliant AS path between any two ASs using a modified version of Dijkstra's algorithm [77]. On the other hand, without knowing AS relationships we are forced to assume shortest path routing, which is likely to lead to unrealistic results. It is well known that actual AS paths in the Internet are substantially longer [107, 108, 109, 110] than the shortest path. Unfortunately, the shortest path routing assumption is made by default in many simulation studies without further investigation. The reason is that existing topology generators do not model AS relationships, which makes it impossible to simulate path inflation effects.

The second reason for which modeling of link types is important is that it enables us to produce more accurate synthetic topologies. Different types of links are likely to exhibit different topological properties. For example, borrowing the terminology of [111], c2p links are more *radial*, in that they connect small degree to large degree ASs. However, p2p links are more *tangential* in that they connect ASs of similar degree. To capture this diversity of properties, it is necessary to build a topology generator that takes into account

[1]Annotation-aware graph properties are properties that take into account that a graph is composed of different types of links and that depend on the link-type diversity of the graph. In contrast, annotation-oblivious graph properties, like the degree distribution of a graph, treat all the links of a graph as equivalent, ignoring different types of links.

the existence of different types of links that may have different topological characteristics. Then, we can effectively model a wider range of topological properties than can currently available generators.

In this chapter, we first focus on the importance of modeling AS relationships in conducting accurate and realistic network simulations. We discuss the following three shortcomings of ignoring AS relationships: 1) AS paths are substantially shorter than in reality, 2) the traffic load on AS links and on individual ASs is substantially lower than in reality, and 3) the number of alternative AS paths available to an AS is substantially larger than in reality. We use simulation experiments to demonstrate these shortcoming and to show how they can effect commonly used performance evaluation metrics.

Next, we introduce our framework for modeling topologies with link annotations and for generating realistic topologies with realistic link annotations. We use our framework to develop an AS topology generator that besides producing realistic AS topologies it also assigns realistic c2p and p2p relationships. We compare generated AS topologies with measured AS topologies annotated with inferred c2p and p2p relationships. We find that our generator produces realistic AS topologies with c2p and p2p annotations that effectively reproduce c2p and p2p patterns observed in the Internet.

The rest of this chapter is organized as follows. In the next section we briefly review related work in the area of topology modeling and topology generation. Then, in section 6.3 we discuss and demonstrate shortcomings of ignoring AS relationships in conducting realistic network simulations. In section 6.4 we introduce the topological properties that enable us to model AS relationships. In section 6.5 we outline our framework for modeling these properties and for generating synthetic AS graphs. In section 6.6 we present our evaluation results. Finally, in section 6.7 we discuss future research directions.

6.2 Related Work

A large number of published works have focused on modeling Internet topologies and on developing realistic topology generators. The first topology generator that became widely known was introduced by Waxman [112]. Waxman generator is a variation of the classical Erdos-Renyi random graphs. Later, after it became evident that networks do not have a random structure, new generators like GT-ITM [113] and Tiers [114] emphasized the hierarchical structure of networks. Consequently, these topology generators were characterized as *structural*. In 1999, Faloutsos *et al*. discovered that the degree distributions of router- and AS-level topologies of the Internet follow a power-law. Structural generators failed to reproduce this power-law, which triggered a number of new topology generators that tried to achieve this goal. These newer topology generators can be classified into causality-aware and causality-oblivious. The first class includes the Barabasi-Albert (BA) [115] preferential attachment model and the model by Chang *et al*. [116] based on the idea of *highly optimized tolerance* [117]. These models grow a network by incrementally adding nodes and links into a graph based on some evolution process so that the resulting graph follows a power-law degree distribution. In the same family belongs the BRITE [118] topology generator, which employs the BA model to generate synthetic Internet topologies. On the other hand, causality-oblivious generators like PLRG [103], Inet [119] and the model by Gkantsidis *et al*. [120] try to match the power-law degree distribution of the Internet without accounting for different rules that might drive the evolution of the topology.

6.3 On simulation artifacts due to the shortest-path routing assumption

Routing policies reflect business agreements and economic incentives, and for this reason they are deemed more important than quality of service criteria and thus they take precedence in the route selection process. Consequently, suboptimal routing and inflated AS paths often occur. The study by Gao and Wang [109] used BGP data to measure the extent

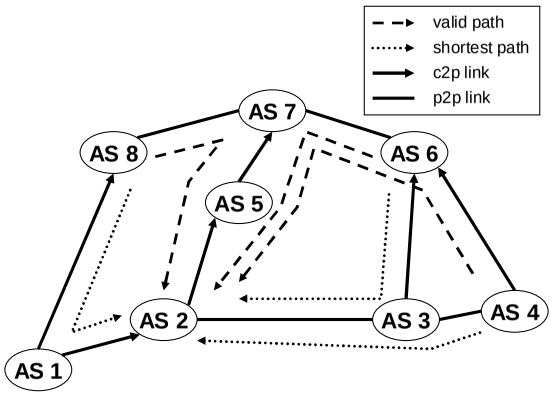


Figure 28. Example AS topology annotated with AS relationships. The topology is extracted from a real AS topology and the AS relationships are inferred using the heuristics in [2]. The dotted lines represent shortest paths between ASs 4, 6 and 8 to AS 2. The dashed lines represent policy compliant paths from the same sources to the same destination.

of AS path inflation due to valley-free and prefer-customer routing in the Internet. They found that at least 45% of the AS paths observed in BGP data are inflated by at least one AS hop and that AS paths can be inflated by as long as 9 AS hops.

Taking into account such inflation effects is important for conducting meaningful and realistic simulation studies. Consider, for example the AS topology in Figure 28 that we extracted from a real topology. Directed links represent c2p relationships that point towards the provider and undirected links represent p2p relationships. If we ignore AS relationships then the shortest paths from ASs 4, 6 and 8 to AS 2 are shown with dotted lines. On the other hand, if we account for AS relationships these paths are no longer valid. In particular, the path 4→3→2 transverses two p2p links; the path 6→3→2 transverses a p2c link followed by a p2p link; and the path 8→1→2 transverses a c2p link after having gone through a p2c link. All these paths violate the hierarchical structure of the valley-free model and thus are not used in practice. The paths actually used are the policy compliant paths marked with dashed lines.

The first effect of taking AS relationships into account is that paths become longer than the corresponding shortest paths. From a performance perspective, longer paths can affect metrics such as end-to-end (e2e) delay, server response time, jitter, convergence time and

Table 11. Total number of paths for each AS with AS relationships enabled and AS relationships disabled.

AS number		1	2	3	4	5	6	7	8
Number of paths	AS relationships disabled	12	13	16	15	13	15	15	13
	AS relationships enabled	12	9	10	8	8	7	9	6

others. To demonstrate this we simulated the topology in Figure 28 using BGP++. We used a single router for each AS and configured appropriate export rules between ASs according the guidelines discussed above. We set the delay of each link to 10 milliseconds and the bandwidth to 400kbps. Then, we configured exponential on/off traffic sources at ASs 4, 6 and 8 that send traffic to AS 2 at a rate of 500kbps. We run the simulation for 120 seconds; for the first 100 seconds we wait for routers to converge[2] and at the 100th second we start the traffic sources. We first measure the e2e delay between the sources and the destination under the following two configuration scenarios: 1) AS relationships disabled, and 2) AS relationships enabled.

In Figure 29 we depict the cumulative distribution function (CDF) of the e2e delays for the two scenarios. First, notice that the CDF corresponding to simulating AS relationships is skewed to the right, which means that there is a significant increase in the e2e delay. In particular, the average e2e delay with AS relationships enabled is 0.853 seconds, whereas without AS relationships it drops to 0.389 seconds. Besides this decrease in the e2e delay, note that in Figure 29 the CDF corresponding to simulating AS relationships is much smoother than the second CDF, which exhibits a step-wise increase. This difference shows that the e2e delay with AS relationships enabled exhibits a much higher variability compared to ignoring AS relationships. This variability is likely to affect other performance metrics like jitter and buffer occupancy.

A second implication of policy routing is that ASs have fewer alternative AS paths. For example, in Figure 28 when ignoring AS relationships AS 7 has three (one through each neighbor) disjoint paths to reach destination 2. One the other hand, with AS relationships

[2]Typically routers take much less than 100 seconds to converge, but to be conservative we used a longer period.

