# Implications of Traffic Characteristics on Interdomain Traffic Engineering

**Thèse**

soumise par **Steve Uhlig**

en vue de l'obtention du grade de

Docteur en Sciences Appliquées

au Département d'Ingénierie Informatique

de l' Université catholique de Louvain

**Membres du Jury :**

Prof. Jean-Didier **Legat** (Président, ELEC/FSA/UCL)

Dr. Christophe **Diot** (Intel Research Cambridge)

Dr. James **Roberts** (France Télécom R&D)

Prof. Olivier **Bonaventure** (INGI/FSA/UCL, promoteur)

Prof. Bernard **Fortz** (POMS/IAG/UCL)

Prof. Marc **Lobelle** (INGI/FSA/UCL)

## Université catholique de Louvain

March, 2004

# Contents

# List of Tables

# List of Figures

# Implications of Traffic Characteristics on Interdomain Traffic Engineering

Steve Uhlig, PhD
Université catholique de Louvain, 2004

Supervisor: Prof. Olivier Bonaventure

## Abstract

The Internet routing system today is divided into two views: intradomain and interdomain. The interdomain Internet is made of autonomous systems (AS). Each autonomous system uses an exterior gateway protocol (EGP) to exchange reachability information between ASes. Autonomous systems are made of routers that constitute the intradomain view of each AS. Routers in a given AS exchange intradomain routing information through an interior gateway protocol (IGP) that distributes the whole map of the intradomain network to all routers of the AS.

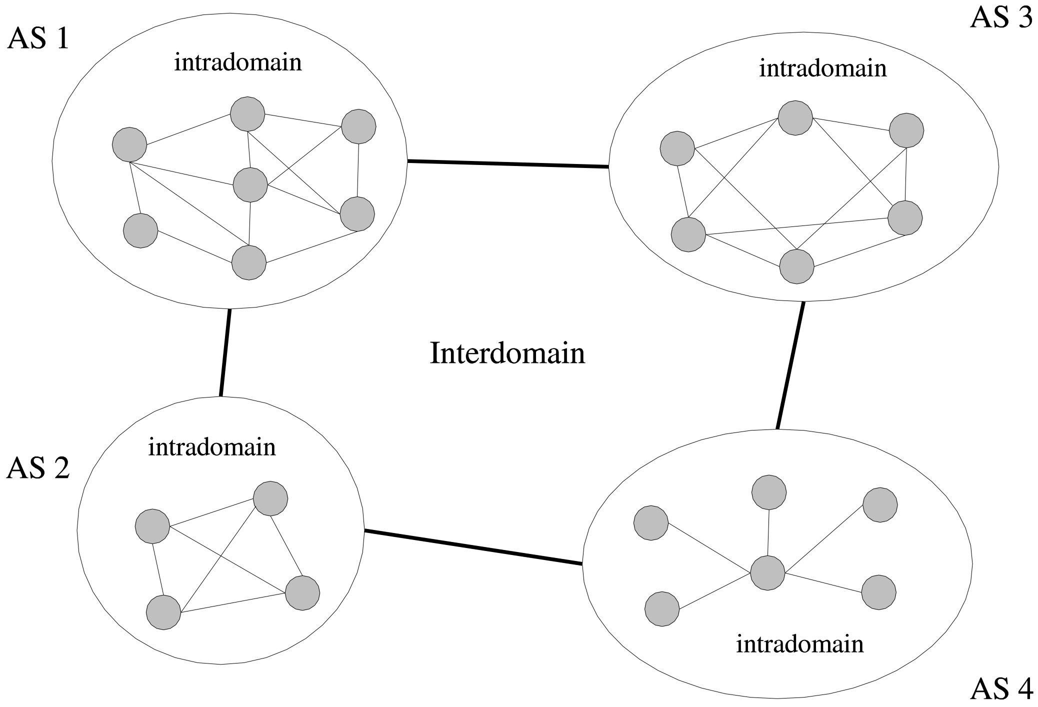



Figure 1: Intradomain and interdomain views of the Internet

Figure 1 illustrates this partitioning of the Internet by showing four ASes, each having a particular intradomain topology with routers running at least an IGP protocol that distributes the whole map of the intradomain topology to all other routers of the AS. Each AS has a different intradomain topology that is not seen by the interdomain view. The interdomain view only sees the four ASes and their interconnections.

At the intradomain level, the topology is known and the control of the routing information can be assumed. Engineering the flow of the traffic within an organization by tweaking the intradomain routing protocols is thus possible [77, 76]. At the interdomain level on the other hand, the approach used today in the Internet is the one of a distributed control of the routing information. The current interdomain routing protocol, BGP [156], knows only a limited part of the topology of the Internet. The global Internet is divided into Autonomous Systems (AS), each running at least an intradomain routing protocol (OSPF or IS-IS) within its boundaries and using BGP to exchange routing information with other ASes.

The objective of BGP is to distribute information about the reachability of IP prefixes (i.e. reachable networks) to other ASes. With BGP, an AS advertises to each neighbor AS all the networks it can reach. Among the IP prefixes that an AS advertises, some are internal prefixes that are reachable within this AS (internal to this AS) and others are prefixes that have been learned through its BGP neighbors. A key feature of BGP is that it allows each network operator to define its routing policies. Those policies are implemented by using filters [87]. A BGP filter is a rule applied upon receiving a BGP route from a BGP peer or before sending a BGP route to a peer AS. BGP filters can prevent some routes from being accepted from or announced to peer ASes, and can also modify the attributes of the BGP routes on a per-AS basis so that some routes be preferred over others.

The graph of the interdomain Internet can be thought as more or less hierarchical [157]. The core of the Internet is made of about twenty large ASes today, called tier-1's. These ASes provide global connectivity in the Internet and are densely interconnected. The main purpose of tier-1's is to provide transit service. Tier-1's carry IP packets that enter their network and whose destination is outside their network. Tier-1's constitute the first level of the hierarchy. Then, we have smaller or less connected transit service providers that make the second level of the hierarchy. These smaller transit ASes are sometimes called tier-2, tier-3, and tier-4 to indicate their less worldwide connectivity. Note that distinguishing between smaller transit ASes is particularly difficult because of the lack of clear differences between them. Finally, we have stub ASes at the edge of the interdomain topology that do not provide transit service but that only generate and receive IP packets without allowing IP packets to otherwise transit through their network. It can host content or provide access to dialup or broadband users. This tier-structure of the Internet is actually quite loose, stub ASes can be connected to any type of transit provider, and transit providers also interconnect between one another without strict consideration of the "type" of the AS. The interconnections between ASes arise for economical as well as geographical reasons so this partitioning of the ASes into the different types as presented in [157] cannot be considered as a definitive classification of ASes. The actual structure of the interdomain Internet is more complex, we only provide a sketchy picture here. More details can be found in [157]. For information, we provide on Figure 2 an illustration of the hierarchical structure of the Internet. We show on Figure 2 the classification of the ASes according to the last BGP dumps used on the website of [157] at the date of the writing of this thesis. The BGP tables date from January 9 2003. The whole topology is made of 14,695 ASes and 30,815 unique undirected edges. The size of each oval on Figure 2 aims at emphasizing the relative importance of each type of AS. We used dashed lines between tier-2, tier-3 and tier-4 on Figure 2 to insist on the loose classification of these three AS types.

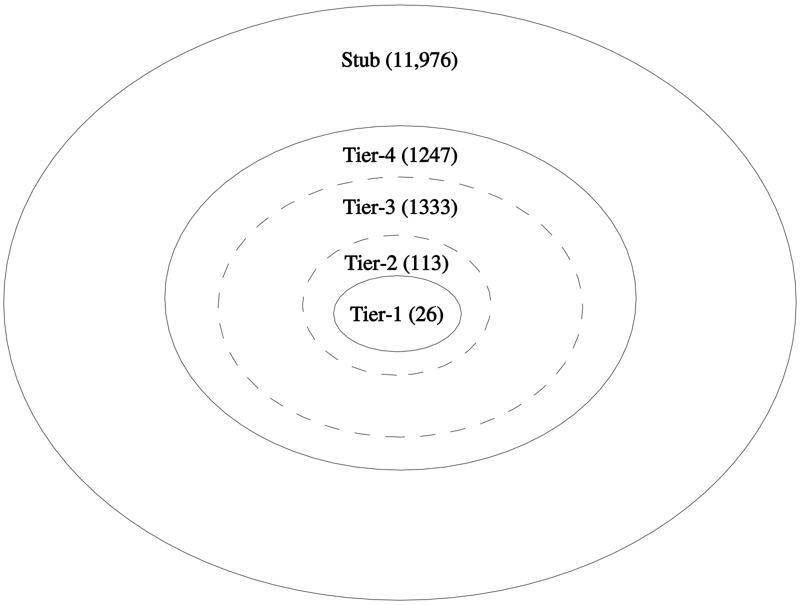



Figure 2: Hierarchical structure of the Internet

The main purpose of this thesis is to evaluate interdomain traffic engineering with BGP, based on the behavior of real traffic. The term "traffic engineering" is defined in RFC 3272 [16] as
*Internet traffic engineering is defined as that aspect of Internet network engineering dealing with the issue of performance evaluation and performance optimization of operational IP networks. Traffic Engineering encompasses the application of technology and scientific principles to the measurement, characterization, modeling, and control of Internet traffic.*

Traffic engineering thus consists of all the available techniques whose purpose is to directly or indirectly allow to modify the behavior of the traffic to achieve certain objectives. Traffic engineering has received a lot of attention during the last few years [16, 184]. Initially, traffic engineering was considered as a solution to allow large tier-1 service providers to optimize the utilization of their network. In these large networks, there are typically several possible paths to reach a given destination or border router. Ideally, to achieve a good network utilization, the traffic should be spread evenly among all the available links. Unfortunately, this does not correspond to the way traditional IP routing protocols behave.

In most cases, the IP routing protocol is not aware of the load on the various parts of the network and selects for each destination the shortest path based on static metrics such as the hop count or the delay. This destination based routing creates an uneven distribution of the traffic that may lead to periods of congestion inside the

Internet service provider (ISP) backbone. Several techniques have been proposed to better spread the load throughout the entire network [16]. A first solution is to select appropriate link metrics based on a known traffic matrix [77, 183]. This solution can provide some interesting results if the traffic matrix is known and stable. A second solution is to rely on a connection-oriented layer-2 technology [18] such as ATM, MPLS or one of the emerging optical technologies. In this case, layer-2 connections can be established statically [58] or dynamically between distant routers and the layout of these connections can be optimized to achieve an even distribution of the traffic inside the network [16]. It is also possible to dynamically create new layer-2 connections in order to quickly respond to link failures or changes in the traffic pattern [16].

At the opposite of large tier-1 providers, small providers and multi-homed corporate networks have different traffic engineering requirements. Their networks have usually a simple topology and are frequently over-provisioned. The traffic engineering solutions mentioned above are not really useful in such networks. For these networks, the costly resource that needs to be optimized with traffic engineering is their interdomain connectivity, i.e. the links that connect them to the rest of the Internet.

Most of the traffic engineering literature has focused on intradomain traffic engineering and large tier-1 providers. This thesis on the other hand aims to improve our understanding of the issues related to interdomain traffic engineering for stub ASes. Throughout this thesis, our approach is to rely on real network traffic traces to evaluate various aspects related closely or remotely to interdomain traffic engineering. The only part of the thesis where we do not rely on traffic traces is Part I that constitutes an absolute prerequisite for all other parts of this thesis.

Now let us explain the logic behind the structure of this thesis. Traffic engineering is about engineering the traffic. What is being done is thus *engineering* while *on what* it is being done is the *traffic*. To understand interdomain traffic engineering, the *traffic* part is as important as the *engineering* one.

In Part I, we provide the necessary information related to BGP and its working for controlling the interdomain traffic. Reading thoroughly Part I is necessary to understand the complexity and issues of interdomain traffic engineering. This first part is largely protocol-centered, focusing on the *engineering* part of interdomain traffic engineering. In Part II, we focus on the *traffic* part of traffic engineering by closely looking at the properties of the interdomain traffic. Part III tackles the optimization of traffic objectives by tweaking BGP. Viewing interdomain traffic engineering as an optimization problem will show us the limitations of pushing the tweaking of BGP towards optimality. Finally, Part IV evaluates interdomain traffic engineering and discusses its future.

Throughout this thesis, our viewpoint will be largely biased towards the *traffic* part of traffic engineering. Our feeling when starting this research (end of year 2000) was that while traffic engineering techniques were being designed, not much was known about the behavior of the interdomain traffic except its scaling properties (see Chapter 1). It is [172] where we evaluated the feasibility of using MPLS for carrying interdomain traffic that lead us to question the feasibility of interdomain traffic engineering for stub ASes.

Today, more and more ISPs are concerned with how to control the traffic at multiple interdomain access points. Recent studies [157, 9] have shown that a large number of today's ASes are multi-homed. More than 60 % of all ASes have at least two BGP peers [157]. An increasing trend towards multi-homing was shown in [9] and is likely to continue as ISPs become aware of interdomain traffic engineering tools and techniques [137, 22, 12]. Examples of traffic engineering tools include the solutions commonly referred as "route optimization" techniques [12]. These route optimization techniques work on relatively short timescales and target multi-homed ASes. Their principle is to find, based on active and passive measurements, the "best" route to attain a destination or/and to be attained by external hosts. These techniques work on timescales in the order of a round-trip time or more and adapt the traffic flow to the current conditions of the network, i.e. they try to find the best upstream provider to send or receive traffic. Their objective is often to optimize the "instantaneous" quality of service (QoS) experienced by the traffic, by measuring the "quality" of the routes available from the upstream providers and trying to choose the best upstream in real time. To the best of our knowledge, no paper in the literature has evaluated nor described how these techniques exactly achieve their goals. We are only aware of [176] that proposed a method to engineer the outbound interdomain traffic for stub-ASes by tweaking the `local-pref` attribute of BGP routes and [67] that discussed the predictability of the BGP routes for outbound traffic engineering.

The current state of the art of interdomain traffic engineering is of manually changing the configuration of the routers to influence the traffic on a trial and error basis [68]. Interdomain traffic engineering is still in its infancy. Understanding the dynamics of the interdomain traffic and the relationships between the topology of the traffic and BGP peering relationships is absolutely necessary to do it properly [14]. The main contribution of this thesis is to partially reduce the gap between the *traffic* part and the *engineering* part of traffic engineering. In addition, we show in Part III that designing interdomain traffic engineering is possible with a proper understanding of the traffic characteristics and interdomain routing.

Although we largely delay the discussion on the rationale of interdomain traffic engineering to Part IV, we nevertheless mention here very shortly a few aspects of the design of the Internet before getting into the content of this thesis. The design objectives during the early years of the Internet (called the ARPANET at the time) had in mind interoperability and robustness. Roughly speaking, the issue of the ARPANET was to allow end hosts to communicate across the network even in the presence of router failures. Today the Internet covers a large fraction of the planet, with millions of hosts connected to the network. While the ARPANET was a communication system, the Internet today is rather a virtual world over which agents interact. The ARPANET was a communication infrastructure to be properly engineered. The Internet nowadays is much more than just another medium to exchange data. Many businesses live from the Internet. Agents that pursue different potentially conflicting objectives share this new virtual world.

# Road map

In Part I, we provide a detailed presentation of the working of the BGP protocol and the interactions between BGP and the control of the interdomain traffic.

In Part II, we study the characteristics of the interdomain traffic from a stub AS viewpoint. We first try to understand in Chapter 1 the dynamics of the interdomain traffic as seen at the access point of several stubs. Chapter 2 then studies the topological distribution of the traffic. The last chapter of this first part is Chapter 3 where we study the dynamics of the traffic on the interdomain topology.

In Part III, we evaluate the feasibility of interdomain traffic engineering by tweaking the BGP routing protocol. Chapter 4 first studies the problem of tweaking BGP to optimize a traffic objective defined on a daily timescale. Chapter 5 then studies the problem of trying to minimize the number of BGP filters to minimize a traffic objective defined on a daily timescale. Chapter 6 studies the optimization of a traffic objective on the relatively short timescales of minutes. The last chapter of Part III, Chapter 7, finally deals with the problem of multiple-objectives traffic engineering with BGP on a daily timescale.

In Part IV, we evaluate the current state of interdomain traffic engineering, envision its future and provide a few guidelines for further work on the interdomain Internet.

Sections that we consider more difficult or technical are marked with an asterisk ($*$), these might be skipped on a first reading without impacting too much on the understanding of the rest of the text. Note however that some important technical terms might be defined in these sections so that checking them whenever an undefined term is encountered is recommended.

Figure 3 illustrates the structure of the thesis and the interdependencies among its four parts. Part I is a prerequisite for all other three parts. Even readers that might think to understand the working of BGP should read Part I, because understanding the BGP protocol is not sufficient to fully understand the complexity of interdomain traffic engineering with BGP. Parts II, III and IV are relatively independent and any of them can be skipped without impeding on the ability to understand others parts.

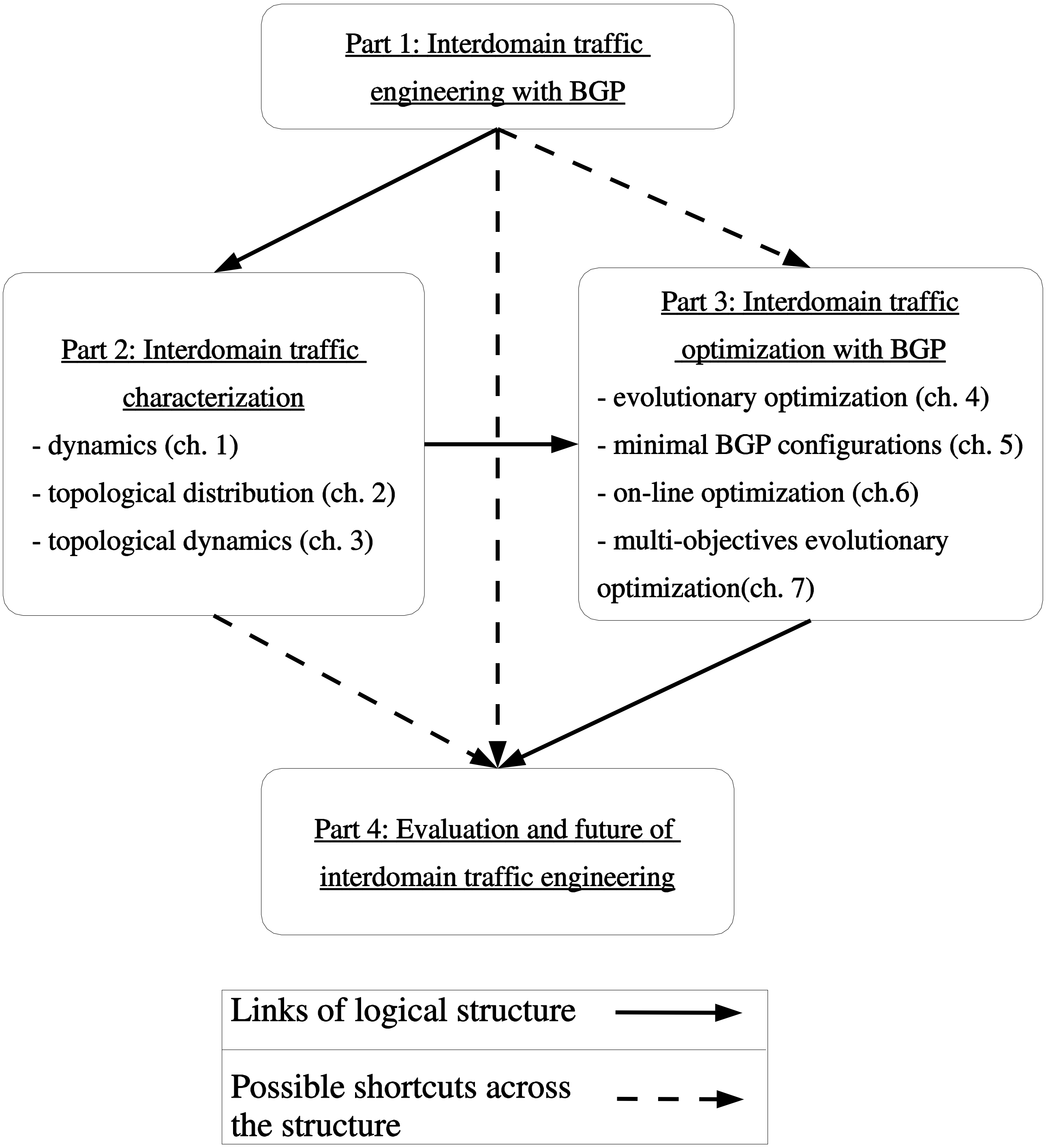



Figure 3: Interdependencies among the different parts of the thesis

# Implications of Traffic Characteristics on Interdomain Traffic Engineering

## Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Steve Uhlig
March 1, 2004

# Acknowledgments

Acknowledging every person that has had a direct or indirect impact on the realization of this thesis is not possible. I have benefited from too many persons to ever being able to mention all of them in such a limited space as this acknowledgement section. I shall thus try to keep short and praise only the first few whom I recall by the time of this writing.

Let me first thank the external members of my thesis committee, Christophe Diot and James Roberts for accepting to be in my thesis committee, for reading this thesis and providing valuable comments on the content. Let me then also acknowledge the comments of the other members of the thesis committee, Bernard Fortz, Marc Lobelle and Jean-Didier Legat.

The very first person responsible for the content of this thesis is my friend and coincidentally thesis advisor, Olivier Bonaventure. I know Olivier since 1999. I did my Ms. degree thesis dissertation under his guidance and worked as a researcher and teaching assistant with him since then. The impact of his experience in networking forged my perception of what research should be. Our respective viewpoints have sometimes been conflicting, with the fortunate consequence of strengthening my personality as a researcher. Olivier never forced me to adopt his opinion, this would have been futile anyway. Rather he obliged me to justify my viewpoint even if it had to go against his. If someone is to be blamed for the ideas contained in this thesis, Olivier is probably the one you should direct your complaints at. He is the person who allowed me to come up with the positions I hold here. He had enough folly to think that something interesting could possibly emerge from my very person. I think he was wrong to think that, but this thesis seems to be the proof of the contrary though. I will never be able to repay this debt, mainly for his time and confidence, as well as the financial support. So for all criticisms concerning the content of this thesis feel free to contact him for complaints. He shall be happy to agree with you.

The second person that provided indirect but important support for this research is Stefaan De Cnodder from Alcatel Antwerp (although he does not seem to be willing to recognize it). Stefaan read almost every paper I wrote during the course of this thesis and provided invaluable comments that changed my feelings about various subjects. Having such a high quality reviewer is something I wish to every researcher.

Chris Rapier from the Pittsburgh Supercomputing Center has been a wonderful paper co-author to work and discuss with. Unfortunately, our contacts have almost exclusively occured through e-mail, except a short meeting at ACM SIGCOMM 2002 in Pittsburgh. Chris generously provided several traffic traces captured at PSC that helped make possible several papers [177, 137, 168, 178] as well as Chapters 1 and 3 of this thesis. I wish this collaboration be as fruitful in the future.

Bruno Quoitin from FUNDP (now UCL) played several key roles. First, he co-authored several important papers [136, 135, 137, 176]. Second, Bruno provided simulations in Part IV of this thesis. Third, Bruno proof-read some difficult papers dealing with the scaling properties of the traffic [177, 168, 170]. Finally, our many discussions contributed to reinforce my opinion on several subjects discussed in Part IV.

Sébastien Tandel co-authored [29]. Additionally, he pushed me to provide the lengthy discussion on the tie-breaking rules of the BGP decision process in Chapter I.

The various researchers in the INFONET group with whom I had the chance to share space and time at the Institut d'Informatique of the Facultés Universtitaires Notre-Dame de la Paix in Namur all deserve a special mention because it is during these two years that the building of my research has been the most critical. Thanks to Vincent Letocart, Stéphane Nicoll, Cristel Pelsser, Bruno Quoitin, Pierre Reinbold, Louis Swinnen and Sébastien Tandel.

Many members of the staff of the Computer Science and Engineering department at the Université Catholique de Louvain contributed to the nice atmosphere that reigns there. Thanks to them too.

Strangely enough, I have rarely had the opportunity to acknowledge all the paper and book authors whose ideas I have actually stolen consciously or not without even having the politeness to gratefully thank them by at least citing their work in the references. To put it in evolutionary algorithms terminology, I am convinced that this research is nothing more than a genetic accident, a simple recombination and sampling of previous ideas, added with many mutations. If by chance some unexpected originality was to occur in this thesis, it is most likely to be accidental and rooted in misunderstandings from my part, rather than constitute intentional contributions. I apologize for having corrupted interesting works and ideas of others, leading to the present document. All complaints in this respect should be directed to me. If some originality was however to occur in this work, then I am declining any responsibility for this.

Before closing this aknowledgement part, I vaguely recall about two persons that also share some responsibility for all this. Obviously, my parents Małgorzata Migalska and Pierre Uhlig can be considered as the main culprits (in a biological sense at least) for the current work. Without them, you would be doing something valuable instead of reading this document. Their support, love and patience allowed this work to find its way up to you.

Before concluding this acknowledgement section, let the reader pardon me for introducing him to one of my preferred citations. The following text comes from a famous Buddhist poem, the Bodhicaryavatara, written around the ninth century A.D. by the Buddhist monk Santideva. While vainly trying to pursue a long Buddhist tradition of humility, may this citation represent the humor of the writer when proposing his sloppy viewpoint before the careful consideration of the reader. May it incense your mind during the journey throughout this thesis, as it incenses mine during my scientific journey...

*"Nothing new will be said here, nor have I any skill in composition. Therefore I do not imagine that I can benefit others. I have done this only to perfume my own mind".*

Santideva, The Bodhicaryavatara.

All remaining omissions, inaccuracies, errors, are the entire responsibility of the author. May the reader be kind enough to contact him to provide any correction that may reduce the inability of the present document to convey its intended meaning.

Steve Uhlig

*Université catholique de Louvain*
*March 2004*

# Part I

# Interdomain traffic engineering with BGP

## 0.1 Introduction

This first part presents the working of the interdomain routing protocol, BGP, and discusses the various ways in which interdomain traffic engineering is possible by tweaking BGP. A large fraction of this part has appeared in [137, 29].

This chapter is structured as follows. In section 0.2 we explain the working of the BGP routing in the Internet. Section 0.3 then introduces the main feature of BGP, route filtering. Section 0.4 explains the working of the BGP decision process. Finally, section 0.5 delves into how to perform interdomain traffic engineering with BGP and discusses its limitations.

## 0.2 BGP routing in the Internet

Internet routing is handled by two distinct protocols with different objectives. Inside a single domain, link-state intradomain routing protocols distribute the entire network topology to all routers and select the shortest path according to a metric chosen by the network administrator. Across interdomain boundaries, the interdomain routing protocol is used to distribute reachability information and to select the best route to each destination according to the policies specified by each domain administrator. For scalability reasons, the interdomain routing protocol is only aware of the interconnections between distinct domains, it does not know any information about the content of each domain. A result of this partition is that a router typically has a detailed knowledge of the topology within its domain and only topological information about other domains.

The Border Gateway Protocol (BGP) [138, 156] is the current de facto standard interdomain routing protocol. In BGP terminology, a domain is often equivalent to an Autonomous System (AS). BGP is a *path-vector protocol* that works by sending *route advertisements*. A route advertisement indicates the reachability of a network, namely a network address and a netmask representing a block of contiguous IP addresses. For instance, `192.168.0.0/24` represents a block of 256 addresses between `192.168.0.0` and `192.168.0.255`. Besides the reachable network and the IP address of the router that must be used to reach this network (known as the `next-hop`), a route advertisement also contains the `AS-path` attribute which contains the list of all the transit ASes that must be used to reach the announced network. A route advertisement may also contain several attributes such as the `local-pref`, Multi-Exit Discriminator (`MED`) or `communities` attributes [138, 156]. An important point about BGP is that if a BGP router of `ASx` sends a route announcement for network $N$ to a neighbor BGP router of `ASy`, this implies that `ASx` accepts to forward IP packets to destination $N$ on behalf of `ASy`.

There are two variants of BGP [138, 156]. The eBGP variant is used to announce the reachable prefixes on a link between routers that are part of distinct ASes. The iBGP variant on the other hand is used to distribute, inside an AS, the best routes learned from neighboring ASes. To distribute the best BGP routes inside an AS, each BGP router of that AS will establish an iBGP session with all the other BGP routers of the AS. A full mesh of iBGP sessions is typically created inside the AS[1].

The top part of Figure 4 presents a simple Internet topology composed of six ASes. Each AS of the simple Internet topology of Figure 4 contains routers named $Rxy$ where $x$ identifies the AS number to which the router belongs and $y$ identifies the router in AS$x$. The bottom part of Figure 4 provides a possible setting of the eBGP and iBGP sessions that would exist in the simple Internet topology of the top part of Figure 4, under the assumption that a full-mesh of iBGP sessions are established between the edge routers in each domain. All routers of the simple Internet topology (top part of Figure 4) dot not need to run BGP. $R_{12}$ for instance is used by the edge routers of AS1 as a transit router for the traffic between the edge routers. The BGP routers of AS1 are not aware of the existence of router $R_{12}$ with BGP since no BGP session is established with $R_{12}$, they might know router $R_{12}$ only through IGP routing.

## 0.3 Route filtering

Inside a single domain, all routers are considered as "equal" and the intradomain routing protocol announces all known paths to all routers. In contrast, in the global Internet, all ASes are not equal and an AS will rarely agree to provide a transit service for all its connected ASes toward all destinations. Therefore, BGP allows a router to be selective in the route advertisements that it sends to neighbor eBGP routers. This route filtering process is the basis of what is called "policy routing" in the Internet. To better understand the operation of BGP, it is useful to consider a simplified view of a BGP router as shown in Figure 5.

A BGP router processes and generates route advertisements as follows. First, the administrator specifies, for each BGP peer, an input filter (left of Figure 5) that is used to select the acceptable advertisements. For example, a BGP router could only select the advertisements with an `AS-Path` containing a set of trusted ASes. Once a route advertisement has been accepted by the input filter, it is placed in the BGP routing table, possibly after having updated some of its attributes. The BGP routing table thus contains all the acceptable routes received from the BGP neighbors.

Second, on the basis of the BGP routing table, the BGP decision process (center of Figure 5) will select the best route toward each known network. Based on the `next-hop` of this best route and on the intradomain routing table, the router will install a route toward this network inside its forwarding table. This table is then looked up for each received packet and indicates the outgoing interface that must be used to reach the packets' destination.

Third, the BGP router will use its output filters (right of Figure 5) to select among the best routes in the BGP routing table the routes that will be advertised to each

[1] Large ASes rely on route reflectors or confederations [23, 166] to reduce the number of iBGP sessions inside their domain.

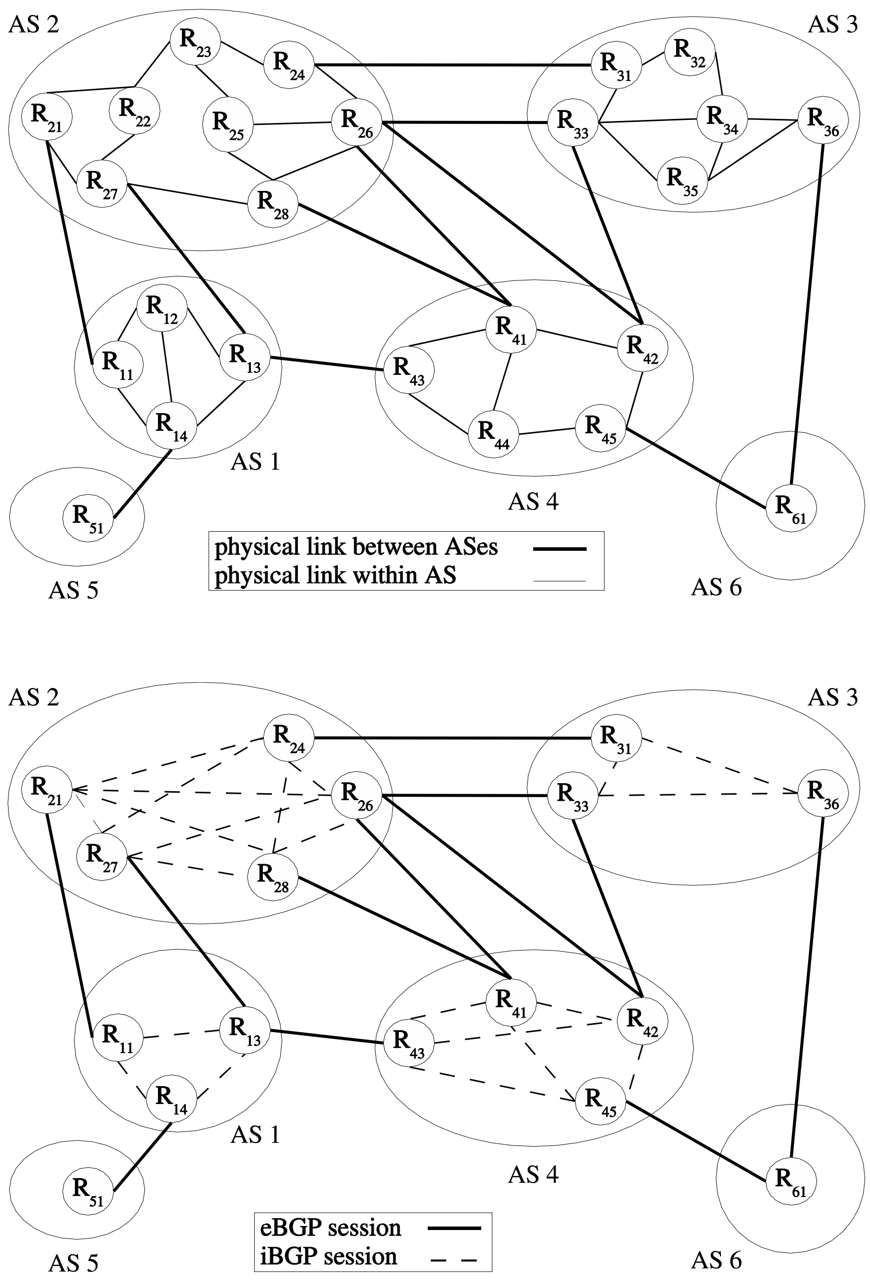



Figure 4: Simple Internet topology (top) and corresponding eBGP and iBGP sessions (bottom)

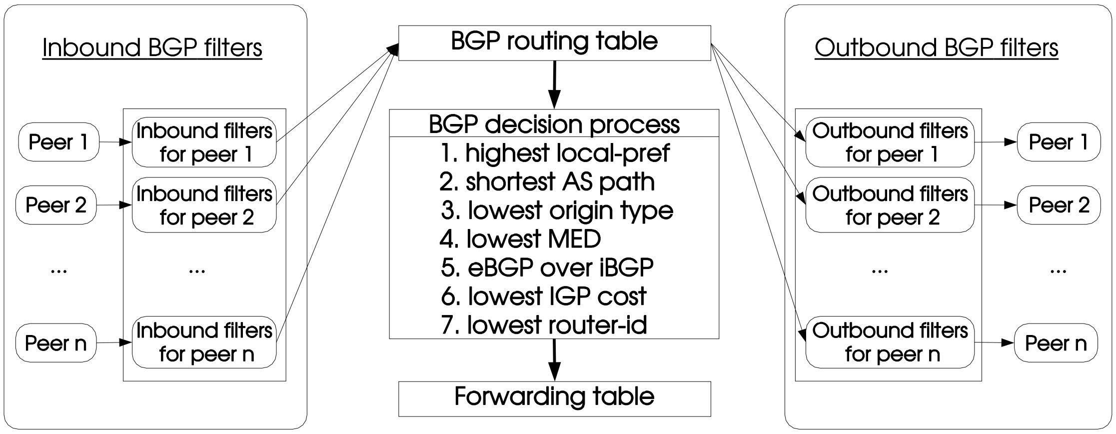



Figure 5: Simplified operation of a BGP router.

BGP peer. At most one route will be advertised for each reachable destination. The BGP router will assemble and send the corresponding route advertisement messages after a possible update of some of their attributes.

The input and output filters used in combination with the BGP decision process are the key mechanisms that allow a network administrator to support within BGP the business relationships between two ASes. Many types of business relationships can be supported by BGP. Two of the most common relationships are the customer-to-provider and the peer-to-peer relationships [157, 79]. To understand how these two relationships are supported by BGP, consider the top part of Figure 4. If `AS5` is `AS1`'s customer, then `AS5` will configure its BGP router to announce its routes to `AS1`. `AS1` will accept these routes and announce them to its peer (`AS4`) and upstream provider (`AS2`). `AS1` will also announce to `AS5` all the routes it receives from `AS2` and `AS4`. If `AS1` and `AS4` have a peer-to-peer relationship on the link between $R_{13}$ and $R_{43}$, then router $R_{13}$ will only announce on this link the internal routes of `AS1` and the routes received from `AS1`'s customer (i.e. `AS5`). The routes received from `AS2` will be filtered and thus not announced on the $R_{13} - R_{43}$ link by router $R_{13}$. Due to this filtering, `AS1` will not carry traffic from `AS4` toward `AS2`.

## 0.4 The BGP decision process

A BGP router receives one route toward each destination from each of its peers. To select the best route among this set of routes, a BGP router relies on a set of criteria called the decision process. Most BGP routers apply a decision process similar in principle to the one shown in Figure 5 [138]. The set of routes with the same destination are analyzed by the criteria in the sequence indicated in Figure 5. These criteria act as filters and the $N^{th}$ criterion is only evaluated if more than one route has passed the $N-1^{th}$ criterion. Most BGP implementations allow the network administrator to optionally disable some of the criteria of the BGP decision process.

The first rule of the BGP decision process uses the `local-pref` attribute. The `local-pref` attribute is an administrative cost specifying the preference among the different routes towards a given destination. The route with the highest value of the `local-pref` attribute is considered as the best one. This attribute can be considered as the one that allows to enforce the different types of peering relationships by indicating which route is to be preferred over another. The `local-pref` attribute is local to the AS receiving the route. It is set upon receival of the BGP route and not advertized outside the local domain.

The second rule of the BGP decision process uses the length of the `AS path` of the route. This rule constitutes a distance metric for the interdomain path followed by packets whose destination IP address matches the prefix of the route. The best route is the one having the shortest `AS path` length. The information contained in the `AS path` attribute concerns only the traffic that goes from the local AS towards the prefix of the route. The actual path used in the reverse direction ought not to be the same due to local policies enforced along the path between the local AS and the destination prefix [159, 80].

The third rule concerns the `origin type` of the route. This attribute identifies how the originating AS learned about the route, where IGP is preferred to EGP (a now defunct distance-vector protocol) which is preferred to INCOMPLETE. This attribute is becoming obsolete although it is still currently used in the BGP decision process [46].