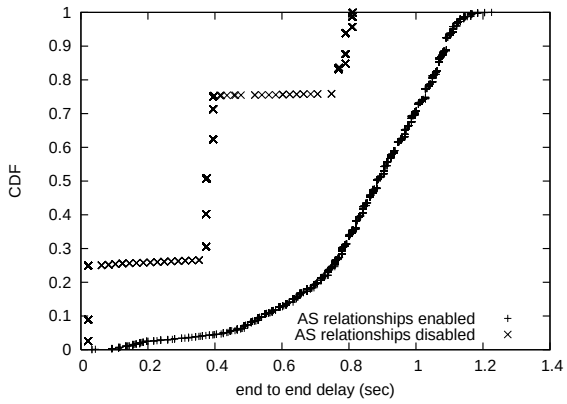


Figure 29. CDF of e2e delay between traffic sources and destination.

enabled, AS 7 has only one possible path through AS 5, since the other two paths are not valley-free. In Table 11, we show the total number of paths we find in the BGP tables of the 8 simulated ASs. The consistent decrease in the number of paths when AS relationships are enabled highlights that *ignoring AS relationships increases the path diversity* of the ASs in a simulation. Path diversity is a property that can play an important role in simulations measuring such properties as network resilience, vulnerability to attacks, links and router failures, load balancing, multi-path routing, convergence of routing protocols, and others.

An additional implication of policy routing is that due to the smaller number of available AS paths as compared to shortest path routing, some ASs or AS links are likely to receive greater load than when assuming shortest path routing. For example, in Figure 28 the dashed paths share the links from AS 7 to AS 5. On the other hand, when assuming shortest path routing the three paths are mostly disjoint, with only the link between AS 3 and AS 2 being shared by two flows. Thus, AS links and ASs will receive greater load than when ignoring AS relationships, which is likely to produce more packet drops, increased delay, congestion, router failures, and other important events. In Table 12 we list the average bandwidth in our simulations for each of the three flows with and without AS relationships enabled. We find that because of the increased load on the links between AS 7 and AS 5 the average bandwidth of the three flows decreases substantially.

Table 12. Average bandwidth per flow with AS relationships enabled or disabled.

Flow		$4 \rightarrow 2$	$6 \rightarrow 2$	$8 \rightarrow 2$
Bandwidth (Kbps)	AS relationships disabled	202	196	397
	AS relationships enabled	113	164	121

In summary, we highlight that ignoring AS relationships produces the following important artifacts:

- AS paths are substantially shorter than in reality.
- The number of alternative AS paths available to an AS is substantially larger than in reality.
- The traffic load on AS links and on individual ASs is substantially lower than in reality.

6.4 AS Relationships-aware topological properties

To represent AS topologies annotated with AS relationships, we use a graph G with edges annotated as c2p or p2p. We ignore s2s relationships, since they typically account for a very small fraction of the total number of edges[3]. c2p edges are *directed* from the customer AS to the provider AS, while p2p edges are *undirected*. We call such graphs *G annotated graphs*. Annotated graphs can be used to represent other link characteristics of interest like link bandwidths or latencies. Here, we describe our modeling framework with respect to AS relationship annotations, however the same methodology is directly applicable for modeling other types of annotations.

The topological property that most state-of-the-art topology generators reproduce is the degree distribution of a network. For our topology generator we choose to reproduce the following three properties.

1. **Degree distribution.** Along the lines of existing topology generators, we reproduce

[3]In our experiments in Chapter 4 the inferred s2s relationships accounted for only 0.46% of the total number of edges in the AS topology.

the degree distribution of the AS topology of the Internet. The degree distribution tells us how many nodes of each degree are in the network.

2. **Annotation-degree distributions.** The degree distribution of an AS topology does not convey any information about the different types of AS relationships in a topology. To take into account AS relationships we look at the number of customers, providers, and peers each AS has. We define the *customer-degree* d_{p2c} of an AS as the number of its customers, the *provider-degree* d_{c2p} as the number of its providers, and the *peer-degree* d_{p2p} as the number of its peers. We collectively refer to d_{p2c}, d_{c2p} and d_{p2p} as *annotation degrees*. The second property we select to reproduce is the annotation-degree distributions of an AS topology. These distributions are a natural generalization of the degree distribution of a network that take into account the presence of different types of AS relationships. The customer-degree distribution tells us how many nodes with a specific number of customers are in the network. Similarly, the provider- and peer-degree distributions tell us how many nodes with a specific number of providers, and peers, respectively, are in the network.

3. **Annotation-degree correlations.** The annotation-degree distributions do not tell us anything about the correlations between these degrees, i.e, how many customers, providers and peers a specific AS has. Correlations between different annotation degrees appear often in the Internet. For example, large tier-1 ASs typically have a large number of customers, i.e., large d_{p2c}, no providers, i.e., zero d_{c2p}, and a small number of peers, i.e., small d_{p2p}. On the other hand, medium size ISPs have a small set of customers, several peers, and few providers. Note, that simply ignoring these correlations can lead to graphs that follow the previous two properties, but have artifacts, like high degree nodes with many providers.

The exact correlations between the annotation degrees of an AS are captured in the joint distribution $P(d_{p2c}, d_{c2p}, d_{p2p})$, which is defined as the number $n(d_{p2c}, d_{c2p}, d_{p2p})$ of nodes in

the network with d_{p2c} customers, d_{c2p} providers, and d_{p2p} peers over the total number of nodes n:

$$P(d_{p2c}, d_{c2p}, d_{p2p}) = n(d_{p2c}, d_{c2p}, d_{p2p})/n.$$

We call this distribution the *joint annotation-degree distribution* (JADD). JADD is a multivariate distribution and its marginals[4] are the annotation-degree distributions, i.e., our second property. From JADD we can also derive the AS-degree distribution simply by summing the annotation degrees of a node. Consequently, JADD is a union of the three properties we have discussed thus far.

We construct a synthetic annotated graph, starting with the JADD of a measured AS topology annotated with inferred AS relationships. Then, we rescale JADD to the desired size, reflecting the size of the target, synthetic AS topology. Given the rescaled JADD, we can introduce p2p and c2p edges *independently* using existing techniques tailored for annotation-oblivious topologies. In section 6.5.2 we describe two techniques for introducing c2p and p2p edges. In general, when introducing c2p and p2p edges it is possible to impose additional topological properties, which the c2p and p2p edges must follow. Our first technique in section 6.5.2 does not impose any additional properties, it simply reproduces the JADD of real AS topologies. Our second construction technique imposes the additional property that the introduced c2p and p2p links must reproduce correlations between AS degrees of neighboring nodes. To illustrate this, consider that the three properties we have discussed thus far do not provide any information on how nodes of different degrees connect. For example, tier-1 ASs have typically p2p relationships with other tier-1 ASs, while they do not have p2p relationships with smaller ASs. This means that large degree nodes connect with other large degree nodes using p2p links. On the other hand, smaller degree nodes have p2p links with other nodes of similar size but not with large degree nodes. To capture how nodes of different degrees connect, we use the joint distribution of the degrees

[4]Given three jointly distributed random variables X, Y and Z, the marginal distribution of X is the probability distribution of X ignoring information about Y and Z, typically calculated by summing or integrating the joint probability distribution over Y and Z.

of connected nodes. The joint degree distribution (JDD) $P_{p2p}(d_1, d_2)$ of p2p edges is equal to the number $n_{p2p}(d_1, d_2)$ of p2p edges between nodes of degrees d_1 and d_2 over the total number of p2p edges in the graph. Thus, the JDD of p2p edges describes the degrees of the nodes that are connected with p2p edges. Similarly, the JDD of c2p edges P_{c2p} gives us the degrees of the nodes that are connected with c2p edges. Thus, the JDDs of p2p and c2p edges is the last property that our framework reproduces. to JADD. Note that similarly to JADD the JDDs of p2p and c2p edges are multivariate distributions.

6.5 AS Topology Generator

In this section we outline our framework for modeling and reproducing the JADD of real AS topologies. We first describe how to model the JADD of a real AS topology annotated with inferred AS relationships. Our topology generation scheme proceeds in two phases. In the first phase, given the number of nodes N in the target graph we produce N degree triplets d^i_{p2c}, d^i_{c2p} and d^i_{p2p}, $1 \leqslant i \leqslant N$, such that the JADD of these triplets follows the JADD of real AS topologies. In the second phase given the N degree triplets we contract an annotated graph.

6.5.1 Modeling JADD

We model JADD using *copulas* [121], an powerful statistical tool that fully quantifies the dependence among multiple random variables. In contrast to other well-known correlation metrics, like Pearson's coefficient, Kendall's tau or Spearman's rho, copulas do not provide a single scalar value but a function that captures complex correlations and fine-grained details of the dependence structure.

According to Sklar's theorem [122], any continuous[5] 3-dimensional multivariate cumulative distribution function (CDF) F can be written in the form:

$$F(x_1, x_2, x_3) = C(F_1(x_1), F_2(x_2), F_2(x_3)), \tag{6}$$

[5]Degree distributions are inherently discrete distributions. Nevertheless, they can be turned into continuous by adding a random uniform noise $U(-0.5, 0.5)$ to each degree sample.

where F_1, F_2 and F_3 denote the marginal CDFs. The function C is called a copula and has uniform distributed marginals in $[0,1]^3$. Given the copula function and the marginal CDFs F_1, F_2 and F_3, we can determine the joint distribution F using equation 6. Thus, copulas have two important properties: 1) given the marginals they *fully* describe the joint distribution F, and most importantly, 2) they enable the practitioner to model the dependence structure *independently* of the marginal distributions.

Modeling marginal distributions is a fairly easy task, since there exists a wealth of statistical methods and distributions for matching univariate samples. To find the appropriate marginal distributions, we constructed an AS topology from RouteViews [49] data. We downloaded a BGP table from the collector `route-views2.oregon-ix.net` on 07/18/2005 and extracted AS links, ignoring private AS numbers and AS sets. We inferred c2p and p2p relationships using the heuristics we developed in Chapter 4. Thus, we derived a real AS topology annotated with c2p and p2p relationships. From this topology, we extracted the customer-, provider-, and peer-degree distributions and evaluated alternative fitting approaches. We model these three distributions using spline smoothing. Spline smoothing is a nonparametric estimator of a function that produces a curve that passes through or near the points of a distribution keeping the curve as smooth as possible. We fit splines to our data using the estimation technique outlined in [123], which is implemented in the generalized smoothing splines (gss) module [124] of the R project[6]. Using the fitted models we can then generate random numbers that follow the customer-, provider- and peer-degree distributions of the AS topology of the Internet. Thus, the first step to construct a graph with N nodes is to generate N customer, N provider, and N peer degrees from the corresponding fitted models.

Next, we model the copula by resampling historical correlation data. We first construct a set with all the degree triplets of the collected AS topology. Then, we sample N degree triplets from this set. These N triplets include information on both the actual annotation

[6] R is a language and environment for statistical computing.

degrees and the correlations between them. We extract the correlation information by mapping each triplet into the $[0,1]^3$ space. To do so, we replace each annotation degree with its rank normalized by $1/N$. The resulting triplets $(u^i_{p2c}, u^i_{c2p}, u^i_{p2p})$ reflect the correlations between the annotation degrees in the original AS topology and are independent of the actual annotation degrees. Each of the u^i_{p2c}, u^i_{c2p} and u^i_{p2p} is uniformly distributed in $[0,1]$.

Finally, we combine the $(u^i_{p2c}, u^i_{c2p}, u^i_{p2p})$ triplets with the generated annotation degrees to derive the final degree triplets that follow the JADD of the original topology. Each $(u^i_{p2c}, u^i_{c2p}, u^i_{p2p})$ triplet is resolved into a degree triplet $(d^i_{p2c}, d^i_{c2p}, d^i_{p2p})$ by mapping each of u^i_{p2c}, u^i_{c2p}, and u^i_{p2c} into the inverse CDF of the corresponding annotation degrees. For example, we map u^i_{p2c} into d^i_{p2c}, where d^i_{p2c} is the value of the inverse CDF of the generated customer-degrees at the point u^i_{p2c}. Thus, we derive the N annotation-degree triplets $(d^i_{p2c}, d^i_{c2p}, d^i_{p2p})$, $1 \leqslant i \leqslant N$, that follow the JADD of the original topology.

6.5.2 Generating Annotated AS topologies

Given the N annotation-degree triplets, there are several methods that can be used to construct a random annotated graph. This is because we can focus on the p2p and c2p links independently and use existing annotation-oblivious techniques to construct p2p and c2p links. In this section we describe two such construction techniques. The first is a generalization of the random matching method used in the PLRG topology generator [103]. Using our generalization we construct a graph that follows the JADD of real AS topologies. The second method is a generalization of the *pseudograph* approach [125], which enables to construct random graphs that reproduce the JDD of the links in a topology. We generalize this method to reproduce the JDD of p2p and c2p links as well as the JADD of real AS topologies.

Our first method works as follows:

1. For each of the generated triplets, we introduce a node with d^i_{p2c} customer stubs, d^i_{c2p}

provider stubs, and d^i_{p2p} peer stubs[7].

2. We connect stubs by performing one random matching between p2p stubs and a second random matching between c2p and p2c stubs. If the number of p2p stubs is odd or if the number of c2p stubs is not equal to the number of p2c stubs, then some stubs will remain unmatched. We ignore such stubs.

3. Random matching can lead to disconnected components, self-loops, and multi-edges. We extract the final graph by extracting the largest connected component and by removing self-loops and multi-edges.

The ordinary random matching method used in the PLRG topology generator creates an undirected graph ignoring AS relationships, i.e., all the stubs are of the same type. We extend this algorithm by using different types of stubs to account for customer, provider, and peer edges. Then, we perform two random matchings[8] between stubs of the same type and of compatible directions. A limitation of the PLRG topology generator is that it produces graphs that contain self-loops and multi-edges. Self-loops and multi-edges usually appear on or between large degree ASs. This is because large degree ASs have many stubs and thus it is quite likely that the random matching will match two stubs that belong to the same AS or more then two stubs between two ASs. In our generalization this two problems are diminished. This is because edges of high degree ASs are mainly customer edges that can only connect to customer ASs, which usually are of small degree and not to other high degree ASs.

The pseudograph approach described in [125] ignores AS relationships and produces an undirected graphs that follows the JDD of the edges in the modeled topology. The algorithm works as follows. Given the number $m(d_1, d_2)$ of edges between d_1- and d_2-degree nodes first prepare a list of $m(d_1, d_2)$ disconnected edges with edge-ends labeled

[7] A stub is a half edge that is adjacent to a single node. By connecting two stubs we get a regular edge.

[8] More generally, we need to perform as many random matchings as the number of different link annotations. Thus, if we also want to model s2s links, then we can add s2s stubs and perform a random matching between them.

by d_1 and d_2. Next, from the list of edge-ends that are labeled with d, randomly select d edge-ends to construct a node of degree d. We can easily adapt this method to reproduce the JDD of c2p and p2p edges. Given the N degree triplets generated from the JADD model, we first estimate the number of c2p and p2p edges between d_1- and d_2-degree nodes. To do so we model the c2p and p2p JDD distributions using the same methodology we outlined for JADD and produce c2p and p2p degree pairs. Then, for each degree pair we introduce c2p or p2p disconnected edges with their edge-ends labeled with their expected degree. Finally, for each degree triplet $(d^i_{p2c}, d^i_{c2p}, d^i_{p2p})$ of total degree $d_i = d^i_{p2c} + d^i_{c2p} + d^i_{p2p}$, we randomly select d^i_{p2c} provider edge-ends, d^i_{c2p} customer edge-ends, and d^i_{p2p} peer edge-ends from the list of edge-ends labeled with d_i. Thus, we construct a node of degree d_i that has d^i_{p2c} customers, d^i_{c2p} providers, and d^i_{p2p} peers. Similarly to the PLRG method, the pseudograph method can also lead to self-loops, multi-edges, or disconnected components. We get the final graph by removing self-loops, multi-edges, and by extracting the largest connected component.

6.6 Evaluation

In this section we describe a set of graph metrics, which we use to evaluate the similarity between generated topologies and measured topologies.