The fourth rule is related to the `MULTI_EXIT_DISC` (for multi exit discriminator) attribute. Comparing routes with respect to the value of the MED attribute requires that these routes be learned from the same neighboring AS. Routes which do not have the MED attribute are considered to have the lowest possible MED value. The route with the lowest MED value is preferred. The MED attribute serves at selecting a particular egress point in the local domain. The input filter might override the `origin type` and the value of the MED attribute so that the use of these attributes for route selection usually depend on a mutual understanding between the two neighboring ASes.

The fifth rule then prefers the routes that have been learned through eBGP over those learned from iBGP.

The sixth rule prefers the route with the lowest IGP path cost to the next hop. This rule allows for what is commonly termed “hot-potato” routing, which means that the local AS tries to get rid of the IP packet as soon as possible. The logic behind “hot-potato” routing is to minimize the resource consumption implied by the forwarding of the IP packet that transits through the local domain. In that way, IP packets are routed inside the local domain through the shortest path within the boundaries of the local domain according to the IGP metric.

The last rule of the BGP decision process chooses as the best route the one whose `router-id` (IP address of the BGP peer) is the lowest. This last rule ensures that no two routes towards the same prefix can be chosen since at most one route towards any given prefix is advertised by a BGP peer.

According to the BGP-4 Internet draft [138], all rules other than the first one are called the "tie-breaking" rules of the BGP decision process. From our viewpoint, the choice of this word "tie-breaking" is unfortunate, if not misleading. The "official" metric for the BGP routes according to the Internet draft is thus the value of their `local-pref` attribute. Given that this attribute is not present in routes exchanged between eBGP routers but its value is set through the input filters of the local BGP routers of the domain, rules 2 to 7 of the decision process shown on Figure 5 have no meaning concerning the quality of the routes. In this thesis on the other hand, we define the "tie-breaking" rules as "*the subset of the rules of the BGP decision process that allow an AS to choose a best route among those that cannot be differentiated according to the rules that are meaningful for the considered AS*". The drawback of the previous definition concerns its dependence on the AS being considered. This definition however admits the consideration of different subsets of the rules of the BGP decision process as the "tie-breaking" rules. Some ASes may simply not realize which rules constitute their actual "tie-breaking" rules. For instance, stubs rarely use rules beyond the second one to differentiate the quality of their routes so all rules after the second act as "tie-breaking" rules. Transit providers on the other hand often consider the rules up to the sixth one to assess route's quality, only rule seven acts as a "tie-breaking" rule. Figure 6 illustrates the three different views of the "tie-breaking" rules according to the BGP-4 Internet draft [138], transit providers and stubs.

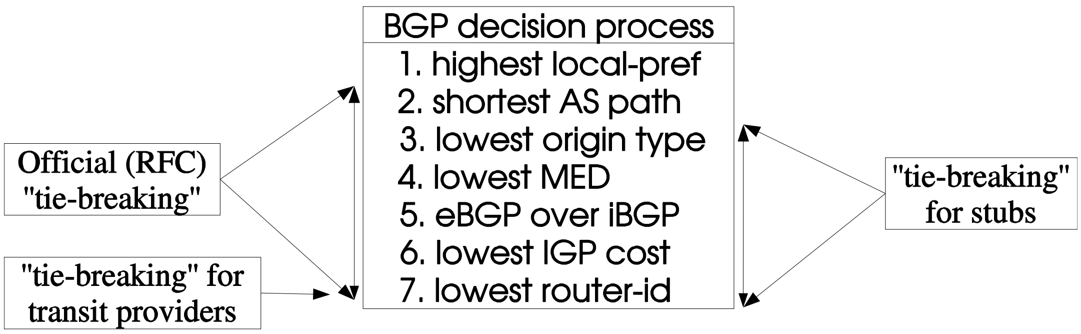



Figure 6: Various perspectives on the meaning of "tie-breaking" rules in the BGP decision process.

## 0.5 Interdomain Traffic Engineering

Interdomain traffic engineering requirements are diverse and often motivated by the need to balance the traffic on links with other ASes and to reduce the cost of carrying traffic on these links. These requirements depend on the connectivity of an AS with others but also on the type of business handled by this AS.

The connectivity between ASes is mainly composed of two types of relationships. The most frequent relationship between ASes is the *customer-to-provider* relationship where a customer AS pays to use a link connected to its provider. The customer obtains a global connectivity to the Internet through its providers and becomes reachable by all other ASes in the Internet. This relationship is the source of most of the interdomain cost of an AS. A stub AS usually tries to maintain at least two of these links for performance and redundancy reasons [157]. In addition, larger ASes try to obtain *peer-to-peer* relationships with other ASes and then share the cost of the link with the other AS. Negotiating the establishment of those *peer-to-peer* relationships is often a complicated process since technical and economical factors, as exposed in [21], need to be taken into account.

Moreover, an AS will want to optimize the way traffic enters or leaves its network, based on its business interests. For instance, content providers that host a lot of web or streaming servers and usually have several customer-to-provider relationships with transit ASes will try to optimize the way traffic leaves their networks. On the other hand, access providers that serve small and medium enterprises, dialup or xDSL users typically wish to optimize how traffic enters their networks. Finally, a transit AS will try to balance the traffic on the multiple links it has with its peers.

Optimizing the way traffic enters or leaves a network means to favor one link over another to reach a given destination or to receive traffic from a given source. Interdomain traffic engineering of this kind can be performed by tweaking the BGP routes of the AS.

Interdomain traffic engineering can be used by operators having different requirements in mind. The first type of requirements originates from stub AS that host a lot of content. Usually, these AS have several customer-to-provider relationships with larger AS and need to optimize how their content leaves their own network. On the other hand, AS that provide Internet access to dialup or broadband users typically wish to optimize how the Internet traffic enters their network. The traffic engineering goal of a content provider could thus conflict with the traffic engineering goal of an access provider. Finally, the transit AS that connect the access and the content providers together also have their own interdomain traffic engineering requirements. Those may also conflict with the requirements of other AS.

### 0.5.1 Control of the outgoing traffic

To control how the traffic leaves its network, an AS must be able to choose which route will be used to reach a particular destination through its peers. Since an AS controls the decision process on its BGP routes, it can influence the selection of the best path. Two techniques are frequently used.

A first technique is to rely on the `local-pref` attribute. This optional attribute is only distributed inside an AS. It can be used to rank routes and is the first criteria of the BGP decision process (center of Figure 5). For example, consider a stub AS with two links toward one upstream provider : a high bandwidth link and a low bandwidth link. In this case, the BGP router of this AS could be configured to insert a low `local-pref` inside routes learned via the low bandwidth link and a higher value to routes learned via the high bandwidth link. A similar situation can occur for a stub AS connected to a cheap and a more expensive upstream provider.

In practice, the manipulation of the `local-pref` attribute can also be based on passive or active measurements. Recently, a few companies have implemented solutions [30, 12] that allow multi-homed stub ASes and content-providers to engineer their interdomain traffic. These solutions usually measure the load on each interdomain link −some also rely on active measurements− to evaluate the performance of interdomain paths. Based on these measurements and some knowledge of the Internet topology, they attach appropriate values of the `local-pref` attribute to indicate which route should be considered as the best one by the BGP routers.

A second technique, often used by large transit providers, is to rely on the intradomain routing protocol to influence how a packet crosses the network of the transit provider. As shown on Figure 5, the BGP decision process will select the nearest IGP neighbor when comparing several equivalent routes received via an iBGP session. For example, consider in Figure 4 that router $R_{27}$ receives one packet whose destination is $R_{45}$. The BGP decision process of router $R_{27}$ will compare two routes towards $R_{45}$, one received via $R_{28}$ and the other received via $R_{26}$. By selecting router $R_{28}$ as the exit border router for this packet, `AS2` ensures that this packet will consume as few resources as possible inside its own network. If a transit AS relies on a tuning of the weights of its intradomain routing protocol as described in [76], this tuning will indirectly influence its outgoing traffic.

## 0.5.2 Control of the incoming traffic

The first method that can be used to control the traffic that enters an AS is to rely on selective advertisements and to announce different route advertisements on different links. Note that such behavior is considered as a wrong behavior on *peer-to-peer* relationships by some ISPs. Taking for example the topology given on the top part of Figure 4, if `AS1` wants to balance the traffic coming from `AS2` over the links $R_{11} - R_{21}$ and $R_{13} - R_{27}$, it could announce only its internal routes on the $R_{11} - R_{21}$ link and only the routes learned from `AS5` on the $R_{13} - R_{27}$ link. Since `AS2` would only learn about `AS5` through router $R_{27}$, it would be forced to send the packets whose destination belongs to `AS5` via router $R_{27}$. Yet if the link $R_{13} - R_{27}$ fails, then `AS2` would not be able to reach `AS5` through `AS1`. This is not desirable and it should be possible to utilize link $R_{11} - R_{21}$ for the packets toward `AS5` at that time without being forced to change the routes that are advertised on this link.

A variant of the selective advertisements is the advertisement of more specific prefixes. This advertisement relies on the fact that an IP router will always select in its forwarding table the most specific route for each packet (i.e. the matching route with the longest prefix). For example, if a forwarding table contains both a route toward `16.0.0.0/8` and a route toward `16.1.2.0/24`, then a packet whose destination is `16.1.2.200` would be forwarded along the second route. The advertisement of more specific routes can also be used to control the incoming traffic. Assume for instance that prefix `16.0.0.0/8` belongs to `AS3` and that several important servers are part of the `16.1.2.0/24` subnetwork. If `AS3` prefers to receive the packets toward its servers on the $R_{24}$-$R_{31}$ link, then it would advertise both `16.0.0.0/8` and `16.1.2.0/24` on this link and only `16.0.0.0/8` on its other external links. This solution has the advantage that if link $R_{24}$-$R_{31}$ fails, then the subnetwork `16.1.2.0/24` would still be reachable through the other links.

Another method would be to allow an AS to indicate a ranking among the various route advertisements that it sends. Based on the use of the length of the `AS path` as the second criterion in the BGP decision process, a possible way to influence the selection of routes by distant ASes is to artificially increase the length of the `AS path` attribute. Coming back to the top part of Figure 4, assume that the primary interdomain link of `AS3` is link $R_{61} - R_{45}$ while link $R_{61} - R_{36}$ is only used as backup primary link. In this case, `AS6` would announce its routes normally on the primary link with an `AS path` of `AS6` while it would add its own AS number several times instead of once in the `AS-Path` attribute on the $R_{61} - R_{36}$ link. The route advertised on the primary link will be considered as the best route by all routers that do not rely on manually configured settings for the `local-pref` attribute. This technique can be combined with selective advertisements. For example, an AS could divide its address space in two prefixes $p1$ and $p2$ and advertise prefix $p1$ without prepending and prefix $p2$ with prepending on its first link and the opposite of its second link.

The last method to allow an AS to control its incoming traffic is to rely on the `MED` attribute. This optional attribute can only be used by an AS multi-connected to another AS to influence the link that should be used by the other AS to send packets toward a specific destination. The utilization of the `MED` attribute is usually subject to a negotiation between the two peering ASes and some ASes do not accept to take the `MED` attribute into account in their decision process.

### 0.5.3 Community-based Traffic Engineering*

In addition to these techniques, several providers have been using the `communities` attribute to give their customers a finer control on the redistribution of their routes. The `communities` attribute is an optional attribute that can be attached to routes. This attribute can contain several 32 bits wide community values. Community values are often used to attach optional information to routes such as a code representing the city where the route was received or a code indicating whether the route was received from a peer or a customer. The community values can also be used for traffic engineering purposes. For instance, predefined `community` values can be attached to routes in order to request actions such as not announcing the route to a specified set of peers, prepending the `AS path` when announcing the route to a specified set of peers or setting the value of the `local-pref` attribute. However, this technique relies on an ad hoc definition of community values and on manual configurations of BGP filters which makes it difficult to use.

The IETF is currently considering the definition of a new standard type of extended communities that are called "redistribution communities" [28] to answer the shortcomings related to the use of classical `communities` for traffic engineering purposes. Redistribution communities can be attached to routes to influence the redistribution of these routes by the upstream AS. The redistribution communities attached to a route contain both the traffic engineering action to be performed and the identification of the BGP peers that are affected by this action. One of the supported actions allows an AS to indicate to its upstream peer to not announce the attached route to some of its BGP peers.

Another type of action allows an AS to request its upstream peer to perform `AS path` prepending when redistributing a route to a specified peer. To understand the usefulness of such redistribution communities, let us consider again the top part of Figure 4, and assume that `AS6` receives a lot of traffic from `AS1` and `AS2` and that it would like to receive the packets from `AS1` (respectively `AS2`) on the $R_{45}$-$R_{61}$ link (respectively the $R_{36}$-$R_{61}$ link). `AS6` cannot ensure such a traffic distribution by performing `AS-Path` prepending itself. However, this becomes possible with the redistribution communities by requesting `AS4` to perform the prepending when announcing the `AS6` routes to external peers. `AS6` could thus advertise to `AS4` its routes with a redistribution community that indicates that this route should be prepended two times when announced to `AS2`. With this redistribution community, `AS4` would advertise AS path `AS4:AS4:AS6` to `AS2` and AS path `AS4:AS6` to `AS1`. `AS2` would hence receive two routes toward `AS6`, `AS4:AS4:AS6` and `AS3:AS6`, and `AS2` would select the route via `AS3`. `AS1` on the other hand would select the `AS4:AS6` route having a shorter AS path than the `AS2:AS3:AS6` route.

### 0.5.4 Limitations

The sections above have described several techniques that can be used by ISPs to engineer their interdomain traffic. However, there are some limitations to be considered when deploying those techniques.

A first point to note is that the control of the outgoing traffic with BGP is based on the selection of the best route among the available ones. This selection can be performed on the basis of various parameters, but is limited by the diversity of routes received from neighboring ASes. The diversity of the routes received depends on the connectivity and the policy of these neighboring ASes. At most one route towards each prefix will be received from any BGP neighbor.

The control of the incoming traffic is based on a careful tuning of the advertisements sent by an AS. This tuning can cause several problems. First, an AS that advertises more specific prefixes or that divides its address space into distinct prefixes to announce them selectively will advertise a number of prefixes larger than required. These prefixes will be propagated throughout the global Internet and will increase the size of the BGP routing tables of potentially all ASes in the Internet. [33] reports that more specific routes constituted more than half of the entries in a BGP table. Faced with this increase of their BGP routing tables, several large ISPs have started to install filters to ignore the BGP advertisements corresponding to more specific prefixes [25]. The deployment of those filters implies that the more specific prefixes will not be announced by those large ISPs. Using more specific prefixes will therefore become much less effective.

When considering the manipulation of the `AS path` attribute, we have mentioned that it can be used on backup links. It is sometimes also used to better balance the traffic. For instance, [33] reports that `AS path` prepending affected 6.5 percent of the BGP routes in November 2001. However, in practice it can be difficult to predict the outcome of performing `AS path` prepending on a given interdomain link [29]. Usually, providers that rely on `AS path` prepending try to find out the right amount of prepending on a trial and error basis.

The redistribution communities can provide a finer granularity than `AS path` prepending or selective announcements. In practice, it can be expected that those communities will be used to influence the redistribution of routes toward transit ISPs having a large number of customers.

An important point to bear in mind concerning the techniques that require changes to the attributes of the BGP advertisements is that any change to an attribute will force the route advertisement to be redistributed to potentially the entire Internet. Although it would be possible to define techniques relying on measurements to dynamically change the BGP advertisements of an AS for traffic engineering purposes, a widespread deployment of such techniques would increase the number of BGP messages exchanged and could lead to BGP instabilities [103, 86]. Any dynamic interdomain traffic engineering technique that involves frequent changes to the values of BGP attributes should be studied carefully before being deployed.

Finally, several features exist in BGP implementations to engineering the traffic. Cisco's BGP multipath [43] allows to load-balance outgoing traffic when several equal cost (only the router-id differs) eBGP routes are learned from a neighboring AS. Per-packet or per-destination load balancing is then performed by the router among the multiple paths. Unequal cost load-balancing is also possible by relying on the extended community attribute [146]. However, Cisco's BGP load-sharing techniques [44] are limited to multiple links on a single router and do not work well with multiple routers. In addition, these techniques are rather limited in their scope: outgoing load-balancing (packet-based or destination-based) and static incoming traffic distribution based on static relative available link bandwidth. The same features are also available on Juniper routers [96].

Table 1 provides a summary of BGP-based traffic engineering methods. Table 1 gives the scope (local or remote) of each method, the traffic direction concerned (inbound or outbound), and the conditions under which the traffic engineering method works.

| **Technique** | **Scope** | **Traffic** | **Conditions** |
|---|---|---|---|
| `local-pref` | local AS | outbound | always works |
| selective announcements | Internet | inbound | always works |
| more specific prefixes | Internet | inbound | if no filtering of long prefixes |
| `AS-Path` prepending | Internet | inbound | unsure |
| `MED` | neighbors AS | inbound | requires bilateral agreement |

Table 1: Summary of BGP-based traffic engineering methods.

## 0.6 Conclusion

In this first chapter, we have described several techniques that are used today to control the flow of packets in the global Internet. We have described the current organization of the Internet and the key role played by BGP. We have explained how BGP is tuned today for interdomain traffic engineering purposes. We have shown that an AS has more control on its outgoing than on its incoming traffic. Several techniques can be used to control the incoming traffic, but they have limitations. The selective advertisements and the more specific prefixes have the drawback of increasing the size of the BGP routing tables. With `AS path` prepending, it can be difficult to select the appropriate value of prepending to achieve a given goal. Finally, we have explained how the redistribution communities could allow an AS to flexibly influence the redistribution of its routes toward non-directly connected ISPs.

# Part II

# Interdomain traffic characterization

In this second part, we analyze in details several interdomain traffic traces to develop a better understanding of the behavior of the traffic. Understanding the behavior of the traffic is important to realize how variable interdomain traffic is at many timescales. In the context of interdomain traffic engineering, the topological distribution of the traffic is particularly important since it will indicate the number of interdomain sources and destinations that will have to be influenced by the traffic engineering technique. This is not to say that traffic dynamics is unimportant. The dynamics of the traffic has deep implications on its prediction and control. Trying to engineer the traffic over timescales where it is known to be too variable can be expected to be more difficult than required. Before thinking about controlling the traffic, one must first understand which aspects of its behavior can be neglected when engineering it.

In Chapter 1, we analyze the behavior of the traffic dynamics that enters or leaves a stub AS over timescales ranging from seconds to several days. We then go to study the topological distribution of the interdomain traffic in Chapter 2. Finally, Chapter 3 analyzes the dynamics of the traffic on the interdomain topology.

# Chapter 1

# Interdomain traffic dynamics

## 1.1 Introduction

It could be argued that the aspect of the dynamics of the interdomain traffic is not fundamentally relevant to interdomain traffic engineering. The purpose of this chapter is to show evidence of the contrary. Many problems in engineering require the knowledge of the timescales over which controlling a system is possible. For most control applications, too wide fluctuations in the systems dynamics mean that another timescale should be used for controlling it. In this chapter, we analyze the behavior of the interdomain traffic to understand on which timescales its behavior seems smooth enough for traffic engineering be possible over these timescales. Note that parts of this chapter have appeared in [173, 177].