1. **AS-degree distribution** The AS-degree distribution is one of the properties our framework models. Thus, the generated graphs should follow the degree-distribution of real AS topologies.

2. **Annotation-degree distributions** The customer-, provider-, and peer-degree distributions are also taken into account in our modeling framework, thus their shape should be reproduced in synthetic AS topologies.

3. **Annotation-degree correlations** This is the last property that our framework captures. To compare the JADD of two topologies we plot scatterplot matrices of the

customer-, provider- and peer-degrees of the two topologies. A scatterplot matrix of three variables is a matrix of scatterplots with one scatterplot for each combination of two variables. Thus, there are six scatterplots in total that illustrate correlations between the three variables.

6.6.1 Evaluation Results

Among the two approaches we described in section 6.5.2, we provide evaluation results only for the pseudograph approach. We find that the matching approach though it successfully recreates the JADD of measured AS topologies, it suffers from the limitation that it does not reproduce p2p relationships between large degree ASs, which is known to happen in reality. The reason is that medium degree ASs have more p2p links than large degree ASs. Thus, by performing random matching between p2p stubs it is more likely to get a p2p edge between a medium degree AS and a large degree AS than to get a p2p edge between two large degree ASs. Thus, random matching results in more p2p links between large degree ASs and medium degree AS than between large degree ASs, which is unrealistic. This problem does not appear with the pseudograph approach, since this approach takes into account the JDD of p2p and c2p edges. Thus, it forces p2p edges to lie between ASs of large degree as it happens in reality.

We use the pseudograph approach to generate a topology of similar size with the measured AS topology we extracted from RouteViews. In Table 13 we list the number of nodes, edges, p2p edges, and c2p edges in the two graphs. We observe that the values of these metrics are very similar in the two graphs. These metrics do not give us any insight on the structure of the two graphs, but they are only rough indicators of how similar the two graphs are.

In Figure 30 we plot the total-degree distribution of the two graphs and in Figure 31 the customer-, provider-, and peer-degree distributions. The empty points show the distributions observed in the measured AS topology, whereas the solid points depict the distributions in the synthetic topology. We observe that in the two figures the distributions for

Table 13. Number of nodes, edges, c2p edges and p2p edges in the measured and in the synthetic AS topology.

Number of	nodes	edges	c2p edges	p2p edges
Measured Topology	19,036	40,115	36,188	3,927
Synthetic Topology	19,141	40,727	37,350	3,377

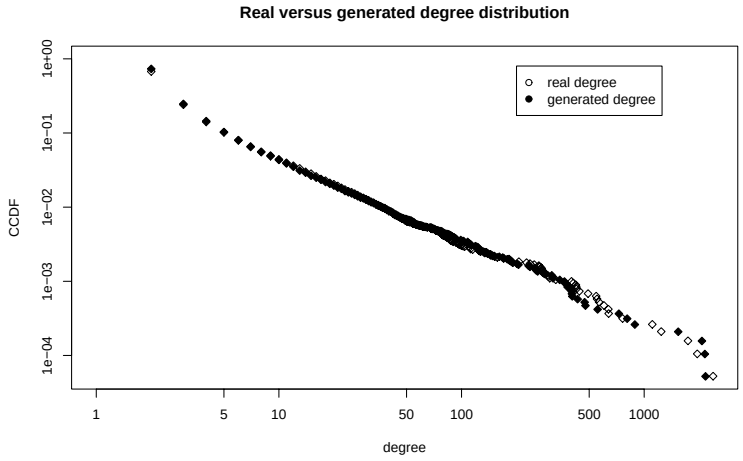


Figure 30. Degree distribution of measured graph and of synthetic graph of similar size.

the synthetic graph follow closely the corresponding distributions for the measured graph. This highlights that we can effectively reproduce the customer-, provider-, peer-, and total degree distributions using our modeling framework. In the second figure we also observe that the customer-degree distribution has the longest tail, followed by the peer-degree distribution, followed by the provider-degree distribution. Thus, there is a large variability in the number of customers. On the other end, there is much smaller variability on the number of providers with most ASs having one or two providers. The maximum number of customers in the measured topology is 2,384, the maximum number of peers is 434, and the maximum number of providers is 17. These distributions confirm that different types of relationships can have radically different properties as we argued in the introduction.

In Figure 32 we plot the matrix scatterplot of the provider-, peer-, and customer-degrees of the real AS topology. The *x*-axis of the subplots in the first, second, third column correspond to the provider-, peer-, and customer-degrees, respectively. Similarly, the *y*-axis of the subplots in the first, second, and third row correspond to the provider-, peer-, and

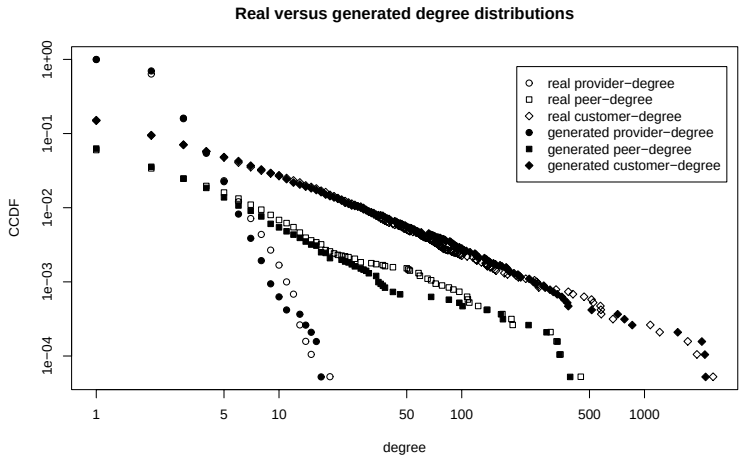


Figure 31. Customer-, provider-, peer-degree distributions of measured graph and of synthetic graph of similar size.

customer-degrees, respectively. Thus, for each combination of annotation degrees there are two scatterplots at the transposed positions of the matrix. The subplots on the left of the first two rows show the dependence between the peer-degree and the provider-degree. Both of the subplots are noisy without a clear indication of a dependence pattern. The two plots in the top right and in the bottom left positions of the matrix illustrate the dependence between the provider-degree and the customer-degree. We first note the pattern that ASs with many customers have very few or no providers[9]. This pattern confirm the empirical knowledge that large tier-1 ISPs have many customers and no providers. The last two subplots on the right of the bottom two rows illustrate the dependence between the customer-degree and the peer-degree. These two plots exhibit the following patterns. First, we see that ASs with many customers have very few peers. This indicates that the largest ISPs in the Internet have p2p relationships with very few other ISPs. Secondly, as we move to smaller customer-degrees we see that ASs start having larger peer-degrees. This observation is consistent with the intuition that medium size ISPs have more open peering policies than large tier-1 ISPs. Finally, ASs with very few customers have very few peers. These ASs are likely small or leaf ASs in the periphery of the AS topology, e.g., company networks,

[9]Observe that the subplot in the right of the first row has very few points for large customer-degrees. This is because most ASs with large customer-degrees have zero provider-degree.

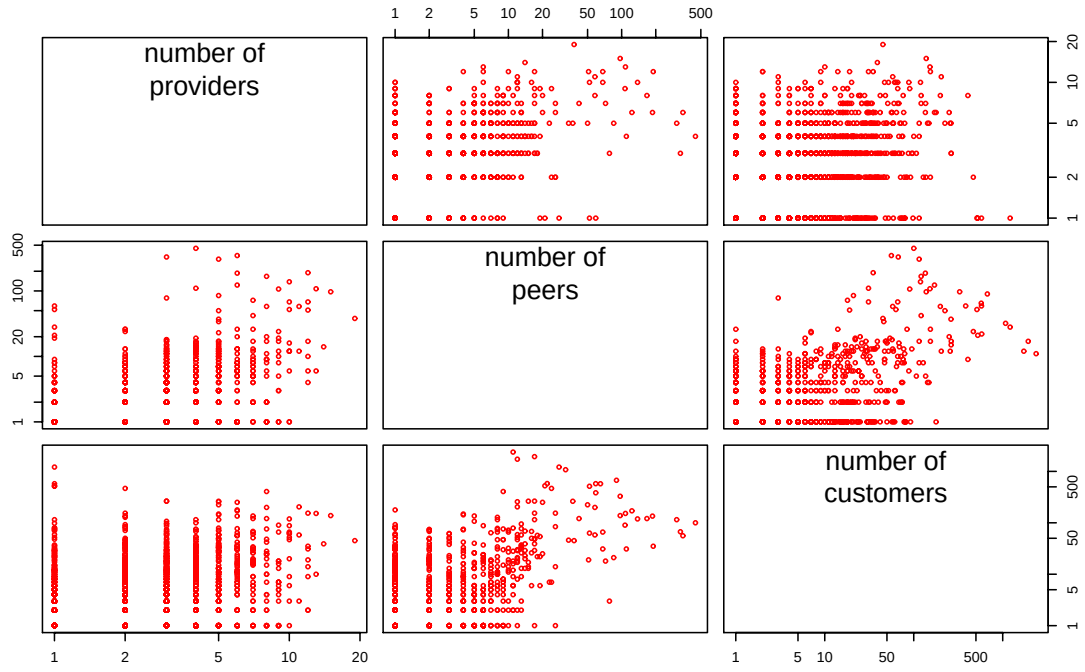


Figure 32. Matrix scatterplot of the provider-, peer-, and customer-degrees of the ASs in the measured AS topology.

which do not typically engage in p2p relationships with other ASs. In Figure 33 we illustrate the same matrix scatterplot for the customer-, provider-, and peer-degrees of the synthetic topology. We observe that the patterns that appear in the measured topology are also reproduced in the synthetic topology. The corresponding subplots in Figures 33 and 32 are very similar, which highlights that our framework effectively reproduces correlations between the annotations-degrees. We note that the subplots are not identical, but exhibit small differences, which is a natural consequence of the random construction method.

Overall, we find that the synthetic topology effectively reproduces the examined evaluation metrics. Thus, we verify that our framework captures the classical degree distribution of the AS topology and in addition it assigns realistic AS relationship annotations, which closely reproduce AS relationships patterns observed in the Internet.

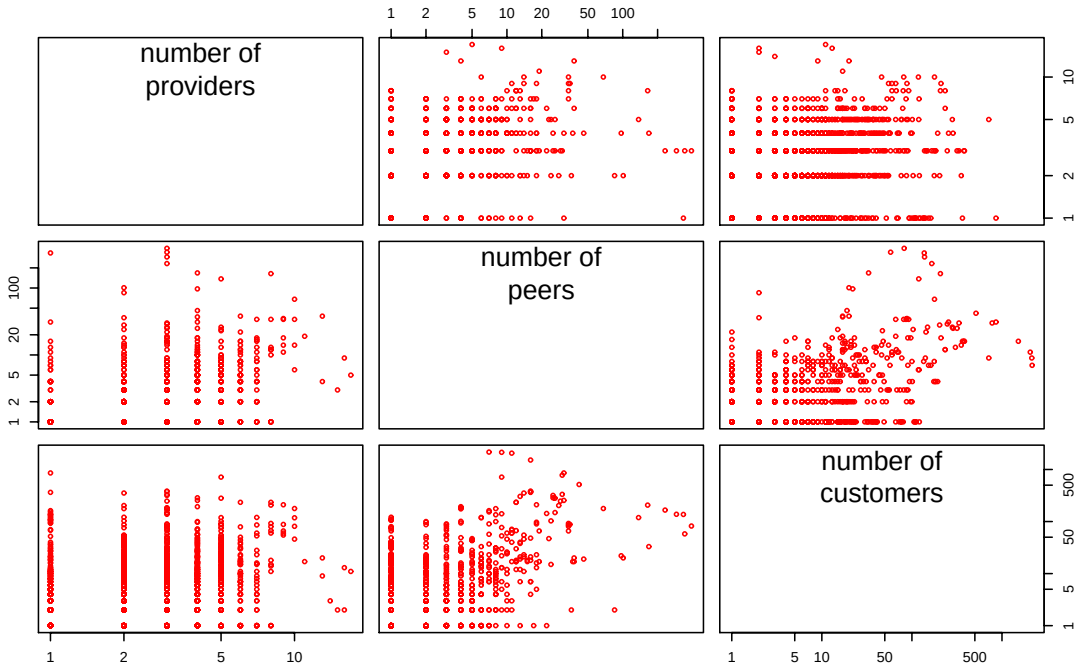


Figure 33. Matrix scatterplot of the provider-, peer-, and customer-degrees of the ASs in the synthetic AS topology.

6.7 Conclusion

We highlighted the problems that the Internet community is facing because of the lack of AS relationship models and discussed its implications on conducting realistic and reliable simulations studies. We used simulation experiments to demonstrate that ignoring AS relationships can change a wide range of performance metrics, which are typically used by researchers in performance evaluation studies. We draw motivation from our findings first to note that shortest path routing is a questionable assumption that should be used with great care in simulation studies and secondly to introduce a novel topology generation framework. Our framework improves the state-of-the-art by producing AS graphs that follow the degree distribution of the Internet as well as two new properties: 1) the annotation-degree distributions, and 2) the joint annotation-degree distribution (JADD). These two properties extract information about the number of customers, providers, and peers of ASs in the Internet and enable us to create synthetic AS graphs with realistic customer, provider, and

peer assignments. We use powerful statistical tools to model these properties on real AS topologies and we introduce an algorithm to reproduce these properties in synthetic AS topologies. Our evaluation results demonstrate that our framework effectively reproduces these properties and that the resulting synthetic topologies exhibit the same annotation patterns we observe in the Internet.

CHAPTER 7

CONCLUSIONS

7.1 Summary of Contributions

This dissertation derived necessary tools and data for conducting realistic BGP simulations. We first developed a BGP simulator, then we introduced measurement and inference methodologies, and finally developed a novel topology modeling framework. Our contributions can be summarized as follows:

1. In Chapter 2 we introduced BGP++, a BGP simulator designed to realize detailed and large-scale simulations. BGP++ design approach was to incorporate a popular open-source BGP router into a packet-level network simulator. Thus, we effectively addressed the need for detail. Moreover, we extended BGP++ to support parallel and distributed simulation techniques and we developed a compact routing table data structure to reduce the memory consumption. Using BGP++ we demonstrated large-scale simulations of thousands of BGP routers. Our work on BGP simulation has resulted in the following publications [126, 127, 128].

2. In Chapters 3, 4, and 5, we introduced techniques to mine vital data of the inter-domain infrastructure necessary for conducting realistic simulations. In particular, we derived methodologies for measuring AS topologies, for inferring AS relationships, and for inferring AS types. These studies have resulted in the following publications [70, 2, 129, 130, 131].

3. In Chapter 6, we introduced a framework for modeling networks and link characteristics as annotated graphs. We identified important annotation-aware properties and developed an algorithm to generate synthetic graphs with realistic annotations. Though, our framework is generic, we used it to model AS topologies annotated with

AS relationships and to derive an AS topology generator. This work has resulted in the following publications [79, 132].

The main motivation for our research was to enhance the integrity of BGP performance evaluation methods, however our results are not only of simulation interest. Rich Internet data are also important for identifying limitations of the routing architecture, for mapping the requirements for future routing architectures, and for many other practical and theoretical applications. For example, the inferred AS relationships and the derived AS classes have been used by network economists to analyze the Internet economy. Moreover, our modeling framework is generic and can be used for modeling arbitrary networks, like biological or social networks.

7.2 New Directions

Our work opens new directions in the areas of routing, simulation, measurements, and modeling.

7.2.1 Routing

BGP++, the acquired data, and our topology generator can and have been helpful in BGP simulation studies. BGP++ enables the practitioner to perform detailed simulations of thousands of BGP routers, which are necessary for studying BGP, BGP modifications, and new routing protocols. For example, we are presently using BGP++ to evaluate the impact of BGP's slow converge on DNS anycast servers. Moreover, BGP++ has been used to study the interplay between overlay routing and BGP routing [33] and the performance of BGP over satellite links [133]. In the future, we intend to use BGP++ to evaluate BGP-based Distributed Denial of Service (DDoS) attack mitigation techniques. The measured or inferred data are useful for improving our understanding of the BGP infrastructure, for analyzing Internet routing, and for configuring realistic BGP simulations. For example, the inferred AS relationships and AS types have been used for creating synthetic BGP routing tables and for mapping IP addresses in traffic traces to different types of users, respectively.