This chapter is structured as follows. Section 1.2 first introduces the main concepts concerning scaling processes. Section 1.2.1 goes on by discussing the practical issues of detecting scaling in real processes. Section 1.3 deals with the total traffic long-term self-similarity. Section 1.4 then considers the scaling in the process of the TCP flows arrivals. Finally, section 1.5 discusses the contributions of this chapter.

## 1.2 Self-similarity and scaling processes

### 1.2.1 Scaling, self-similarity, and LRD*

*Scaling* behavior is a general term connected with the absence of a particular characteristic scale controlling the process under study. For an introduction to scaling in network traffic see [4]. [129, 62] provide detailed treatments of various aspects concerning scaling in network traffic. For more general discussions of scaling in natural phenomena see [152, 88].

*Self-similarity* on the real line, or more precisely self-affinity [113], with parameter $H$ $(0 < H < 1)$, is defined by

$$\{X(t), t \in \mathbb{R}\} \stackrel{d}{=} \{c^H X(t/c), t \in \mathbb{R}\}, \forall c > 0, \tag{1.1}$$

where $\stackrel{d}{=}$ denotes equivalence in distributions. Equation 1.1 means that the distribution of the original process $X$ and the distribution of the rescaled process $c^H X(t/c)$ are the same. For $c > 1$, rescaling corresponds to zooming in (finer resolution), equivalent to viewing the process $X$ at smaller timescales. $c < 1$ on the other hand corresponds to a zooming out of the process or a larger timescales perspective of the dynamics of the process $X$. The reason for the name "self-similarity" is that all timescales of the process are related to one another through this relationship between the distributions of the rescaled process, hence a similarity of the process with itself. The distributions of the process $X$ seen at any timescale are connected through the affine relation defined by equation 1.1. The parameter $H$ controls the behavior of the process $X$ at all timescales.

The following property of self-similar processes concerns the scaling of their moments

$$E|X(t)|^q = E|X(1)|^q \ |t|^{qH} \ . \tag{1.2}$$

This latter property expresses the behavior of moment $q$ (if it exists) of the process as a power law of time, defining a high-order scaling which is linear in $H$. A self-similar process will exhibit the same fluctuations (after rescaling) independent of the considered scale, hence cannot be stationary due to the dependency of all its moments on time. Recall that for a process to be stationary, its statistical properties must not depend on time.

In the networking literature, strict self-similarity as defined in equation 1.1 is rarely used. Rather, second-order stationarity is assumed, meaning that both the mean of the process and its variance are finite and time-invariant. This second-order stationarity assumption allows for an easier statistical analysis of the process' properties [27]. In addition, in practice even if the original signal to be analyzed is continuous, one deals with discrete-time sample of the process.

Statistical *self-similarity* in the context of discrete-time processes is defined through the following procedure. Let $X = \{X_i, i \geq 1\}$ be a second-order stationary sequence and build from it the following sequence

$$X^{(m)}(k) = \frac{1}{m} \sum_{i=(k-1)m+1}^{km} X_i, \quad k = 1, 2, ... \tag{1.3}$$

Formula 1.3 defines the *m-aggregated* sequence obtained by partitioning the original sequence $X$ into non-overlapping blocks of length $m$ and viewing the block average as a new point of the sequence $X^{(m)}$. The series $X^{(m)} = \{X_k^{(m)} : k = 1, 2, ...\}$ is said to be *asymptotically self-similar* if

$$X \stackrel{d}{=} m^{1-H} \ X^{(m)} \quad as \quad m \to \infty \tag{1.4}$$

where $\stackrel{d}{=}$ denotes equivalence in distribution. The previous property can be seen as the discrete-time equivalent of self-similarity on the real line of equation 1.1. The statistical self-similarity property states that all *m-aggregated* sequences ($X^{(m)}$) rescaled by $m^{1-H}$ have the same distribution as the original process $X$. If the sequence is not second-order stationary, then the distribution of the normalized *m-aggregated* sequence might not converge. Asymptotically, the first two moments of the process might even be infinite, in which case the procedure used for building the *m-aggregated* sequence is meaningless. Deviations about second-order stationarity might bias the convergence of the distributions, even though finite discrete-time processes have finite moments. Note that we use the term "sequence" to refer to the succession of samples in time while the term "series" denotes the set of the *m-aggregated* sequences for $m = 1, 2, \ldots$

When one restricts the study of the process to its second-order properties, then comes LRD. Long-range dependence (LRD), or long-memory, is associated with second-order stationary processes. A second-order stationary process displays LRD [27, 186] if its autocovariance $r(k)$ behaves as

$$r(k) \sim c_r |k|^{2H-2} \;\; as \;\; |k| \to \infty \tag{1.5}$$

when $1/2 < H < 1$. In such a case, the correlations decay so slowly that $\sum_{k=-\infty}^{\infty} r(k) = \infty$. When $H = 1/2$, the process is uncorrelated ($r(k) \sim 0$). On the other hand, when $0 < H < 1/2$, we speak of short-range dependence (SRD), or anti-persistence, due to negative dependence ($r(k) < 0$ for $k \neq 0$) between time lags. The divergence of the correlations when $1/2 < H < 1$ also means that the spectral density behaves as

$$f(\lambda) \sim c_f |\lambda|^{1-2H} \;\; as \;\; |\lambda| \to 0, \tag{1.6}$$

hence diverges at the origin ($f(0) = \infty$). But because of stationarity, LRD processes cannot exhibit fluctuations at any arbitrary scale, as do strictly self-similar processes.

An often mistaken relationship between LRD and self-similarity concerns the implication of LRD by strict self-similarity for $H > 1/2$. There are actually two types of LRD behavior: one that is limited to the strict definition in 1.5 and 1.6 for a second-order stationary process for which strict self-similarity does not hold, and the consequence of LRD that arises for a strictly self-similar process for $H > 1/2$. The problem in the case of self-similar processes for whom second-order stationarity does not hold is that any second-order analysis is biased by the non-stationarity of the process. Spectral and correlation analysis thus become tricky because the results of these analyses might be caused by the second-order properties of the process as well as bias from non-stationarity [132]. This distinction is not merely formal, but has important practical consequences on the type of process considered. For example, it is not always possible to distinguish in discrete signals between pure statistical correlations that span more than the length of the studied sample, pure LRD and LRD produced by a self-similar process. Only the knowledge of the physics of the process can make the difference between these processes or to rely on a longer time-series, as will be illustrated in the next section.

### Scaling detection in practice

The previous section dealt with theoretical properties of signals. The purpose of this section is to sensitize the reader to the difficulty of scaling detection and the bias caused by non-stationarity. Because in real signals neither LRD nor self-similarity occur in an asymptotic sense, these two properties point out to two different types of signals. LRD is related to statistical dependence within the time-series. In practice dependence in a time series has an upper cutoff value corresponding to the largest correlation length between two time instants. By "largest correlation length between two time instants", we mean the smallest distance between two time instants that makes the autocorrelation fall below the 0.95 confidence limit of $2/\sqrt{N}$ where $N$ is the length of the sample. The autocorrelation of a time-series is the correlation between the original time-series and time-shifted versions of the time-series. The autocorrelation for time lag $t$ measures the correlation within the time-series between the samples separated by $t$ time intervals. The upper cutoff value is the smallest value of the time lag after which the autocorrelation of the time-series falls below this $2/\sqrt{N}$ confidence limit. LRD in this sense manifests from the smallest timescale up to this upper cutoff value.

Equation 1.4 defining statistical self-similarity does not care about any cut-off value because it is asymptotic, the definition only defines a relationship between the original process and its *m-aggregated* series for $m$ going to infinity. The relation in Equation 1.4 holds for large values of $m$, starting at some value of $m$. Practical constraints however require that the length of the sample be finite, so only the *m-aggregated* series up to a given value of $m$ can be computed. The time-series analyst is thus faced with the double issue of finding the value of $m$ from which statistical self-similarity seems to hold, and of relying on a sufficiently long time-series to be able to estimate the value of $H$ in Equation 1.4.

Self-similarity is a sort of global dependence arising through the consistent relationship between the original time series and its *m-aggregated* series. While this self-similarity implies LRD for $H > 1/2$ in the second-order stationary case, an upper cut-off value for the correlation length of a LRD signal will make a non-negligible difference between self-similarity and LRD in practice. Because LRD stops at this cutoff value (call it $C$), which can be smaller than the length of the time-series, the *m-aggregated* series of the LRD signal will not be self-similar any more for $m > C$ generally speaking.

To illustrate this point, Figure 1.1 compares the *m-aggregated* series of a LRD signal with an upper correlation length cutoff value $C$ of one thousand time units with a self-similar process (fractional Brownian motion) with Gaussian increments (fractional Gaussian noise). The units have no meaning on Figure 1.1. The LRD signal has been generated by inducing a correlation length of one thousand by generating a stationary Poisson process of mean equal to ten, looping over the whole length of the generated series and adding the current value of the Poisson process to every time interval for the next one thousand time units, creating a dependence of lag one thousand in the time series. The self-similar process has been obtained by integrating the Gaussian noise[1]. Both processes start at a value of ten thousand units and end close to each other, but the LRD process is known to be second-order stationary while the self-similar is non-stationary.

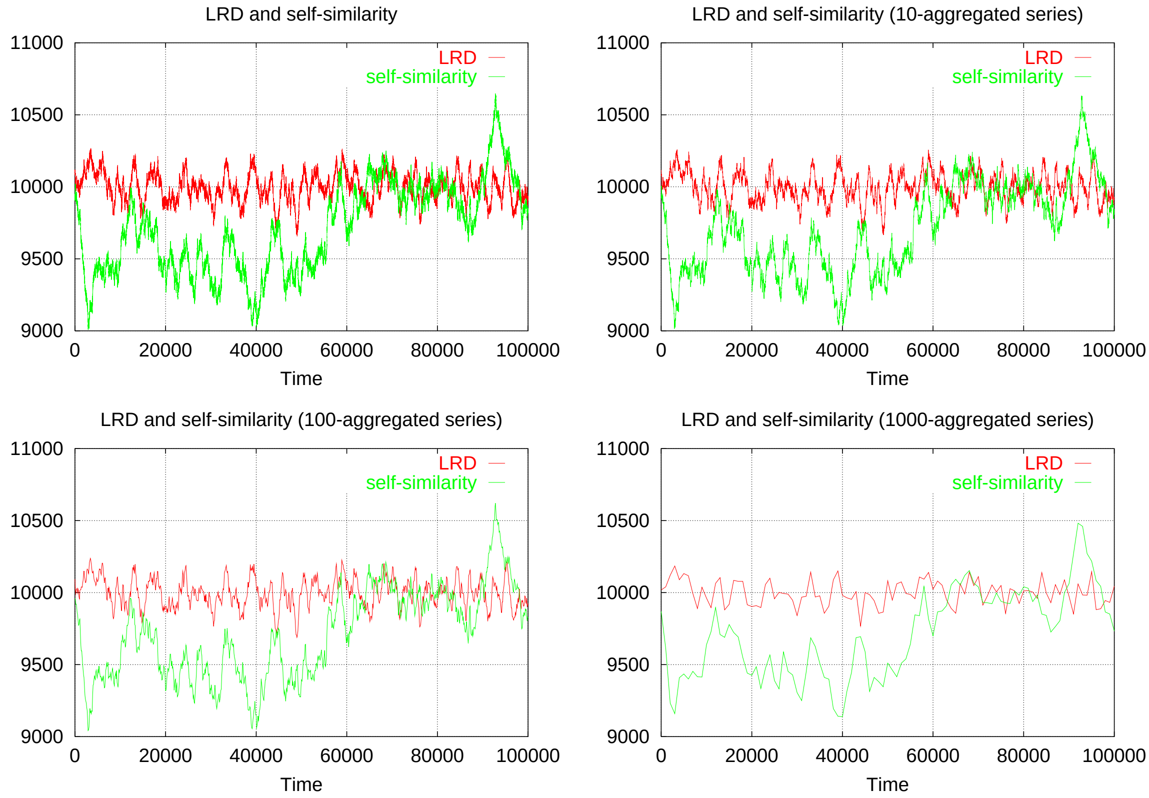



Figure 1.1: Comparing LRD and self-similarity in practice.

The two signals compared in Figure 1.1 can be said to be qualitatively different because we know how they were generated. Suppose on the other hand that one has to analyze these two signals without knowing beforehand that one is LRD and the other self-similar. For example, suppose that we want to check for LRD. We thus apply the definitions of LRD (from formula 1.6), considering that both processes are second-order stationary. The reader might contend that the self-similar signal of Figure 1.2 is *obviously* non-stationary. This is true indeed, because we chose the length of the time-series to be sufficiently long so that non-stationarity be visible on Figure 1.1. If we had samples corresponding to the signals within the [70000,80000] time interval only, non-stationarity would not appear obvious. Many processes are physically constrained in the extent of their variation. For example, a finite number of users may connect to a web server at a given instant, or the size of the traffic bursts a single machine can send is limited by its network access capacity. The boundary between stationarity and non-stationarity is thus difficult to decide [41, 122].

When comparing the spectrum of the two signals (left of Figure 1.2), we do not see a significant difference. The graphs of the two signals however (Figure 1.1)

[1] More sophisticated techniques to generate self-similar or LRD processes exist [20, 53].

show that the LRD process seems stationary while the self-similar one seems not. The autocorrelation function of the two processes (right of Figure 1.2) on the other hand shows the difference between them. The stationary signal does not exhibit correlations beyond a time lag of one thousand since its correlations fall below the $2/\sqrt{N}$ confidence limit after a time lag of one thousand. The self-similar signal biases the autocorrelation analysis so much that the statistics tells that correlations are still close to one even for a time lag of one thousand.

Even though self-similarity and LRD are strongly related, these two concepts correspond to different processes. It is easy to show in synthetic signals that a process is second-order stationary while another is non-stationary. In practice many signals are non-stationary, in which case the time-series analyst has to decide what is non-stationarity and what type of de-trending should be performed. Removing non-stationarity from a time-series requires that one makes assumptions on the properties of the signal at each timescale. When different types of signals mix within the different timescales, then this de-trending is not possible without a-priori assumptions on the nature of the signal. For instance, in the case of the two synthetic signals we provided in this section, the right part of Figure 1.2 showing the autocorrelation allowed to see the difference between the two signals because we knew that one of them had limited correlations. If we had not known that correlations allow to see the difference between the two signals shown on Figure 1.1, we would have had to wait to find the estimator that analyzes the right property of the signal to be able to know exactly in which way non-stationarity biases other estimators.

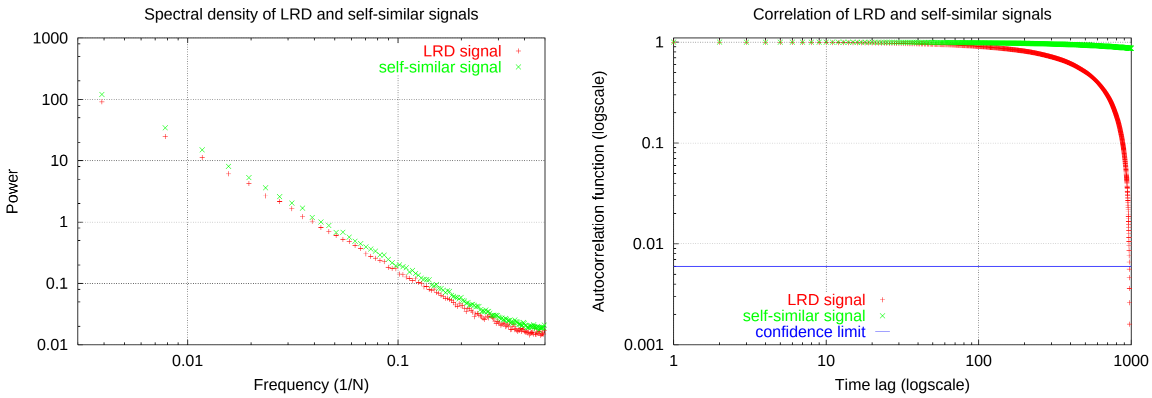



Figure 1.2: LRD from statistical correlation and self-similarity.

### 1.2.2 Identification of scaling processes

In this section, we review the classical (non wavelet-based) methods for the analysis of scaling processes (and the Hurst parameter $H$). We only consider the basic ones, not their refinements, because it is mainly the aim and basic assumptions of a method that guides what can be inferred from it, refinements mostly affect the statistical bias of the analysis technique [161]. All of them are graphical, and consist in plotting a statistics on a log-log plot and determining a region where a linear behavior appears. The value of $H$ is then inferred based on a least-square regression over the region of the graph where scaling is visually identified [27].

The first estimator is the well-known $R/S$ statistic [27, 113]. Plotting the $R/S$ statistic for large $k$ on a log-log scale should be scattered around a straight line of slope $H$. A nice property of the $R/S$ statistic is its robustness with respect to long-tailed marginal distributions, namely departure from Gaussianity [110]. See part VII of [113] for an in-depth treatment of the $R/S$ statistic.

The second method for measuring $H$ is the *aggregated variance* method [27], where one computes the sample variance of the each *m-aggregated* sequence $X^{(m)}$ around its sample mean $\bar{X}^{(m)}$

$$s^2(m) = \frac{1}{m-1}\sum_{i=1}^{m}(X^{(m)}(k) - \bar{X}^{(m)})^2. \tag{1.7}$$

Then, the plot of $s^2(m)$ as a function of $m$ on a log-log scale should follow for large values of $m$ a straight line with negative slope $2H-2$. Because this *aggregated variance* method computes the variance of each complete *m-aggregated* sequence and not a local one (for subsets of the sequence), it can be affected by non-stationarity.

The previous two estimators mainly deal with the process variability (second order), hence characterize the extent of the variations in the signal and their persistence with time aggregation.

The third estimator for $H$, the correlogram, relies on the time dependence within the samples. Long-range dependent data should exhibit very slowly decaying sample correlations proportional to $k^{2H-2}$ for $1/2 < H < 1$, $k$ representing the time lag in the series. To say anything meaningful based on the autocorrelation function $r(k)$ requires the assumption of second order stationarity. Variations in the mean and variance in the signal limit the use of this statistic, hence de-trending and variance normalizing should be performed on the signal before computing it.

Finally, while the three previous methods were based on the time-domain, the final one is the periodogram, in the frequency-domain. If the series exhibits *long-memory* (LRD) then the estimated spectral density at the origin should behave as in equation 1.6. Plotting the periodogram near the origin with a log-log scale should roughly follow a straight line of slope $1-2H$. The biggest problem with frequency-domain statistics concerns the physical meaning of “frequency” components [132, 26]. If the signal is due to contain well-defined periodicities, then a smoothed periodogram might provide useful information about the signal's properties. In the case of statistical self-similarity on the other hand, there are not always good reasons to think of meaningful frequency components in the process, hence focusing on too small subsets of the periodogram might pose problems. Anyway, a frequency-domain analysis might show that global scaling exists in the signal through a $1/f$ noise-like behavior in the spectrum [112, 113]. See also [7] for a discussion of the estimation of the spectrum of $1/f$ processes in the wavelet domain.

Using estimators that rely on a stationarity assumption with non-stationary data might bias the estimation [52, 142]. A self-similar signal ideally requires an analysis tool capable of disentangling the scaling properties at each timescale. In the case of a signal containing both dependence and self-similarity, one has to check the cause of this dependence. A common practice in time-series analysis is to model non-stationarity by a trend and remove it from the data before analyzing the signal. In the case of scaling processes however, it is difficult to distinguish between actual periodic components, non-periodic shifts in the mean, or large variations in the signal due to a large variance. Since scaling processes have no particular localization in the frequency-domain (no characteristic timescale), it is often impossible to decide whether one should model the long timescales with a seasonal trend or not model it at all and let self-similarity explain non-stationarity that looks like a periodicity.