In general, there is wide range of routing applications in which BGP++, the retrieved data, and our topology generator, can and we hope will be useful.

7.2.2 Simulation

The BGP++ development methodology can set a example in developing simulation software. In particular, using UNIX protocol implementations to derive corresponding simulation software is a minimally explored method to developing detailed network simulators. This direction has two key benefits: 1) it allows us to introduce detailed protocol implementations in simulators without the extensive effort of developing a protocol from scratch and 2) it enables us to develop further protocol modifications once and immediately apply them to both the simulator and the UNIX version of the software. A promising path is the development of interfaces with which we can seamlessly or with minimal effort use networking software on top of simulated hosts or routers. This requires the development of kernel-like interfaces on the simulated nodes that would for example simulate system calls or the BSD sockets interface.

7.2.3 Measurements

In the Internet measurements area, our AS taxonomy methodology can be extended to compute more interesting classifications. Methodologically there is a wide spectrum of machine-learning and information retrieval algorithms that can be used to analyze Internet data. Our classification approach sits on the one end of the spectrum, in which the classification rules are derived automatically without any expert knowledge. One the other end of the spectrum, expert systems rely entirely on experts embedding their knowledge into the classification rules. Both of these approaches have a number of advantages and disadvantages, which rises the question of how we can combine these techniques to derive more accurate classification methodologies. An additional direction is to derive more fine-grained taxonomies that would identify for example web-hosting ASs or other AS types of interest. For this purpose, we need richer and more relevant data, from which we extract

patterns associated with different types of ASs.

7.2.4 Modeling

The generic nature of our topology modeling framework opens new paths to modeling topologies annotated with interesting link attributes. For example, we could use our framework to model router-level topologies annotated with link bandwidths and then generate synthetic router-level topologies with realistic bandwidth annotations. A natural extension of our framework is to adapt it so that we can also model node annotations. Node annotations, could represent other interesting network characteristics, like router locations, AS types, or router vendor models.

7.3 Concluding Remarks

Concluding we highlight that to support future research effors we have made publicly available the following open-source software and data repositories:

1. In Chatper 2 we introduced BGP++. BGP++ and its supporting tools can be downloaded from [38].

2. For our work in Chapter 3 on mapping AS topologies using BGP updates, we did not develop a data repository. The reason is that the same methodology was independently explored by Zhang *et al*. [69], who provided an open repository of AS topologies constructed from BGP updates. These AS topologies can be downloaded from [134].

3. In collaboration with CAIDA we have automated the c2p, p2p, and s2s inference heuristics we described in Chapter 4. We compute and archive AS relationships on a weekly basis and make our data publicly available on the web [84].

4. The AS data we used to classify ASs in Chapter 5 and the inferred AS types can be found in [30].

5. Currently, we are implementing a public release of the topology generator outlined in Chapter 6 and in the future we will make it publicly available.

REFERENCES

[1] "Zebra System Architecture Figure." http://www.zebra.org/zebra/System-Architecture.html#System%20Architecture. Date accessed: 05/11/2006.

[2] X. Dimitropoulos, D. Krioukov, B. Huffaker, kc claffy, and G. Riley, "Inferring AS relationships: Dead end or lively beginning?," in *Proceedings of 4th Workshop on Efficient and Experimental Algorithms (WEA' 05)*, May 2005.

[3] Y. Rekhter, T. Li, and S. Hares, "RFC 4271, A Border Gateway Protocol 4 (BGP-4)," Jan. 2006.

[4] L. Gao, "On inferring Autonomous System relationships in the Internet," in *IEEE/ACM Transactions on Networking*, December 2001.

[5] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfiguration," in *Proceedings of ACM SIGCOMM*, August 2002.

[6] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet routing convergence," in *Proceedings of ACM SIGCOMM*, pp. 175–187, 2000.

[7] C. Labovitz, G. R. Malan, and F. Jahanian, "Internet routing instability," *IEEE/ACM Transactions on Networking*, vol. 6, no. 5, pp. 515–528, 1998.

[8] S. Kent, C. Lynn, and K. Seo, "Secure Border Gateway Protocol (Secure-BGP)," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 582–592, April 2000.

[9] T. Griffin and G. Willfong, "A safe path vector protocol," in *Proceedings of IEEE INFOCOMM*, March 2000.

[10] T. G. Griffin, A. D. Jaggard, and V. Ramachandran, "Design principles of policy languages for path vector protocols," in *Proceedings of ACM SIGCOMM*, August 2003.

[11] G. Huston, "BGP routing table analysis reports." http://bgp.potaroo.net/. Date accessed: 05/11/2006.

[12] N. Spring, R. Mahajan, and T. Anderson, "Quantifying the causes of path inflation," in *Proceedings of ACM SIGCOMM*, August 2003.

[13] L. Gao and F. Wang, "The extent of AS path inflation by routing policies," in *Proceeding of Global Internet 2002*, pp. 188–195, Nov. 2002.

[14] H. Tangmunarunkit, R. Govindan, and S. Shenker, "Internet path inflation due to policy routing," in *Proceeding of SPIE ITCom 2001, Denver 19-24*, pp. 188–195, Aug. 2001.

[15] V. J. Bono, "7007 Explanation and Apology." http://www.merit.edu/mail.archives/nanog/1997-04/msg00444.html. Date accessed: 05/11/2006.

[16] "WorldCom's network struggles." http://www.networkworld.com/news/2002/1003dpwc.html, Oct 2002. Date accessed: 05/11/2006.

[17] S. Ramachandra, Y. Rekhter, R. Fernando, J. G. Scudder, and E. Chen, *Graceful Restart Mechanism for BGP*. IETF, Internet Draft, Jun. 2002.

[18] P. Verkaik, A. Broido, kc claffy, R. Gao, Y. Hyun, and R. van der Pol, "Beyond CIDR aggregation," February 2004. CAIDA Technical Report TR-2004-1.

[19] D. Pei, X. Zhao, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, "Improving BGP convergence through consistency assertions," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM-02)*, vol. 2, (Piscataway, NJ), pp. 902–911, IEEE Computer Society, June 2002.

[20] A. Bremler-Barr, Y. Afek, and S. Schwarz, "Improved bgp convergence via ghost flushing."

[21] D. Pei, M. Azuma, N. Nguyen, J. Chen, D. Massey, and L. Zhang, "BGP-RCN: Improving BGP Convergence Through Root Cause Notification," Tech. Rep. TR-030047, UCLA Department of Computer Science, Oct 2003.

[22] I. Castineyra, N. Chiappa, and M. Steenstrup, *The Nimrod Routing Architecture*. IETF, RFC 1992, 1996.

[23] F. Kastenholz, *ISLAY: A New Routing and Addressing Architecture*. IRTF, Internet Draft, 2002.

[24] X. Yang, "NIRA: A new Internet routing architecture," September 2004. Ph.D. Thesis, Massachusetts Institute of Technology.

[25] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica, "Towards a next generation inter-domain routing protocol," in *Proceedings of ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-III)*, November 2004.

[26] S. Agarwal, C.-N. Chuah, and R. Katz, "OPCA: Robust interdomain policy routing and traffic control," in *IEEE OPENARCH*, June 2002.

[27] B. Raghavan and A. Snoeren, "A system for authenticated policy-compliant routing," in *Proceedings of ACM SIGCOMM*, Aug 2004.

[28] National Science Foundation, "The GENI Initiative." www.nsf.gov/cise/geni/, 2005. Date accessed: 05/11/2006.

[29] K. Ishiguro, "GNU Zebra." http://www.zebra.org. Date accessed: 05/11/2006.

[30] X. Dimitropoulos, "Autonomous System Taxonomy Repository." http://www.ece.gatech.edu/research/labs/MANIACS/as_taxonomy/. Date accessed: 05/11/2006.

[31] T. Griffin and B. Premore, "An experimental analysis of BGP convergence time," in *Proceeding of the 9th International Conference on Network Protocols*, Nov. 2001.