### 1.2.3 Wavelet-based scaling detection*

The estimators introduced in the previous sections are the most widely used ones in the literature. Nevertheless, their use for scaling processes is highly questionable because of the bias from non-stationarity in scaling processes. Wavelet-based tools constitute a large step forward in the detection of scaling thanks to a limited bias from non-stationarity in the estimation. A detailed treatment of wavelet analysis is outside the scope of this thesis. Here we only introduce basic properties of wavelets. For a complete treatment of wavelet theory and analysis, we refer respectively to [54] and [5, 4, 6].

Wavelet signal decomposition consists in analyzing a signal $X(t)$ through a bandpass oscillating function $\psi_{j,k}$ where $j$ represents the timescale and $k$ the time instant. Throughout this chapter, small values of $j$ represent the smallest timescales and large values represent the larger timescales. By scaling and shifting this function $\psi$, it is possible to break the signal into its timescale parts (at timescale $j$) and within each timescale along the time axis (at time $k$):

$$\psi_{j,k}(t) = 2^{-j/2}\psi(2^{-j}t - k), \quad j \in \mathbb{Z}, \quad k \in \mathbb{Z} \tag{1.8}$$

where the $\psi_{j,k}(t)$ form an orthonormal basis of $L_2$, the space of twice integrable functions. Any square integrable signal can hence be approximated by a finite linear combination of the $\psi_{j,k}(t)$.

A dilation and shifting operation by a factor $(j, k)$ moves the original frequency $f_0$ of the function $\psi$ to frequency $2^{-j}f_0$ while shifts it in time by a factor of $2^j k$. Recall that larger values of $j$ relate to larger timescales, hence smaller frequencies $(2^{-j}f_0)$. Figure 1.3 shows the tiling of the time-frequency plane when using a dyadic dilation and shifting operators. On Figure 1.3, we show the original frequency $f_0$ and the corresponding time resolution. A dilation operation of factor $j$ moves the original frequency $f_0$ to $2^{-j}f_0$, hence going to a coarser time resolution of a factor $2^j$.

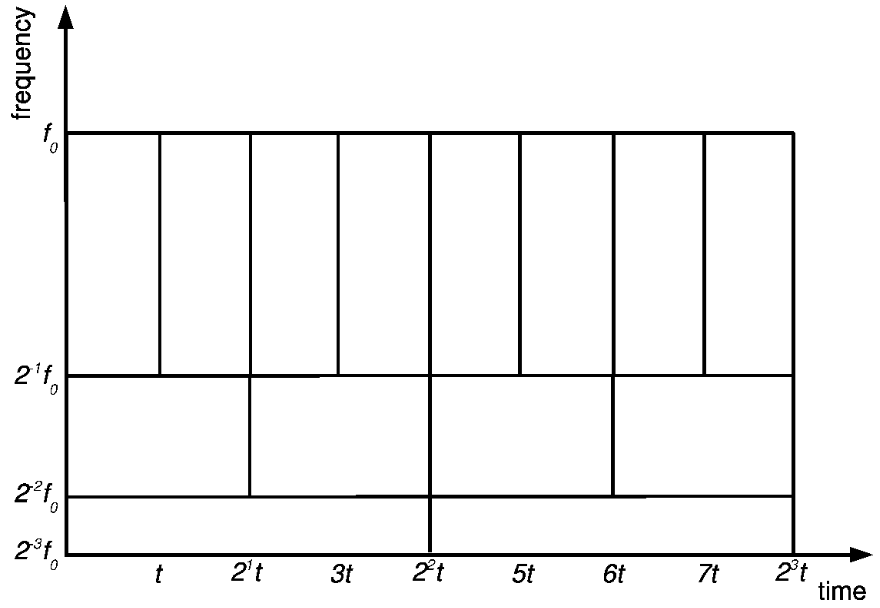



Figure 1.3: Time-frequency plane for dyadic dilation and shifting operators.

The discrete wavelet transform algorithm performs a dyadic tree decomposition (multiresolution analysis) of the signal

$$X(t) = \sum_k c_X(j_0, k)\phi_{j_0,k} + \sum_{j \le j_0}^{J} \sum_k d_X(j,k)\psi_{j,k}(t) \tag{1.9}$$

where the first term on the right-hand side represents an approximation of the signal while the second term represents the details, $j_0$ being the resolution depth, i.e. the coarsest resolution at which the signal is analyzed. The $c_X(j_0, k)$ are called the scaling coefficients and $\phi$ the scaling function. The $d_X(j,k)$ are called the wavelet coefficients and are of special interest because they are used as a substitute for the increments of the process $X$ over the dyadic tree of the time-timescale plane. The orthogonal basis property of $\phi$ and $\psi$ allow the easy computation of the coefficients by simple inner products with the signal. The nice properties of the wavelet transform make the wavelet coefficients better suited for statistical analysis in comparison to the original increments of the process. In addition, the wavelet transform has a $O(n)$ time complexity, making its computation fast. From now on, we shall deal with the $d(j,k)$ instead of the process increments $X_\delta(k)$.

Now let us shortly describe how the wavelet coefficients $d(j,k)$ are computed. For this, we rely on Figure 1.4. During the successive iterations of the wavelet analysis, a discrete time-domain analysis filter corresponding to the wavelet function is applied on the signal at the successive timescales. This analysis filter is a sequence of weights (a filter) that carries out the working of the continuous wavelet function. This analysis filter does two things. First it generates the wavelet coefficients representing the variations in the signal that could be matched by the the filter at the current timescale $j$ ($d(j,k)$). Second, the analysis filter also produces the new signal ($c(j,k)$ on Figure 1.4) to be analyzed at the next timescale $j+1$ which is twice as large as the current timescale. This new signal contains everything in the sequence at the previous iteration that could not be matched by the wavelet function. So actually, the wavelet transform relies on two filters, a high-pass filter that generates the wavelet coefficients, and a low-pass filter that generates the smoothed version of the signal to be analyzed by the next iteration of the algorithm. For further details on the working of the discrete wavelet transform, we refer to [185, 180].

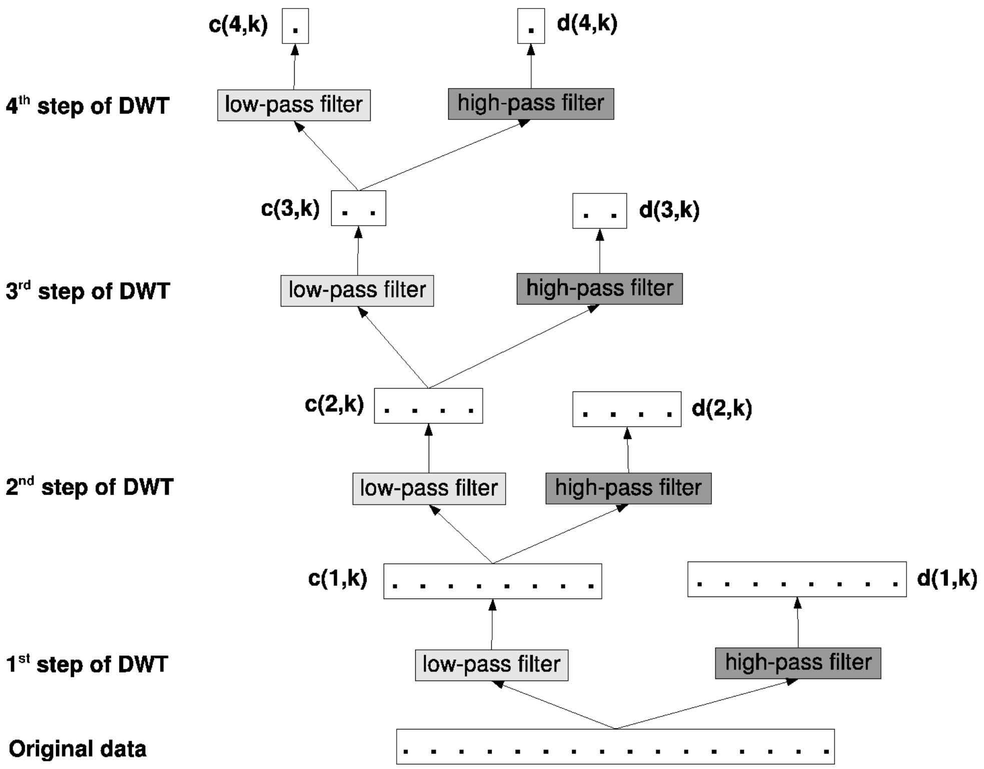



Figure 1.4: Discrete (forward) wavelet transform.

Assume on Figure 1.4 that we start with a sequence containing 16 points on which we perform the discrete wavelet transform. After the first iteration of the algorithm, we get the 8 wavelet coefficients ($d(1,k)$) representing the variations in the origianal signal at a timescale equal to twice the precision of the original sequence. This is because the effect of any iteration of the algorithm is to replace the original sequence by the wavelet coefficients in one half of the sequence length, and by the new sequence ($c(j,k)$) to be analyzed during the next iteration in the other half of the original sequence. The reason for the working of the discrete wavelet transform algorithm is to allow all iterations to be performed in-line in the space used by the original sequence. The original sequence is hence lost but the algorithm has both a space and time complexity of $O(n)$ where $n$ is the length of the original sequence. On Figure 1.4,

The properties that make wavelet analysis suitable for studying scaling processes are the following:

1. scale invariance: the wavelet basis being built through the dilation operator, the analyzing functions family has a built-in scaling property that reproduces scaling present in the data;

2. robustness against non-stationarity: the mother wavelet ($\psi_0$) has a number $N \geq 1$ of vanishing moments $\int t^k \psi_0(t)dt \equiv 0,\ k = 0, ..., N-1$ , allowing the wavelet to remove any polynomial trend of order up to $N-1$;

3. almost decorrelation: under the assumption of $N \geq H + 1/2$, global LRD among the increments of the process can be turned into short-range dependence [164].

These properties ensure that the wavelet transform captures the scaling properties of the signal (point 1) and that a sound analysis of the signal be performed at each timescale independently of other timescales (point 2), with a controlled bias coming from non-stationarity in the signal (point 3).

The second-order wavelet-based estimator of scaling, the *logscale diagram* (LD) [5] consists in plotting $y_j = log_2(\mu_j)$ against $j$ together with the confidence intervals, where

$$\mu_j = \frac{1}{n_j} \sum_{k=1}^{n_j} |d_X(j,k)|^2. \tag{1.10}$$

$n_j$ in Equation 1.10 denotes the number of wavelet coefficients at octave $j$ and $d_X(j,.)$ represent the wavelet coefficients at octave $j$. The *logscale diagram* is a second-order statistic of the wavelet coefficients that measures the variance of the wavelet coefficients at each octave $j$. Scaling detection can be performed through observation of linear alignment of the confidence intervals of the $y_j$ within some octave range. If, and only if alignment is detected, can the estimation of scaling parameters be performed [5].

The slope of the LD indicates qualitative information about the signal's variability. Three cases are possible for the slope of the LD: negative, about zero, and positive. A slope of the LD about zero for some octave range points to a process whose variability is constant over the corresponding timescales. Such a process is uncorrelated, like white noise or a Poisson process. It is not obvious to understand why a zero slope implies an uncorrelated process. A zero slope of the LD means that the extent of the variability of the process is independent of the considered timescale, at least for the concerned timescale range. If the variations in a process do not depend on the considered timescale, this process cannot contain correlations that are not caused by non-stationarity. To understand the implication of correlations on scaling relationships in a signal, it is necessary to explain what correlations actually are. Suppose a zero mean process. Positive correlations in such a signal mean that whenever the value of the signal deviates (positively or negatively) from zero, then the probability that successive values of the signal will take values that deviate from the mean in the same way is larger than 1/2. Successive positive or negative values are more likely than a positive value followed by a negative one on average. Negative correlations on the other hand mean that the probability is larger than 1/2 for deviations about the mean in opposite directions for successive samples. Hence a positive value followed by a negative one is more likely than successive values of the same sign. Now let us consider the autocorrelation function of the zero-mean process. The autocorrelation function indicates how on average values separated by $t$ time intervals are correlated over the whole process. A positive (negative) value of the autocorrelation function for time lag $t$ means that samples of the process separated by $t$ time units are positively (negatively) correlated. If any time lag of the autocorrelation function exhibits a different value compared to other time lags, there will be a shift in the LD for the corresponding timescale. The reason for this phenomenon is that large correlations (positive or negative) at time lag 4 for instance means a larger likelihood of experiencing large deviations about the mean in the process every four samples, hence more energy in the LD for octave two. Up to now, we have explained how correlations at some lag may imply changes in the LD for some octave. Scaling on the other hand concerns relationships among timescales in a process.

To understand how relationships among timescales can give us information about a process, it is best to come back at the procedure of the discrete wavelet transform. Assume that we actually apply the successive iterations of the algorithm on a zero-mean sequence. The wavelet coefficients at the first step will serve for computing the value of the LD at octave 1. Each successive iteration of the algorithm removes the variability matched at the current timescale analyzed by the algorithm. Getting a zero slope of the LD means that all timescales provide the same amount of variability. Now recall the third nice property of wavelets mentioned in section 1.2.3, the decorrelation of the wavelet coefficients at each timescale. This property ensures that the variance computed by the LD is the variance of an uncorrelated process, the wavelet coefficients at the particular timescale considered. In addition this process is very close to be zero-mean since the integral of the wavelet is zero (the first vanishing moment, moment zero). Hence the LD measures the variance of a zero-mean uncorrelated process. Note however that this process is not second-order stationary because although almost uncorrelated, the wavelet coefficients might have a non-stationary variance.

Now we come back to the problem of understanding the behavior of the LD for different types of signals. Each iteration of the discrete wavelet transform algorithm removes the contribution to the variability of the signal at the current timescale. What does mean that the contribution to the global variability at each timescale is the same ? It means that the variability of the signal is not explained more by some timescale than some other. Another way to put it is to say that each timescale is responsible for the variations in the signal in exactly the same way. No variation in the signal occurs for some time lag preferentially: the variations in the process are random. If some correlation existed for some time lag in the process, the variations in the signal at this timescale would be larger, so would be the value of the LD for this timescale only. This explains that a slope of zero in the LD requires either no correlations or a constant autocorrelation for all time lags. For stationary signals, a constant autocorrelation function only occurs for white noise (zero correlations) or a constant signal (see [26] p. 135). The case of a constant signal however is uninteresting. A zero slope for the LD therefore implies a scaling process with $H = 1/2$.

The case of a positive slope of the LD on the other hand means that successive timescales contribute to increasingly large fractions of the total variability of the signal. The successive iterations of the discrete wavelet transform find wavelet coefficients with larger and larger variances. If the process is second-order stationary, the increasing variance for the successive timescales cannot come from non-stationarity but only from homogeneous variability in the signal corresponding to large scale correlations. In the case of LRD signals, the positive slope extends to the timescale corresponding to the upper cutoff time lag, the largest correlation length allowed by the physics of the process. LRD is defined in equation 1.5 through an autocorrelation function that decreases geometrically as the time lag goes to infinity. Even though the discrete wavelet transform removes the correlation contained in the timescale analyzed by the current iteration of the algorithm, the successive timescales analyzed by the wavelet transform deal with increasingly large fractions of the autocorrelation's time lags. This increasing span of the time lags in the correlations of the process allow the variability to build up for increasing timescales, explaining the larger variances of the wavelet coefficients for larger timescales. In the case of strict self-similarity on the other hand, the positive slope of the LD is not caused by physical correlations in the signal but by the larger variability that arises with the scaling of the moments with time of equation 1.2. Even if the extent of the variations in the self-similar signal are limited by the physics of the process, LRD arising for $H > 1/2$ is enough to explain the positive slope of the LD. It was shown in [5] however that the slope of the LD for a strictly self-similar signal will be different from the slope arising due to a LRD one.

The case of a negative slope of the LD is quite similar to the one of the positive slope. The negative slope of the LD means that increasing timescales see a smaller variability. This behavior points to an exponentially decreasing autocorrelation function for which the larger span of the time lags for the successive iterations of the wavelet transform algorithm do not permit the correlations to build up as in the case of LRD. Such a process must have short-range dependence (anti-persistent correlations), so $H < 1/2$.

## 1.3 Long-term self-similarity of interdomain traffic

The previous section has introduced the necessary notions to understand the nature of self-similarity in network traffic. In this section, we analyze the long-term properties of the traffic. By long-term, we mean timescales of minutes up to days.

### 1.3.1 Traffic Statistics

Many researchers have chosen to study the behavior of network traffic by relying on packet level traces at a single link (see [107, 131, 115] for instance). Such traces allow the analysis of the traffic at the packet scale, but require a large storage space. For this reason, most of the used traces either correspond to a low traffic volume or a short period of time. Here, we have taken a different approach. In order to better understand the long-term behavior of the traffic, we consider a less precise trace that spans several physical links over six complete days. For this, we rely on a `Netflow` [45] trace collected at the border routers of the Belgian research Internet service provider BELNET in December 1999. The trace covers all the interdomain traffic received by the BELNET on all its external links and corresponds to 2.1 TBytes of traffic. BELNET provides access to the Internet as well as to high speed European research networks to universities, government and research institutions in Belgium. At that time, its national network was based on a 34 Mbps backbone linking major Belgian universities. Its users are mainly researchers or students with high speed connections (at least Ethernet) to the 34 Mbps backbone, although some institutions also provide dial-up service to their users. It was also at that time the Internet service provider with the largest external capacity in Belgium.

The BELNET network is connected to a few tens of peers with high bandwidth links. It maintains high bandwidth peerings with two transit providers, with the Dutch SURFNET network and was connected to the TEN-155 European research network, without providing any transit service to its peers. In addition, BELNET is present with high bandwidth links at the Belgian (BNIX) and Dutch (AMS-IX) national interconnection points with a total of about 40 peering agreements in operation. The `Netflow` trace we used aggregated the information from all upstream links, so that we have the incoming traffic information as if it was received through a single external link.

`Netflow` provides the aggregated information of the layer-4 flows, by recording the starting time, the ending time and the total volume in bytes for each unidirectional TCP and UDP flow. `Netflow` raises some issues concerning the layer-4 flow-level measurements, because it uses a timeout policy for flushing out its flow cache. Depending on the configuration of the router, some flows could be considered as terminated while this need not be true from the TCP connection viewpoint. Fortunately, this problem mainly concerns fine-grained measurements for flow arrivals which we do not consider here. Furthermore, we use a coarse time scale and only study the total traffic for aggregated traffic sources (not layer-4). Hence, the bias from the flow timeout policy should not interfere too much with our results. The finest notion of a flow we use here is the traffic evolution every minute for some external IP address. Many layer-4 flows may thus be aggregated into a single flow.

The utilization of `Netflow` forces us to approximate the layer-4 flows as equivalent to fluid flows. More precisely, a flow transmitting $M$ bytes between $T_{start}$ and $T_{stop}$ is modeled as a fluid flow transmitting $M/(T_{stop} - T_{start})$ bytes every second between $T_{start}$ and $T_{stop}$. This approximation leads to an inaccurate estimation of the short-term burstiness of the traffic but allows for longer traffic traces collection. All traffic volume information is summarized over equally-spaced one minute intervals throughout the six days. The time granularity of the trace is consequently of one minute.