[32] R. Vasudevan, M. Mao, O. Spatscheck, and K. V. der Merwe, "Reval: A tool for real-time evaluation of ddos mitigation strategies," in *Proceedings of Usenix Technical Conference*, May 2006.

[33] S. Seetharaman and M. Ammar, "On the Interaction between Dynamic Routing in the Native and Overlay Layers," in *IEEE INFOCOM*, April 2006.

[34] R. Gao, C. Dovrolis, and E. Zegura, "Avoiding oscillations due to intelligent route control systems," in *INFOCOM*, April 2006.

[35] S. McCanne and S. Floyd, "The LBNL network simulator." http://www.isi.edu/nsnam, 1997. Lawrence Berkeley Laboratory, Date accessed: 05/11/2006.

[36] G. F. Riley, R. M. Fujimoto, and M. H. Ammar, "Parallel/Distributed ns." http://www.cc.gatech.edu/computing/compass/pdns/index.html, 2000. Georgia Institute of Technology, Date accessed: 05/11/2006.

[37] R. M. Fujimoto, K. S. Perumalla, A. Park, M. A. H. Wu, and G. F. Riley, "Large-Scale Network Simulation. How Big? How Fast?," in *Proceedings of Eleventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'03)*, October 2003.

[38] X. Dimitropoulos, S. Beeram, and G. Riley, "BGP++ Home Page." http://www.ece.gatech.edu/research/labs/MANIACS/BGP++/. Date accessed: 05/11/2006.

[39] J. H. Cowie, D. M. Nicol, and A. T. Ogielski, "Modeling the global Internet," *Computing in Science and Engineering*, Jan. 1999.

[40] S. Tandel and C. de Launois, "C-BGP Home Page." http://cbgp.info.ucl.ac.be/. Date accessed: 05/11/2006.

[41] F. Hao and P. Koppol, "An Internet scale simulation setup for bgp," *Computer Communication Review*, vol. 33, no. 3, pp. 43–57, 2003.

[42] T. D. Feng, R. Ballantyne, and L. Trajkovi, "Implementation of BGP in a network simulator," in *Proc. of Advanced Simulation Technologies Conference 2004 (ASTC'04)*, April 2004.

[43] "Towards a BGP model in JavaSim." http://www.info.ucl.ac.be/~bqu/jsim/. Date accessed: 05/11/2006.

[44] B. K. Szymanski, Y. Liu, and R. Gupta, "Parallel network simulation under distributed Genesis," in *In Proceedings of ACM/IEEE/SCS of Workshop on Parallel and Distributed Simulation (PADS)*, June 2003.

[45] J.-Y. Tyan and C.-J. Hou, "JavaSim: A component-based compositional network simulation environment," in *Proceedings of the Western Simulation Multiconference, Communication Networks And Distributed Systems Modeling And Simulation*, Jan 2001.

[46] "Genesis Home Page." `http://www.genesis-sim.org/GENESIS/`. Date accessed: 05/11/2006.

[47] M. Pettersson, "Performace Counters." `http://sourceforge.net/projects/perfctr/`. Date accessed: 05/11/2006.

[48] J. Heidemann, K. Mills, and S. Kumar, "Expanding confidence in network simulation," Research Report 00-522, USC/Information Sciences Institute, April 2000. submitted for publication, IEEE Computer.

[49] D. Meyer, "University of Oregon Route Views Project," 2004.

[50] L. Gao, "On inferring autonomous system relationships in the Internet," in *Proc. IEEE Global Internet Symposium*, Nov. 2000.

[51] G. F. Riley, M. H. Ammar, and E. W. Zegura, "Efficient routing using nix-vectors," in *2001 IEEE Workshop on High Performance Switching and Routing*, May 2001.

[52] R. M. Fujimoto, K. Permualla, and I. Tacic, "Design of high performance RTI software," in *Distributed Simulation and Real-Time Applications*, Aug. 2000.

[53] D. M. Nicol, "Scalability, locality, partitioning and synchronization in PDES," in *Proceedings of the Parallel and Distributed Simulation Conference (PADS'98)*, 1998. Banff, Canada.

[54] K. Yocum, E. Eade, J. Degesys, D. Becker, J. Chase, and A. Vahdat, "Toward scaling network emulation using topology partitioning," in *Proceedings of Eleventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'03)*, October 2003.

[55] X. Liu and A. A. Chien, "Traffic-based load balance for scalable network emulation," in *in Proceedings of the ACM Conference on High Performance Computing and Networking*, November 2003.

[56] B. Hendrickson and R. Leland, "The Chaco user's guide," 1994.

[57] G. Karypis and V. Kumar, *MeTis: Unstrctured Graph Partitioning and Sparse Matrix Ordering System*.

[58] T. Tannenbaum and M. Litzkow, "Checkpointing and migration of UNIX processes in the Condor distributed processing system," *Dr Dobbs Journal*, February 1995.

[59] G. Riley, "GTNetS: A Packet-Level Network Simulator." http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS/. Date accessed: 05/11/2006.

[60] S. Beeram and G. Riley, "BGP++ GTNetS Integration." http://www.ece.gatech.edu/research/labs/MANIACS/BGP++/. Date accessed: 05/11/2006.

[61] C. Labovitz, A. Ahuja, R. Wattenhofer, and S. Venkatachary, "The impact of Internet policy and topology on delayed routing convergence," in *Proceedings of INFOCOM*, pp. 537–546, 2001.

[62] "RIPE." http://www.ripe.net. Date accessed: 05/11/2006.

[63] "CERNET BGP View." Date accessed: 05/11/2006.

[64] "A traceroute server list." http://www.traceroute.org, 2004.

[65] Y. Hyun, A. Broido, and kc claffy, "On third-party addresses in traceroute paths," in *Proceedings of 4th Passive and Active Measurement Workshop (PAM' 03)*, 2003.

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

[67] CAIDA, "Macroscopic Topology AS Adjacencies." http://sk-aslinks.caida.org. Date accessed: 05/11/2006.

[68] IRR, "Internet Routing Registries." http://www.irr.net/.

[69] B. Zhang, R. Liu, D. Massey, and L. Zhang, "Collecting the Internet AS-level topology," *ACM SIGCOMM Computer Communication Review (CCR)*, Jan 2005.

[70] X. Dimitropoulos, D. Krioukov, and G. Riley, "Revisiting Internet AS-level topology discovery," in *Proceedings of 6th Passive and Active Measurement Workshop (PAM' 05)*, March 2005.

[71] H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger, "Towards capturing representative AS-level Internet topologies," *Computer Networks Journal*, vol. 44, pp. 737–755, April 2004.

[72] D. Andersen, N. Feamster, S. Bauer, and H. Balakrishnan, "Topology Inference from BGP Routing Dynamics," in *Internet Measurement Workshop*, November 2002.

[73] O. Maennel and A. Feldmann, "Realistic BGP Traffic for Test Labs," in *Proceedings of ACM SIGCOMM*, August 2002.

[74] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica, "HLP: A next generation inter-domain routing protocol," in *Proceedings of ACM SIGCOMM*, 2005.

[75] X. Wang and D. Loguinov, "Wealth-based evolution model for the Internet AS-level topology," in *INFOCOM*, 2006.

[76] H. Chang, S. Jamin, and W. Willinger, "To peer or not to peer: Modeling the evolution of the Internet's AS-level topology," in *INFOCOM*, 2006.

[77] Z. M. Mao, L. Qiu, J. Wang, and Y. Zhang, "On AS-level path inference," in *SIGMETRICS*, 2005.

[78] H. Narayan, R. Govindan, and G. Varghese, "The impact of address allocation and routing on the structure and implementation of routing tables," in *Proceedings of ACM SIGCOMM*, 2003.

[79] X. Dimitropoulos, G. Riley, D. Krioukov, and R. Sundaram, "Towards a topology generator modeling AS relationships," in *ICNP (extended abstract)*, 2005.

[80] B. Krishnamurthy and J. Wang, "On network-aware clustering of web," in *Proceedings of ACM SIGCOMM*, 2000.

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

[82] G. D. Battista, M. Patrignani, and M. Pizzonia, "Computing the types of the relationships between Autonomous Systems," in *IEEE INFOCOM*, 2003.

[83] T. Erlebach, A. Hall, and T. Schank, "Classifying customer-provider relationships in the Internet," in *Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN)*, 2002.