The part of the `Netflow` statistics on which we rely is the information of the total traffic volume received from each external IP address for every minute of the measurements. For each one minute interval of the measurements, we have the number of IP addresses that are sending traffic during that particular minute as well as the corresponding traffic volume for each of them. This choice of one minute for the time granularity was driven by our intuition that these timescales are sufficiently coarse not to be too much biased by the traffic dynamics due to TCP flows, but rather IP-level characteristics. The actual time granularity of the original trace was one second.

Even though the time granularity of the trace is one minute, it accounts for more than 42 million values recorded over more than 8600 samples. On average, 4912 external IP addresses were sending traffic each minute. This trace hence provides a fine measurement of the traffic dynamics for time-scales spanning minutes to hours.

### 1.3.2 Total Traffic Self-similarity

In this section, we focus on detecting the statistical self-similarity of the total traffic time-series. We look at the total traffic received at the incoming access links for every one-minute time interval during the 6 days of the measurements. Figure 1.5 shows the evolution of the total traffic during the period of the measurements. While the global evolution of total traffic exhibits a stable daily periodicity with peak hours located during the day, there are important deviations around the average traffic evolution throughout the day that give self-similar traffic this highly bursty look. The mean traffic over the six days was about 32 Mbps, with a one-minute maximum peak at 126 Mbps and a standard deviation of 21 Mbps. The trace begins around 1 AM and finishes six days later around 1 AM also.

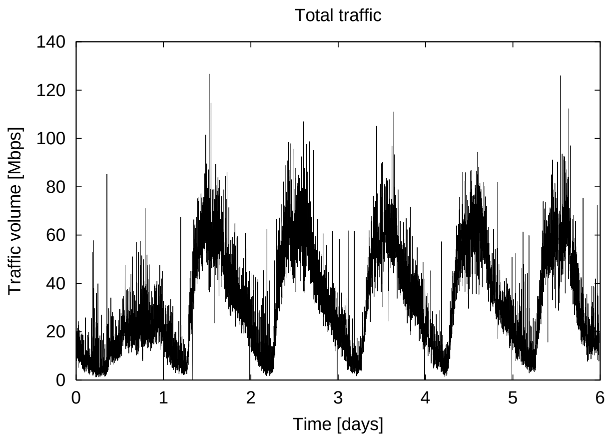



Figure 1.5: Total traffic evolution.

Figure 1.6 presents the estimation of the Hurst parameter using the four estimators introduced in section 1.2.1. The R/S (top left of Figure 1.6) and the aggregated variance (top right of Figure 1.6) estimators indicate a value of $H$ close to 1. The correlogram (bottom left of Figure 1.6) for its part indicates a large correlation for timescales up to one hour. Finally, the periodogram shows that a bias from non-stationarity occurs with a highly negative slope at low frequencies characteristic of a non-stationary $1/f$ noise [111].

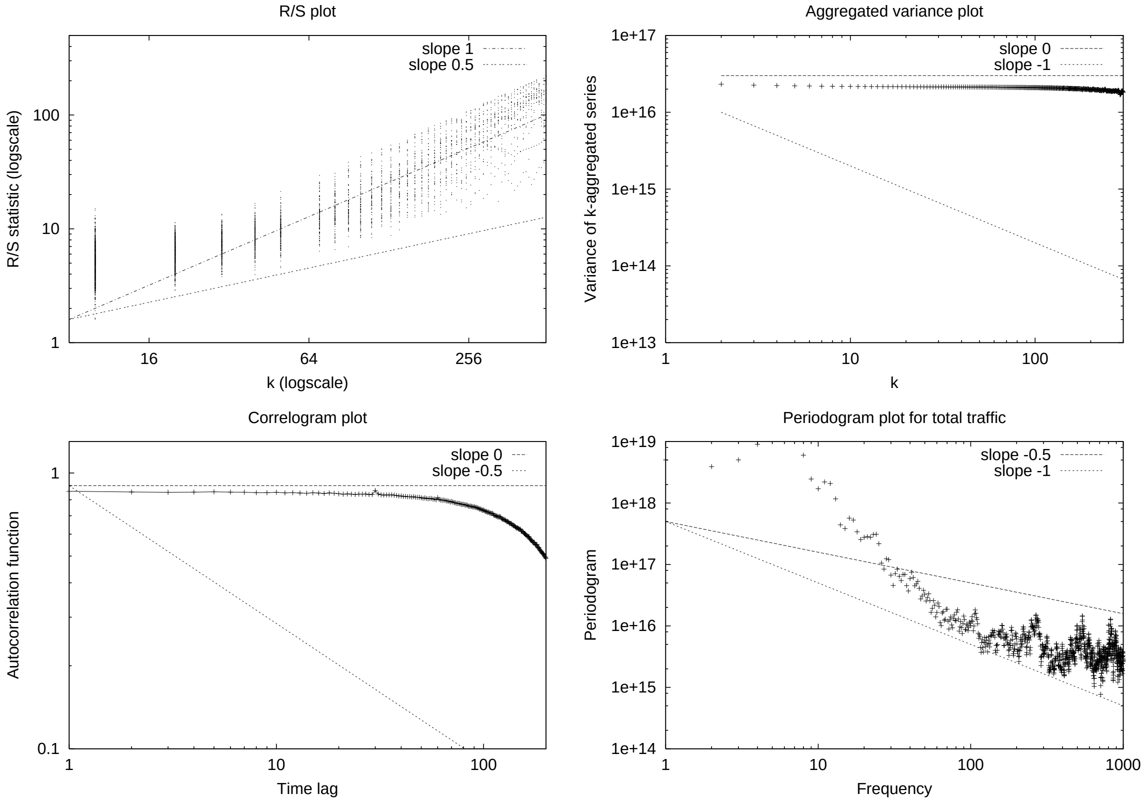



Figure 1.6: Estimation of $H$ for total traffic.

Most of the estimators for $H$ are defined in an asymptotic way. They provide a good estimate for large datasets only. However, our time-series containing more than 8600 samples, we can be quite confident with the previous estimators at least in the value range they provided, with $H$ closer to 1 than 1/2 for all of them. Non-stationarity in the signal might however bias these estimators. To confirm the presence of scaling while minimizing the bias from non-stationarity, Figure 1.7 provides the *logscale diagram* for the total traffic. Figure 1.7 shows the *logscale diagram* of the total traffic trace together with the 0.95 Gaussian confidence intervals. The *logscale diagram* shows that only octaves 5 to 9, timescales ranging from 30 minutes to 8 hours, contain scaling with a measured $H$ of 0.84. The variance of the estimation of the value of $H$ is 0.04. This variance of the estimation of $H$ depends partly on the data length. The short timescales ($1 \leq j \leq 5$, up to 15 minutes) show a non-correlated signal with a value of $H$ close to 1/2, indicating that short-term variations can be considered as "random" on such a coarse-grained trace. Large timescales ($j \geq 10$, half a day and more) show a large standard deviation of the estimator $y_j$ partly caused by the too short length of the data.

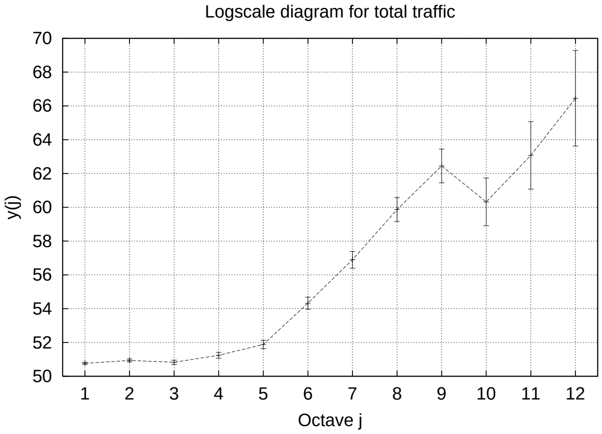



Figure 1.7: Logscale Diagram for total traffic.

## 1.4 Self-similarity in TCP flow arrivals

The previous section showed that the total traffic seen on access links contained scaling for timescales between minutes and up to one hour. It showed that scaling was present at the interdomain level, but not why. Scaling in the total traffic can have several roots. Three main roads have been explored concerning the explanation of the presence of scaling in network traffic.

The first concerns distributional properties of the flow activity periods that were shown to be heavy-tailed [51, 131]. The complementary proof of Taqqu, Willinger and Sherman [163] then provided a formal justification for the presence of self-similarity through the superposition of a large number of independent ON/OFF sources with heavy-tailed ON and/or OFF periods. Park, Kim and Crovella [127] made the connection between distributional properties of the file sizes and the modulating effect of the TCP/IP stack and showed that heavy-tails in the applicative flows were mapped to heavy-tailed activity periods at the network layer.

The second road was investigated by Feldmann, Gilbert and Willinger in [72] that studied the short time-scales behavior of the TCP protocol to show that multiplicative processes described well its working, adding multifractals and cascades among the scaling behaviors present in network traffic. These two roads, heavy-tails and the TCP dynamics, concern traffic volume. The first is related to the size of the files to be carried over the network while the second concerns the way flows are broken by TCP to be sent as IP packets. While the previous two aspects tell that the information to be exchanged as well as the behavior of the network and its protocol can be considered as partly responsible for scaling in the traffic, one may wonder whether users of the network have anything to do with scaling in the traffic, allowing us to introduce the third road.

The third road must be traced back in 1995, when Paxson and Floyd [131] found that the connection arrival process of some applications failed to be Poisson. This finding was unexpected due to the belief that users initiating actions independently should lead to a Poisson process. With the growth of the web, the situation was due to change even more, because many flows are initiated for every user session. The first step to understand this aspect was [73] that explained self-similarity in the TCP connections arrival process by heavy-tails in the number of TCP connections within user sessions. Another remarkable effort in the modeling of the TCP connection arrivals is Feldmann in [70] that showed that TCP connections are self-similar, through a second-order analysis. The aim of [70] was not to study the time-scale dependent properties of this process, but rather to fit a probability distribution to model TCP connection arrivals. The first thorough study devoted only to the scaling properties of the TCP flows arrivals was [177] that showed that this process is by no means simply self-similar. [177] showed that the TCP flow arrivals is actually a non-stationary process with different scaling components at different timescales. [177] showed that there exists a non-stationary statistical correlation at the seconds to minutes timescales, stable self-similarity at the minutes to hours timescales and that the "time of the day" trend occurs for timescales larger than about half an hour. Due to second-order properties analysis, [177] was not able reveal the exact nature of the components at each timescale. The goal of the present section is to provide a detailed study of the fine scaling properties of the process of the flow arrivals. Our aim being to determine the nature of the components of this process, for timescales ranging between less than seconds to days.

### 1.4.1 TCP Traffic Traces

In this section, we describe the traffic traces on which we rely to analyze the behavior of the TCP flow arrivals.

#### LBL-CONN-7

The "lbl-conn-7" trace, available on the Internet Traffic Archive [3] contains thirty days' worth of all wide-area TCP connections ($606,497$ wide-area connections) between the Lawrence Berkeley Laboratory (LBL) and the rest of the world. It was gathered with `tcpdump`[2] on a Sun Sparcstation using the BPF kernel packet filter between September 16 1993 through October 15 1993. The time precision of the trace is one microsecond but we used a one hundred milliseconds precision, by not having taken into account the digits corresponding to these smaller timescales of the timestamps. Only the TCP connections that ended normally (with both SYN and FIN) were used. We call this trace the "LBL" trace. More details about this trace can be found in [130, 131, 3].

#### UC Berkeley Home IP Web

The "UC Berkeley Home IP Web" trace is also available from the Internet Traffic Archive [3]. This dataset consists of 18 days of `http` traces gathered from the Home IP service offered by UC Berkeley to its students, faculty, and staff. All clients were

[2]ftp://ftp.ee.lbl.gov/tcpdump.tar.Z

using modems. This client trace was collected with a packet sniffing machine placed at the head-end of the Home IP modem bank. Only traffic destined for port 80 was traced, all non-*http* protocols and *http* connections for other ports were excluded from this trace. It contains $9,244,728$ flows spanning the period from November 1st 1996 through November 19th 1996. Although a better precision was available, we used a one hundred milliseconds time precision for the timestamps by removing further digits in the timestamp. More information on this trace can be found on the ITA website [85]. We refer to this trace as the "UCB" trace in the remainder of this chapter.

### AUCK IV

The Auckland-IV data set [124] is a continuous 45 days GPS-synchronized IP header trace captured with a DAG3 system [123] at the University of Auckland Internet uplink by the WAND research group between February and April 2001. The University of Auckland ITSS department is operating an OC3 ATM link via Clear Communications, which is carrying a variety of services off the main campus. This trace contains 6 1/2 weeks of GPS-synchronized IP headers taken with a pair of DAG3 cards at the University of Auckland Internet access link. The tap was installed at an OC3 link carrying a number of Classical-IP-over-ATM, LANE and POTS services. The trace contains all Classical-IP headers of a pair of redundant VPI/VCI's, which connects the university to their local service provider. We had a different trace for both directions, incoming and outgoing. We selected a contiguous 7-days sample in this trace for our analysis. There were 11,919,853 incoming flows (12,335,605 outgoing) during this sample 7-days period. In this trace, we solely rely on packets that had the SYN bit set and relied on a ten milliseconds precision mainly due to trace size limitations. We refer to this trace as the "AUCK" trace in the remainder of this chapter.

### Yucom

The Yucom trace consists of all flows between modem clients of a small commercial Belgian Internet service provider, Yucom. Yucom provides Internet access to dialup users through regular modem pools. The trace spans less than five days between April 17th and April 21st 2001. It is a Netflow trace [45], not a packet-level one. The collecting machine was located within the premises of the service provider and was running OSU *flow-tools* [141]. Due to the flow timeout policy used by Netflow, we did not rely on the raw flow information because it over-estimates the number of flows. Instead we considered the incoming flows that had the SYN flag set, hence TCP traffic only. This means that even if we do not have the correct TCP flows information because of the Netflow timeout, we have the correct information concerning the new TCP flows (SYN). During the whole period of the trace, $59,581,814$ incoming TCP flows that had the SYN flag set were recorded. Among the 574 GBytes of IP packets in the incoming TCP flows, `http` and `https` represent about 72 percent of the traffic volume and 77 percent of the flows. Port numbers corresponding to "chat" applications (mainly 6667) represent about 16 percent of the traffic volume and 8 percent of the flows. Finally, e-mail applications had about 5 percent of the traffic volume and 6 percent of the flows. We call this trace the "Yucom" trace in the remainder of this section. We used a one hundred milliseconds time precision for this trace and only study its incoming flows.

#### PSC

The PSC trace consists of all outgoing flows seen on two monitoring points internal to the Pittsburgh Supercomputing Center, an American commodity and Internet2 network provider to organizations in western Pennsylvania. The flows were collected continuously over 24 hours starting from March 12$^{th}$ 2002 16:40 GMT at both monitoring points [177]. These points covered all outbound commodity traffic from non-dormitory hosts at a large university, in addition to all outbound commodity traffic at several smaller organizations. The collection was made using the CoralReef package from CAIDA [98] using typical Netflow settings for flow expiration. During the flow period the network served by one monitoring point experienced considerable congestion, so that rate limits during peak usage times were enforced. We consider the TCP traffic only, but because we had no information about the TCP flags all terminated flows (TCP FIN or CoralReef timeout) were treated as complete flows. We used a ten milliseconds time precision for this trace. There were $39,786,219$ TCP flows in the trace, representing 574 GBytes of IP traffic. The important applications we identified were `Gnutella` with about twenty-two percent of the bytes and about twelve percent of the flows, `kazaa` with about twenty percent of the bytes and eight percent of the flows, and `http` with about ten percent of the bytes and sixty percent of the flows. A large fraction of the bytes were thus generated by P2P applications. We call this trace "PSC" in the remainder of this chapter.

### 1.4.2 TCP flow arrivals

This section studies the process of the flow arrivals defined as the number of arriving flows seen for every time interval, for the five traces. We first study the second-order properties of this process, globally over the traces as well as in a time-dependent way. After, we deal with the higher-order properties. Finally, we study the behavior of the largest clients of the PSC trace to understand the difference found between the traces. Each analysis is preceded by a presentation of the estimators on which we rely in the analysis.

In this section we rely solely on wavelet-based scaling estimators, hence reading the introductory section on wavelet analysis (section 1.2.3) is highly recommended. Because of the different time precision of the traces, the same octaves on different graphs may represent different timescales. The Auckland trace and the PSC trace use a ten milliseconds precision, hence octave one corresponds to a time unit of of twenty milliseconds. For all other traces, we use a one hundred milliseconds time precision so that octave one represents two hundred milliseconds. Recall that each subsequent octave reduces the amount of data by two, corresponding to a timescale two times larger than the previous octave. Having relied on a common timescale for all traces would have prevented us to show some interesting behaviors, so we preferred working with different time granularities for the traces although it certainly complexifies the task of the reader.

### Second-order scaling

The second-order wavelet-based estimator of scaling, the *logscale diagram* (LD) [5] consists in plotting $y_j = log_2(\mu_j)$ against $j$ together with the confidence intervals, where

$$\mu_j = \frac{1}{n_j} \sum_{k=1}^{n_j} |d_X(j,k)|^2. \tag{1.11}$$

$n_j$ in Equation 1.10 denotes the number of wavelet coefficients at octave $j$ and $d_X(j,.)$ represent the wavelet coefficients at octave $j$. The *logscale diagram* is a second-order statistic of the wavelet coefficients that measures the variance of the wavelet coefficients at each octave $j$. Scaling detection can be performed through observation of linear alignment of the confidence intervals of the $y_j$ within some octave range. If, and only if alignment is detected, can the estimation of scaling parameters be performed [5].

Because of the importance of non-stationarity for understanding the flow arrivals, we rely on a method introduced in [167, 177] that consists in computing the LD over fixed-length time intervals. In this method, one plots on a three dimensional graph the evolution in time of the LDs computed over fixed size time intervals, hence its name "3D-LD". Correctly interpreting the 3D-LD requires some care. The first property of the analyzed trace seen on the 3D-LD is the evolution in time of the general level of each LD. Since the LD measures the variance of the wavelet coefficients of the signal, the value of the LD for octave one provides an indication of the total energy contained in the signal. The higher the value of the LD at octave one, the higher the variations in the signal. A second-order stationary signal should thus have a 3D-LD whose first octave does not change with time. Non-stationarity in the variance can thus be detected by studying the time variations in the first octave of the successive LDs. Because of the vanishing moments property of the wavelet, non-stationarity in the mean cannot be detected since trends matched by a polynomial of order up to $N-1$, with $N$ is the number of vanishing moments of the wavelet, are automatically removed. Nevertheless, non-stationarity in the mean can still be indirectly detected in the 3D-LD with the value of the last octave that represents the energy contained in the largest timescales of the analyzed block. However, the value of the LD for the last octave represents the energy of the signal that could not be matched by the wavelet analysis for smaller timescales, so it does not always represent an accurate picture of the variations over time of the whole block analyzed by the LD, but depends on the analyzing wavelet used. Depending on the type of non-stationarity in the mean that the signal contains, different wavelet functions will match such non-stationarities differently. Non-stationarity in the variance of the wavelet coefficients will thus appear in the 3D-LD through shifts in the level of the surface that follow the non-stationarity of the signal over timescales larger than the size of a block over which any LD is computed.