[84] CAIDA, "Automated Autonomous System (AS) ranking." Research Project. `http://as-rank.caida.org`.

[85] M. Rimondini, "Statistics and comparisons about two solutions for computing the types of relationships between Autonomous Systems." `http://www.dia.uniroma3.it/~compunet/files/ToR-solutions-comparison.pdf`, 2002. Date accessed: 05/11/2006.

[86] Y. Collette and P. Siarry, *Multiobjective Optimization: Principles and Case Studies*. Berlin: Springer-Verlag, 2003.

[87] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman and Company, 1979.

[88] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM*, vol. 42, no. 6, pp. 1115–1145, 1995.

[89] H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger, "Towards capturing representative AS-level Internet topologies," *Computer Networks Journal*, vol. 44, pp. 737–755, April 2004.

[90] J. Xia and L. Gao, "On the evaluation of AS relationship inferences," in *IEEE GLOBECOM*, 2004.

[91] S. Busygin, "QUick ALmost EXact maximum weight clique/independent set solver." http://www.busygin.dp.ua/npc.html. Date accessed: 05/11/2006.

[92] A. S. S. GmbH, "L E D A library." http://www.algorithmic-solutions.com/enleda.htm, 2004. Date accessed: 05/11/2006.

[93] S. Benson, Y. Ye, and X. Zhang, "A dual-scaling algorithm for semidefinite programming." http://www-unix.mcs.anl.gov/~benson/dsdp/, 2004. Date accessed: 05/11/2006.

[94] I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani, "*k*-core decomposition: A tool for the analysis of large scale Internet graphs." http://arxiv.org/abs/cs.NI/0511007.

[95] R. Govindan and A. Reddy, "An analysis of Internet inter-domain topology and route stability," in *IEEE INFOCOM*, 1997.

[96] Z. Ge, D. Figueiredo, S. Jaiwal, , and L. Gao, "On the hierarchical structure of the logical Internet graph," in *SPIE ITCOM*, 2001.

[97] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M. Terpstra, *Routing Policy Specification Language (RPSL)*. IETF, RFC 2622, 1999.

[98] T. Bates, P. Smith, and G. Huston, "The CIDR report." http://bgp.potaroo.net/cidr/.

[99] T. Pedersen, "WordNet stop list." http://www.d.umn.edu/\~tpederse/Group01/wordnet.html.

[100] M. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, pp. 130–137, 1980.

[101] Y. Freund and R. E. Schapire, "A decision theoretic generalization of on-line learning and an application to boosting," in *Second European Conference on Computational Learning Theory (EuroCOLT-95)*, pp. pages 23–37, 1995.

[102] R. E. Schapire and Y. Singer, "BoosTexter: A boosting-based system for text categorization," *Machine Learning*, vol. 39, pp. 135–168, 2000.

[103] W. Aiello, F. Chung, and L. Lu, "A random graph model for massive graphs," in *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 171–180, ACM Press, 2000.

[104] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology," in *Proceedings of ACM SIGCOMM*, 1999.

[105] A. Fabrikant, E. Koutsoupias, and C. H. Papadimitriou, "Heuristically optimized trade-offs: A new paradigm for power laws in the Internet," in *Proceeding of the 29th International Colloquium on Automata, Languages, and Programming (ICALP)*, vol. 2380 of *Lecture Notes in Computer Science*, pp. 110–122, Springer, 2002.

[106] D. Maltz, G. Xie, J. Zhan, H. Zhang, G. Hjalmtysson, and A. Greenberg, "Routing design in operational networks:a look from the inside," in *Proceedings of ACM SIGCOMM*, Aug. 2004.

[107] H. Tangmunarunkit, R. Govindan, S. Shenker, and D. Estrin, "The impact of routing policy on Internet paths," in *IEEE INFOCOM*, 2001.

[108] H. Tangmunarunkit, R. Govindan, and S. Shenker, "Internet path inflation due to policy routing," in *SPIE ITCom*, 2001.

[109] L. Gao and F. Wang, "The extent of AS path inflation by routing policies," in *IEEE Global Internet Symposium*, 2002.

[110] N. Spring, R. Mahajan, and T. Anderson, "Quantifying the causes of path inflation," in *Proceedings of ACM SIGCOMM*, 2003.

[111] P. Mahadevan, D. Krioukov, M. Fomenkov, B. Huffaker, X. Dimitropoulos, kc claffy, and A. Vahdat, "Lesson from three views of the Internet topology," Technical Report TR-2005-02, CAIDA, 2005. Date accessed: 05/11/2006.

[112] B. M. Waxman, "Routing of multipoint connections," *IEEE JSAC*, December 1988.

[113] E. W. Zegura, K. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *IEEE INFOCOM*, 1996.

[114] M. Doar, "A better model for generating test networks," in *IEEE GLOBECOM*, 1996.

[115] R. Albert and A.-L. Barabási, "Topology of evolving networks: Local events and universality," *Physical Review Letters*, vol. 85, no. 24, pp. 5234–5237, 2000.

[116] H. Chang, S. Jamin, and W. Willinger, "Internet connectivity at the AS level: An optimization driven modeling approach," in *Proceedings of MoMeTools*, 2003.

[117] J. M. Carlson and J. Doyle, "Highly optimized tolerance: A mechanism for power-laws in designed systems," *Physical Review E*, 1999.

[118] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: An approach to universal topology generation," in *MASCOTS*, 2001.

[119] J. Winick and S. Jamin, "Inet-3.0: Internet topology generator," Technical Report UM-CSE-TR-456-02, University of Michigan, 2002.

[120] C. Gkantsidis, M. Mihail, and E. Zegura, "The markov chain simulation method for generating connected power law random graphs," in *SIAM Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2003.

[121] R. B. Nelson, "An introduction to copulas," *Springer-Verlag Lecture Notes in Statistics*, vol. 139, p. 216, 1999.

[122] A. Sklar, "Fonctions de repartition a n dimensions et leurs marges," 1959.

[123] C. Gu, *Smoothing Spline ANOVA Models*. New York: Springer-Verlag, 2002.

[124] C. Gu, "gss: Generalized Smoothing Splines." `http://cran.r-project.org/src/contrib/Descriptions/gss.html`. Date accessed: 05/11/2006.

[125] P. Mahadevan, D. Krioukov, K. Fall, and A. Vahdat, "A Basis for Systematic Analysis of Network Topologies," in *Proceedings of ACM SIGCOMM*, September 2006.

[126] X. Dimitropoulos and G. Riley, "Creating realistic BGP models," in *Proceedings of Eleventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'03)*, October 2003.

[127] X. Dimitropoulos and G. Riley, "Large-scale simulation models of BGP," in *Proceedings of Eleventh International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'04)*, October 2004.

[128] X. Dimitropoulos and G. Riley, "Efficient large-scale BGP simulations," *To appear on Elsevier Computer Networks, Special Issue on Network Modeling and Simulation*, 2006.

[129] X. Dimitropoulos, D. Krioukov, G. Riley, and K. Claffy, "Classifying the types of autonomous systems in the Internet," in *ACM SIGCOMM (extended abstract)*, 2005.

[130] X. Dimitropoulos, D. Krioukov, G. Riley, and K. Claffy, "Revealing the autonomous system taxonomy: The machine learning approach," in *Proceedings of 7th Passive and Active Measurement Workshop (PAM' 06)*, March 2006.

[131] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, K. Claffy, and G. Riley, "AS relationships: Inference and validation." `http://arxiv.org/abs/cs/0604017`. Date accessed: 05/11/2006.

[132] X. Dimitropoulos and G. Riley, "Modeling autonomous system relationships," in *Proceedings of Principles of Advanced and Distributed Simulation (PADS'06)*, May 2006. Singapore.

[133] R. Cole, L. Benmohamed, A. DeSimone, and B. Doshi, "BORDER GATEWAY PROTOCOL 4 (BGP4) PERFORMANCE OVER INTERMITTENT SATELLITE LINKS," in *IEEE PacRim Confernce on Communications, Computers and Signal Processing*, Aug 2005.

[134] B. Zhang and R. Liu, "Internet Topology Page." `http://irl.cs.ucla.edu/topology/`. Date accessed: 05/11/2006.