The main information about the variability of the process provided by the 3D-LD concerns how the variability of the process is decomposed among the different timescales. As already discussed in section 1.2.3, the slope of the LD over some octave range gives qualitative information about the signal. A slope close to zero points to an uncorrelated signal. A positive slope points to LRD or self-similarity. A negative slope means that the process is short-range dependent or self-similar with $H < 1/2$. The interest of the 3D-LD is to help visualize the changes of the LD in time. It has been shown in [167, 177] that whenever several components exist at different timescales and whose behavior change with time, the LD by itself might be largely biased. If the non-stationarity in the signal is important or the different components of the signal at the different timescales complex, then the LD might point to a process whose scaling properties are far from those of the real process. Then the 3D-LD is a valuable tool, to validate and eventually correct the information provided by the LD over the whole analyzed sequence. Nevertheless, the nature of scaling cannot be automatically inferred simply from the 3D-LD, it is a graphical method that helps to better understand the time-varying nature of the LD. The nature of scaling has to be found in the physical understanding of the process or its high-order properties.

Note that to reduce the (already important) verbosity of the text we shall frequently use the following two acronyms in the remainder of this section, “LD” for the logscale diagram and “3D-LD” for the three-dimensional version of the logscale diagram.

### Second-order analysis

In this section, we study the second-order scaling properties of the flow arrivals as defined in section 1.4.1. We study their global scaling properties with the LD as well as their time-dependent second-order scaling properties with the 3D-LD.

Let us first analyze the LD of the flow arrivals. The interest of the LD is at identifying the presence of strict scaling over a range of timescales in the process taken as a whole. If a straight line of positive slope can be fitted to a portion of the LD, then scaling (be it self-similarity, LRD or even statistical dependence) has been detected.

Figure 1.8 shows the logscale diagram for all traces. As it appears on the graphs, no strict scaling can be found on any trace, but only some limited octave ranges that could be fitted a straight line with respect to confidence intervals. Based on these figures only, it is difficult to say much about except that there are roughly three different regions on each of the graphs. A property of the logscale diagram is that, in theory, it is possible to infer the type of process present within some octave range based on the value of the slope of the logscale diagram [5]. Here we can say that timescales below seconds (octaves $< 8$ for Auckland and PSC, octaves $< 5$ for UCB, Yucom and LBL) have quite small correlations (or uncorrelated for Auckland outgoing, UCB and LBL), timescales between seconds and minutes have statistical dependence or LRD (slope $< 1$) while the timescales between minutes and hours could be self-similar (slope $> 1$). The problem for the largest timescales is that the confidence intervals tend to be so large that inference is hopeless, but we have studied these large timescales in [177] and found that the "time of the day" behavior begins at about more than half an hour.

With the knowledge of the flows arrivals process whose second-order properties were shown to be non-stationary in [177], one should not trust the LD described above because it does not show how scaling evolves within the time span of the studied time-series. Henceforth, we now turn to the 3D-LD of the flow arrivals, which still is a second-order statistic, but with a reduced bias against second-order non-stationarity. Interpreting the 3D-LD relies on the correct interpretation of the LD, explained in section 1.2.3. Figure 1.9 shows the evolution of the LD computed for time blocks of a little less than three hours for Auckland, one hundred and ten minutes for UCB, three and a half hours for Yucom, forty-five minutes for PSC and twenty-nine hours for LBL. First, the "time of the day" pattern is apparent from the largest octaves of the graph. Second, all graphs provide a 3D-LD that has roughly the same look as the LD although the "time of the day" non-stationarity is apparent for Auckland and Yucom as well as the "day of the week" pattern for LBL. This daily pattern constitutes the first source of non-stationarity. The second type of non-stationarity appears for the incoming flows of Auckland (top right of Figure 1.9), where days two and three exhibit sharp peaks explaining the larger values of the LD for the Auckland incoming flows over octaves [8,15] (top right of Figure 1.8). This shows that a single short-term event in the traffic might bias the LD in a significant way, simply because it measures the variance of the wavelet coefficients for a given timescale as a whole. Finally, the PSC trace is known to contain the effect of rate limitation for the largest client. Its effect is quite evident (bottom left of Figure 1.9) with a constant positive slope for octaves [1,8]. One can see that this component hides the surface that would be there if no rate limitation had been performed. Any sort of shaping or policing that tends to spread the flows and induce some statistical dependence among the flow arrivals would have such an effect, for timescales of up to a few seconds here.

Because the logscale diagram only deals with second-order properties, it is difficult to infer the qualitative nature of the process at each timescale. While this section has shown that we should expect to see at least three components in the traffic at sub second, second to minutes and higher than minutes timescales, we cannot precisely determine what type of scaling occurs at which timescale: correlation, LRD, self-similarity ? The purpose of the next section is to try to answer this question.

### High-order scaling*

The wavelet-based partition function is defined as

$$S(q,j) = \sum_k |2^{-j/2}\, d_X(j,k)|^q \tag{1.12}$$

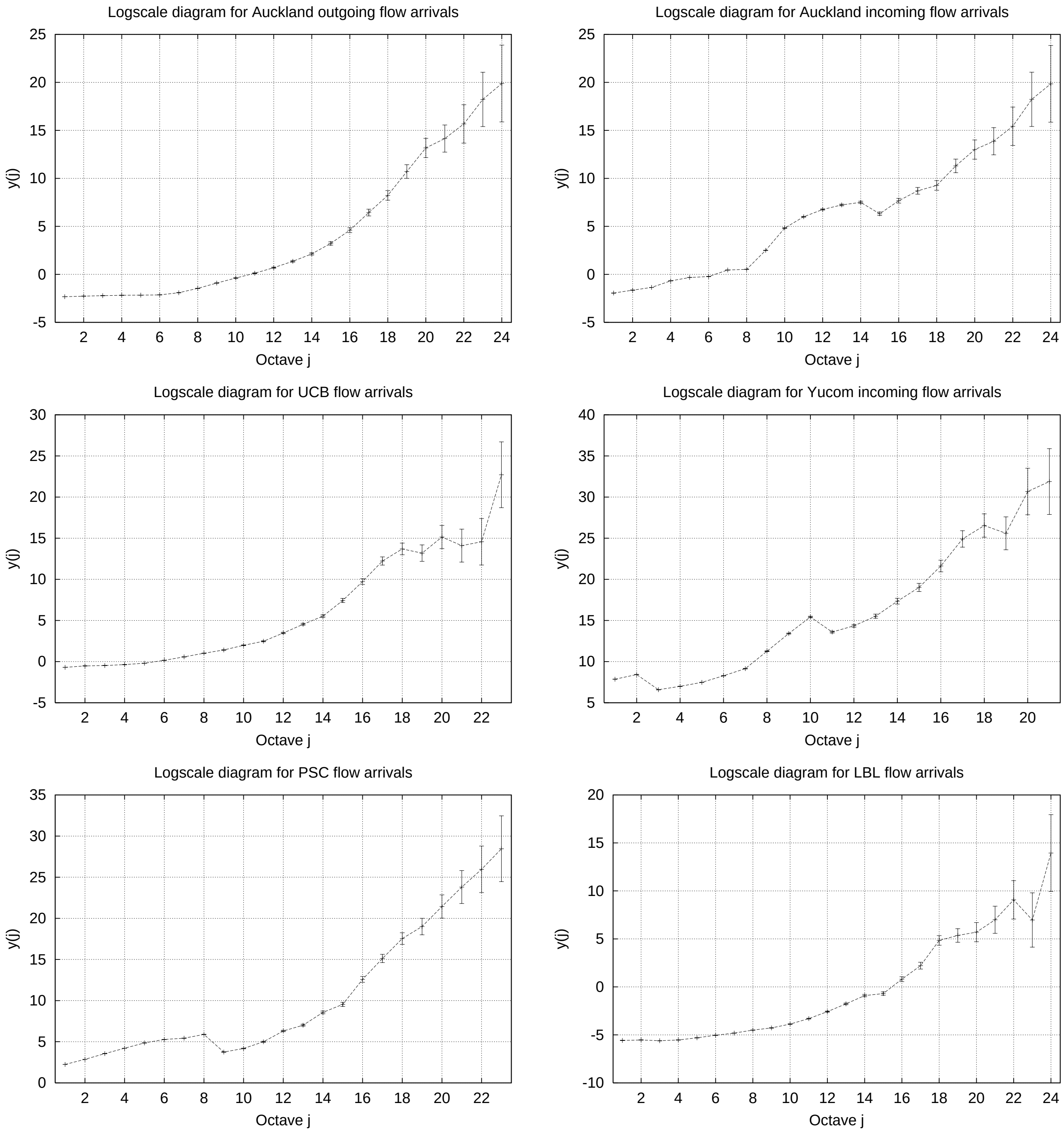



Figure 1.8: Logscale diagrams.

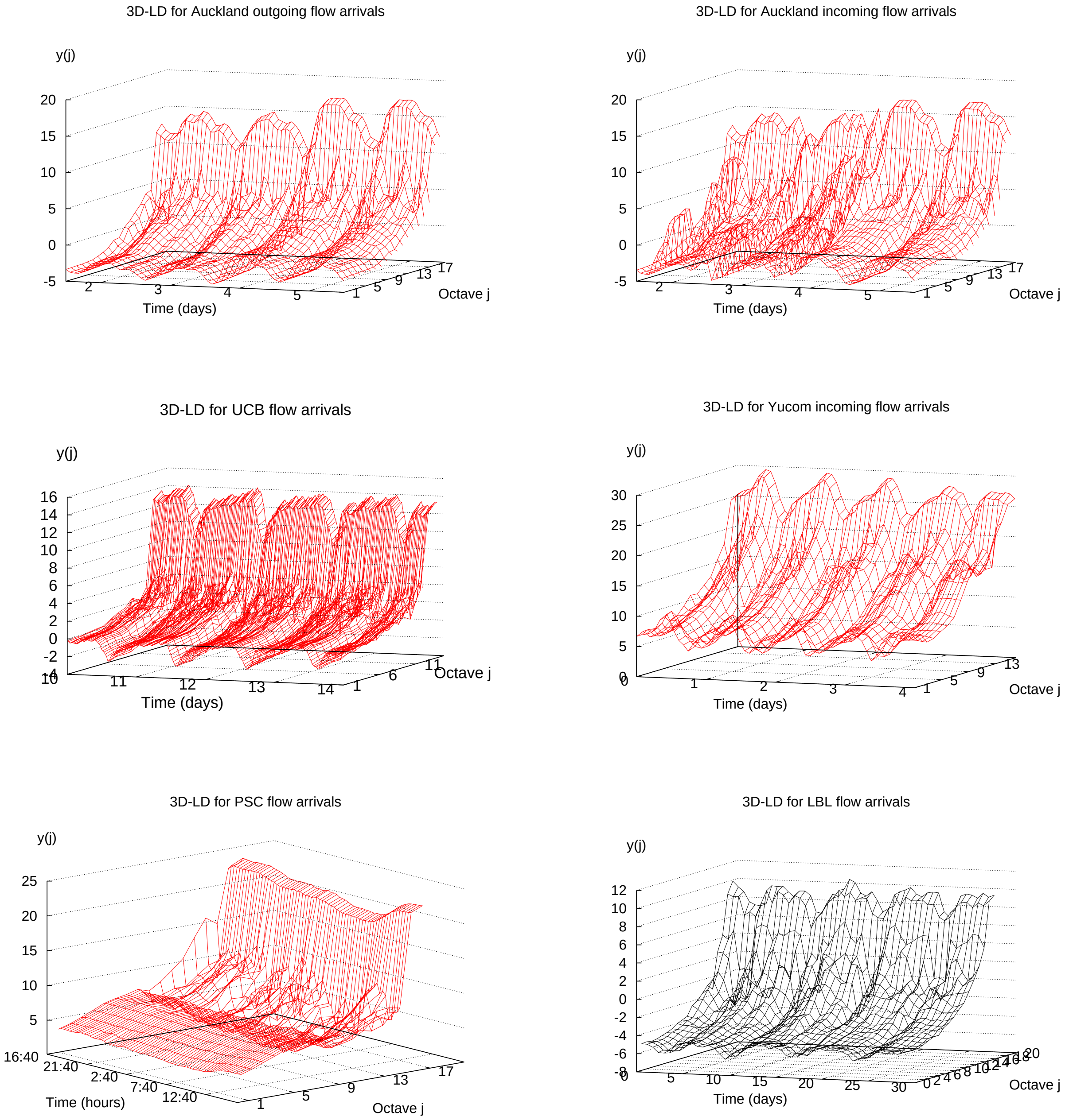



Figure 1.9: 3D-logscale diagrams.

where $2^{-j/2}$ is a normalizing factor to remove the $L_2$ normalization of the wavelet. The partition function captures the local scaling behavior of a signal because raising the wavelet coefficients to an exponent magnifies the importance of the largest coefficients that arise due to a local irregularity, while it reduces the importance of small coefficients. Recall that a wavelet is an oscillating function, hence the value of the wavelet coefficient is proportional to the size of the variation in the signal that is matched by the inner product with the wavelet. The smaller the wavelet coefficients for some scale $j$, the smaller the value of the partition function for $S(q,j)$ for large $q$. This permits to study the importance of the local irregularities at some scale $j$. A relatively smooth process that has no particularly large local irregularities will have small values of the $S(q,j)$ for $q > 2$, we call such a process a "second-order" process in this chapter. An irregular process on the other hand will tend to have larger values of the $S(q,j)$ for large values of $q$, we call such a process a "high-order" process in this chapter. Note that we only study the positive moments because we are only interested in the irregularity of a process, not its smoothness.

*Multiscaling* is a relaxation of strict scaling where the linearity relation among moments does not hold anymore but can vary for each moment $q$ [152, 179]:

- self-similarity : $E|d(j,k)|^q \propto exp(qH \ \ln(2^j))$,
- multiscaling : $E|d(j,k)|^q \propto exp(H(q) \ \ln(2^j))$.

Strict self-similarity implies that all moments behave like a power-law of the timescale, where the power-law is linear with respect to $H$. Multiscaling on the other hand allows the power-law relationship among moments to depend on the moment $q$. Self-similarity imposes that a single parameter $H$ controls the behavior of the whole dynamics of the process. Multiscaling on the other hand implies a non-linear behavior of the moments with respect to one another. An example of multiscaling is multifractality [95, 140] where there exists a whole spectrum of local scaling exponents related to the local (ir)regularity of the sample path of the process. Let $\alpha(t_0)$ denote the local scaling exponent of the process at time $t_0$, then a self-similar process has $\alpha(t_0) = H$ while a multifractal process has a non-constant scaling exponent $\alpha(t_0) = H(t_0)$. This exponent $\alpha(t_0)$ is related to the local geometrical properties of the function, with large values of $(\alpha(t_0) > 1)$ mean small local variations while $\alpha(t_0) < 1$ arises when the local irregularities are large. Multifractality occurs when $H(q)$ depends non-linearly on $q$ but linearly with respect to $j$: each moment $S(q,j)$ then tends to diverge relative to one another for the smallest timescales.

Besides power-laws of the moments **in time**, it is also possible to study the power-law relationship among moments ($q$) at a given timescale. The partition function provides a means to determine the type of process that arises at each timescale through the multiscaling paradigm. Multiscaling allows for three possible behaviors related to the linearity of high-order moments relative to low-order ones :

1. Sub-linearity of a LRD, statistically dependent or non-scaling process implies that $M(j) < 1$,
2. Linearity of a self-similar process implies that $M(j) \sim 1$,

3. Sup-linearity of a multiscaling process implies that $M(j) > 1$,

with

$$M(j) = \left\langle \left| \frac{log(S(q+1,j)) - log(S(q,j))}{log(S(2,j)) - log(S(1,j))} \right| \right\rangle_q, \tag{1.13}$$

where $\langle\rangle_q$ represents the averaging over the values of $q$. Care must be taken when arguing about "the nature of the process" for as with the logscale diagram, what one sees is often a worst case behavior. The reason is that summing or averaging leads to a potential bias of large values occurring in some part of the data. For example, the LD is largely biased against second- and higher-order non-stationarity. The partition function presents the same defect so that multiscaling in the process could hide self-similarity or LRD if the value of the irregularities of this component is sufficiently large compared to the other components of the process at a given timescale. Hence, care must be taken not to consider too seriously what the $M(j)$ says without carrying a time-dependent analysis in order to check whether non-stationarity in the process does not bias the estimation. Instead of the $M(j)$, we shall rather use $log(M(j))$ that maps the three types of scaling processes, LRD (or statistical dependence), self-similarity and multiscaling respectively to the negative, zero and positive values of $log(M(j))$. Another important thing to be noticed is that this scaling among moments cannot be used without considering a sufficiently large number of timescales in order to the qualitative analysis to be meaningful. It is not possible to analyze one single timescale through the $M(j)$ because scaling still has to be present in the low order properties for the high-order analysis to make sense. The multiscaling paradigm is useful as a complement to second-order analysis, not as a tool by itself. Again, because of the averaging process the value 0 should not be taken too strictly.

#### High-order analysis*

The counterpart of the LD for high-order properties is the partition function described in section 1.4.2. The additional information it provides about a signal concerns the degree of homogeneity in time of the wavelet coefficients at each timescale. While the LD informs one about the variance of the irregularities in the process, the partition function tells about the distribution of large and small wavelet coefficients in the signal for each timescale. For this, we plot $log(S(q,j))$ as a function of the scale $j$, for a range of values of the moments $q$. If the low order moments ($q \leq 2$) have a larger $log(S(q,j))$ than high order moments, we conclude that the process is a second-order one, well described by the low order properties of the wavelet coefficients (the variance of its increments). This process in turn must be relatively smooth, without too many large-valued coefficients. Figure 1.10 presents the partition function for all traces.

Two important things must be looked at when interpreting the partition function. First, discriminating between second-order and high-order processes can be done by determining for which values of $q$ the $log(S(q,j))$ is the largest. A second-order process will have larger $log(S(q,j))$ for $q \leq 2$ while a high-order process for $q > 2$.

Figure 1.10 shows that Auckland outgoing flows, UCB and LBL are second-order processes for the timescales larger than a few seconds because of their larger values of the partition function for $q \leq 2$. The Auckland incoming flows on the other hand are a second-order process for timescales larger than 5 minutes only. All other traces are high-order processes for all the considered timescales with their larger value of the partition function for large values of $q$. This implies that most of the traces contain large wavelet coefficients at most of the considered timescales.

A recurrent problem with wavelet analysis, as well as all the analysis techniques known so far, concerns non-stationarity. Wavelet estimators of scaling rely on some sufficiently large sample of a time series where the wavelet coefficients on the fixed dyadic grid are assumed to be stationary, at least for what concerns the studied properties. The wavelet coefficients then can be used to compute a meaningful statistic of some property of the process. The problem in practice is that one would like to infer properties without having the certainty that this stationarity assumption holds. A way to deal with this problem consists in computing the statistics over many subsamples of the time-series and compare the subsamples to determine what is a true property of the process and what is an "outlier".

Now we turn to the time-dependent partition function, to determine whether the high-order behavior identified in the partition function arises for limited periods of the process or is a true property of the process that lasts for its whole duration. Figure 1.11 shows the distribution of the partition function for subsets of the process for moments two and six only. Note that the size of these subsets is at least $2^{16}$ in order to rely on a sufficiently large sample for each subsequence. Figure 1.11 shows how frequently the high-order properties ($log(S(6,j))$) are larger than the low order ones ($log(S(2,j))$) on the constant-size subsamples. The Auckland outgoing plot (top left of Figure 1.11) reveals a single high-order subsample whose presence was not able to bias the partition function (top left of Figure 1.10). The Auckland incoming plot (top right of Figure 1.11) on the other hand exhibits a few subsamples having a high-order behavior for octaves [8,14]. This time these events were sufficiently strong to bias the partition function so as to consider these timescales as high-order while they are low-order most of the time. The same bias occurs for the PSC trace for octaves [9,14]. The other traces do not seem to contain such a pathological behavior.

Up to now, we have shown that the process of the flow arrivals can be second-order or high-order depending on the considered trace. Still remains to be answered the question of the "nature" of the process that arises at any given timescale, through the use of the $M(j)$ estimator defined in Formula 1.13. In the same way as for the partition function, we compute for subsamples of the whole traces the $M(j)$ for each $j$, and plot $log(M(j))$ as a function of $j$. Figure 1.12 presents the values of the $log(M(j))$ for each $j$ and each trace.

Negative values on some graph of Figure 1.12 indicate LRD, statistical dependence or no scaling at all depending on the results of the second-order analysis. A true LRD process being second-order stationary requires a positive slope of the LD for the largest timescales, a stationary 3D-LD, no high-order scaling and a negative $log(M(j))$. Statistical dependence is like LRD except that the positive slope of the LD stops before the largest timescales, at the upper cutoff value of the correlation

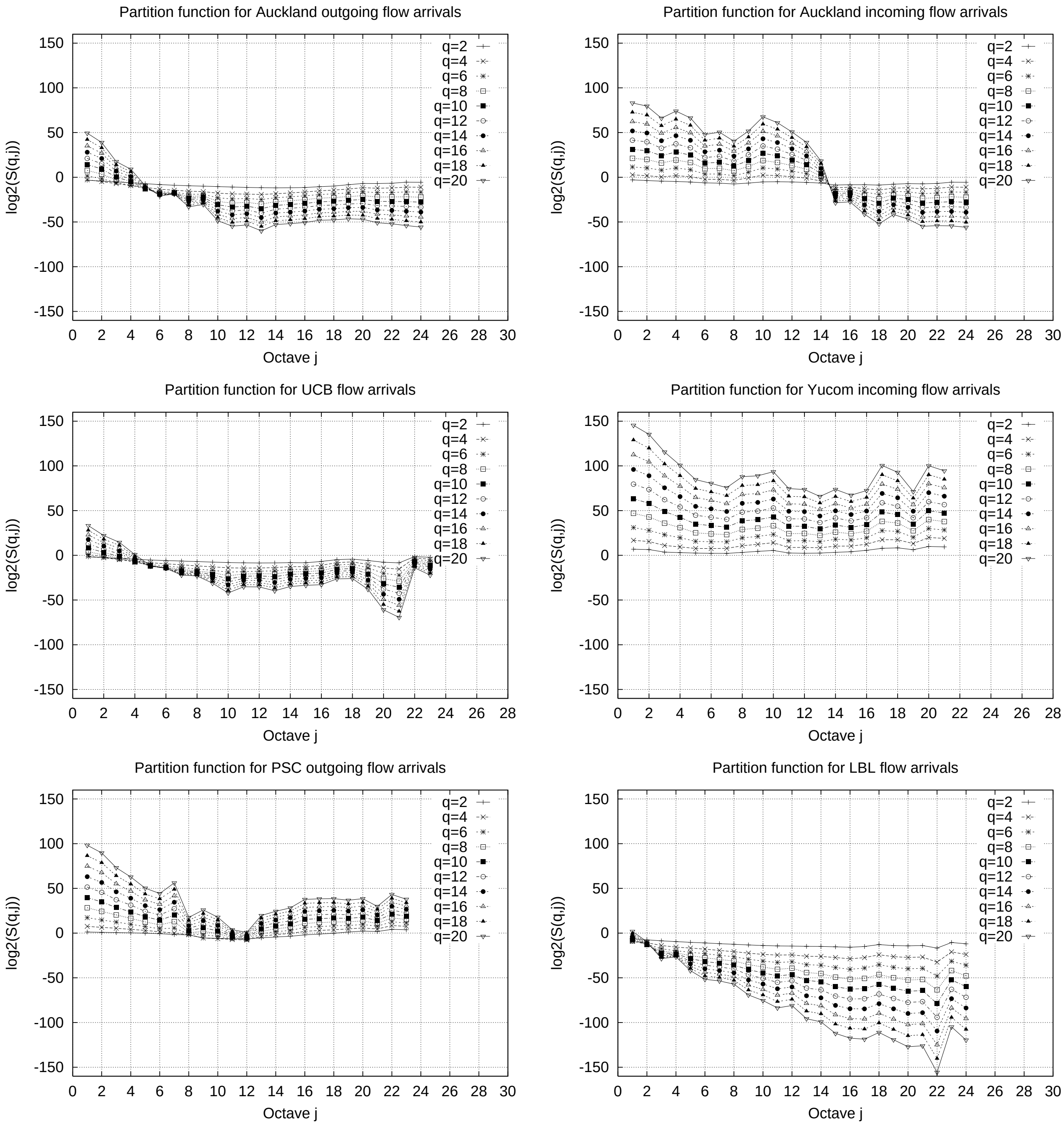



Figure 1.10: Partition functions.

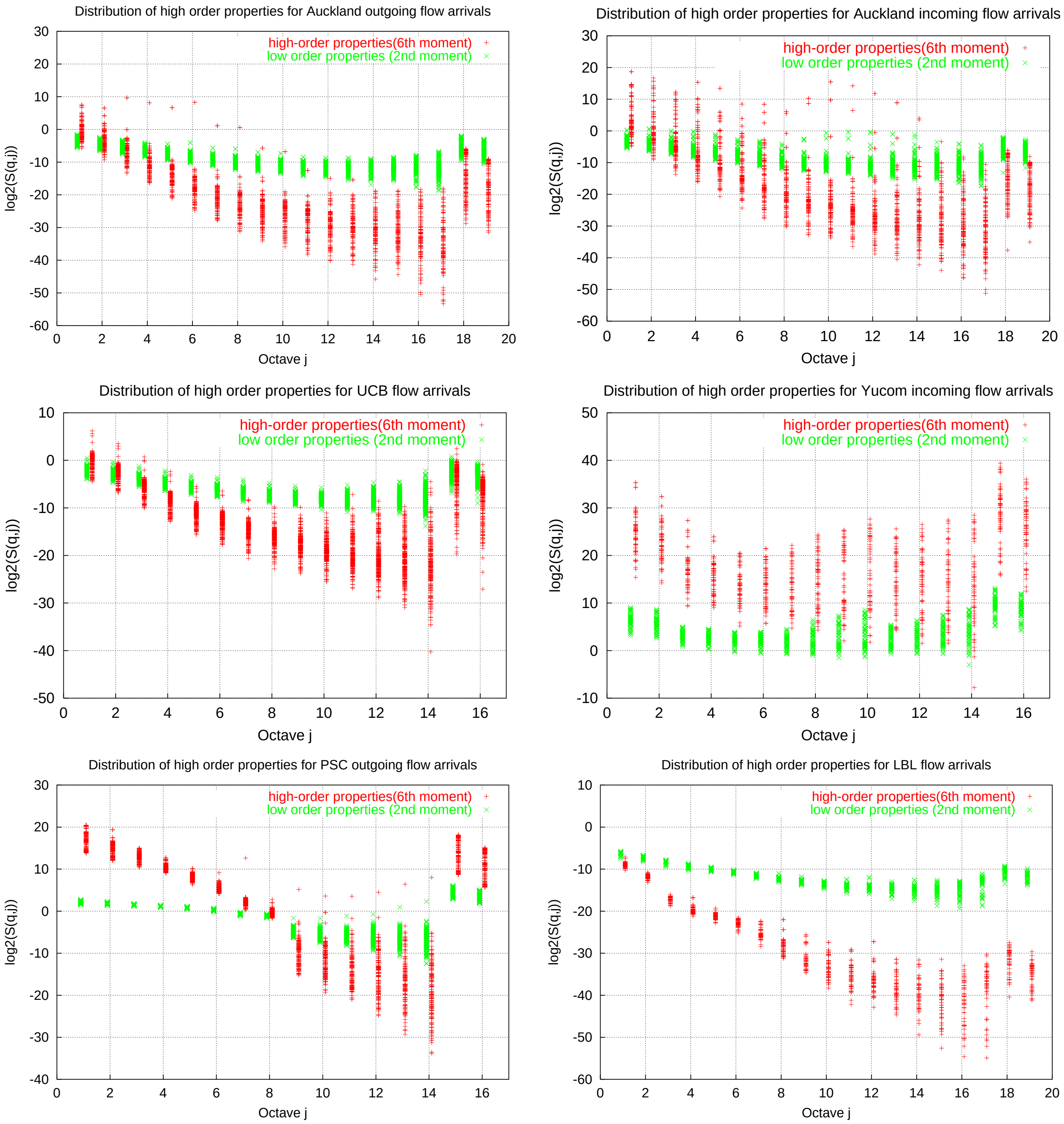



Figure 1.11: Stationarity check for partition function.

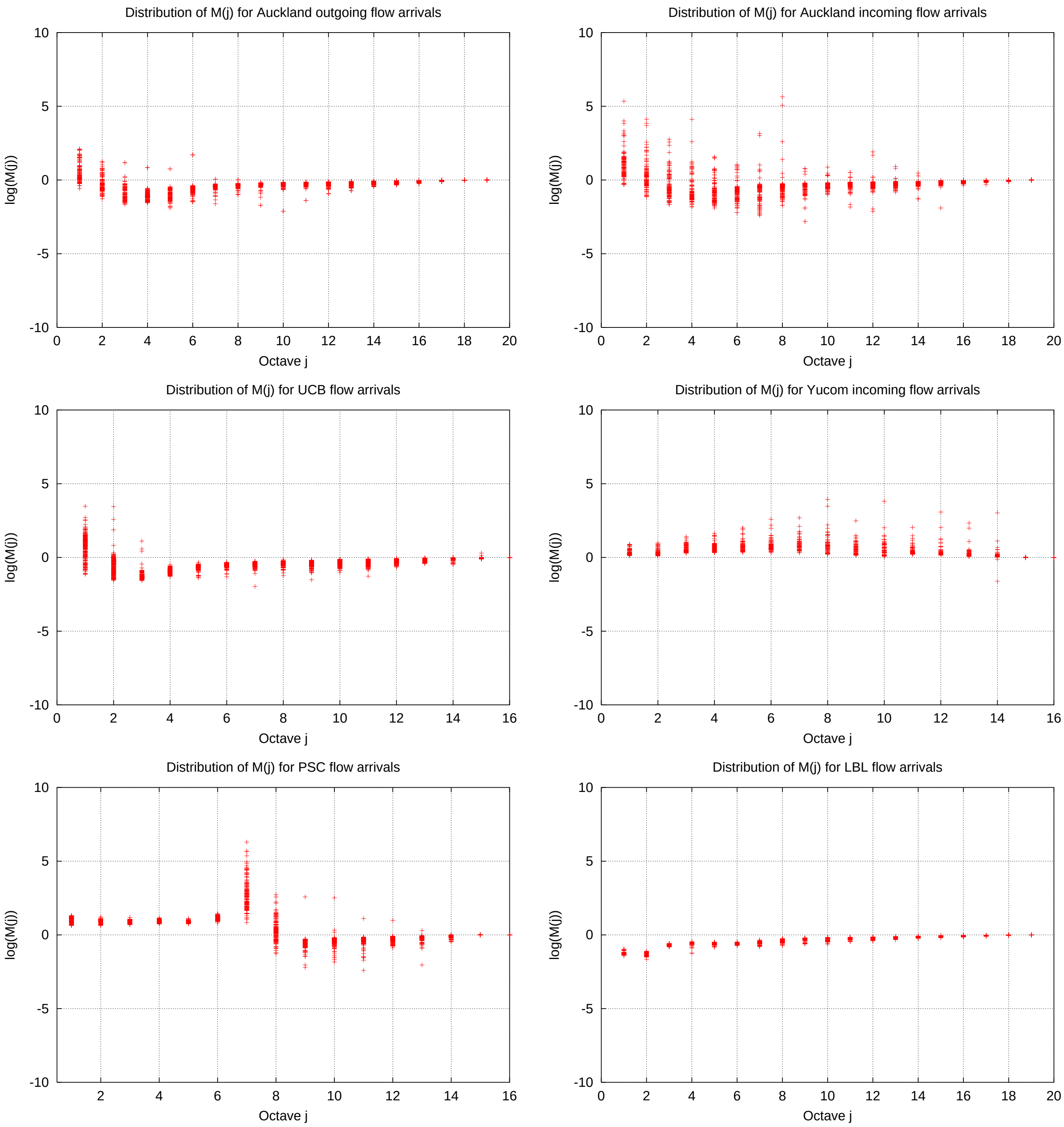



Figure 1.12: M(j) estimator.

length. A self-similar process requires a positive slope of the LD for all timescales, a stationary 3D-LD, high-order scaling, and a $log(M(j))$ of zero. Multiscaling appears for positive values of the $log(M(j))$.

The Auckland traces and UCB have mostly negative values of the $log(M(j))$, with the largest timescales being close to self-similar with an almost zero $log(M(j))$. Going towards smaller timescales shows decreasing values of the $log(M(j))$ until octave five (less than five hundred milliseconds) where it increases in a multifractal-like and time-dependent way, the values of the $log(M(j))$ increasing more or less linearly. Note that this multifractal behavior is probably due to a Poisson process since the second-order analysis pointed to an almost uncorrelated process at these timescales with a slope of the LD close to zero.

Yucom on the other hand has all values of the $log(M(j))$ well above zero all the time, with sharp peaks quite often. This process is not multiscaling but contains some kind of high-order intermittency due to the instability with time of the $M(j)$ estimator. The would-be multiscaling behavior of this trace looks like a noisy high-order process, with sharp peaks occurring without much consistency. The second-order properties of this process however do not differ from the ones of the other flow arrival processes, the only difference with being the unidentified non-stationarity through its second-order properties. All `http` traffic of the Yucom trace, incoming as well as outgoing, behaved in that way.

PSC contains a mix of stationarity for the smallest and the largest timescales, with stable multiscaling at the smallest timescales (octaves [1,5]), LRD, statistical dependence or no scaling at all at octaves [9,13] while self-similarity at octaves larger 13 where second-order self-similarity was found in [177]. Octaves [6,9] on the other hand exhibit a spikiness which we shall explain in the next section.

Finally, the LBL trace shows a neat graph with negative values of the $log(M(j))$ for almost all timescales and an increase of the $log(M(j))$ when going towards the largest timescales, corresponding to LRD or statistical dependence.

### 1.4.3 Applications behavior

The previous section has left unanswered the reasons for the differences in behaviors we found among the different traces. In this section, we partly address this issue by studying one particular trace, the PSC trace. We focus on this trace for three reasons. First, we have the information about the internal clients of PSC. We can analyze each client's traffic separately. Second, we know that the largest client undergoes rate limitation, whose impact on clients' behavior can be studied. Third, we know that a large fraction of the traffic is P2P (kazaa, Gnutella,...). We can thus compare the difference in scaling behavior between P2P applications and `http` traffic.

Table 1.1 first compares the flows number for the two largest clients, for `http` and P2P (only kazaa and Gnutella, ports 1214 and 6346). The number of P2P flows is quite large, representing 18 percent of the flows for the largest clients, 57 percent

Table 1.1: Comparison of flows mix for largest 2 PSC clients.

| Client | http | P2P | Total |
|---|---|---|---|
| 1 | 9,767,651 | 2,265,086 | 12,032,737 |
| 2 | 2,038,776 | 2,769,587 | 4,808,363 |

of the flows for the second largest client. This type of traffic seems to become more and more common, so a careful study of its behavior is appropriate.

Figure 1.13 presents the LD and 3D-LD for these two largest clients of PSC. The difference between the second-order properties of the two clients is obvious: client 1 has the mark of its rate limitation while not client 2. The LD of client 1 indicates an increase of the variance of its wavelet coefficients for the small timescales (up to 3 seconds) while client 2 has a more common LD. The 3D-LD confirms this behavior by showing an increased variance of the wavelet coefficients for client 1 throughout the day, so that rate limitation affects its flows arrivals all the time.

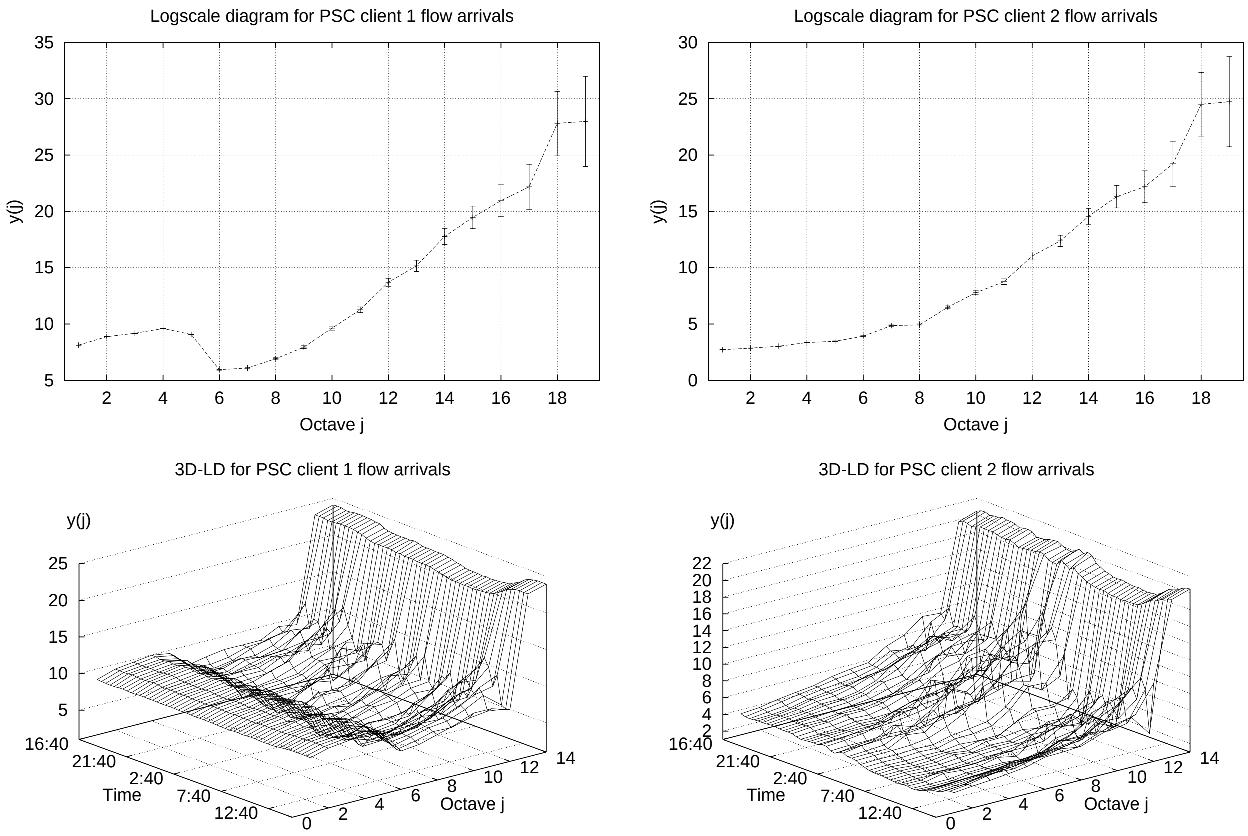



Figure 1.13: LD and 3D-LD for largest 2 PSC clients.

Figure 1.14 presents the evolution of the $log(M(j))$ over the day of the PSC trace also for the largest two clients. The effect of rate limitation (through a token bucket) for client 1 is to induce a linear high-order moments scaling (left of Figure 1.11) for timescales up to a few as well as stable multiscaling (bottom left of Figure 1.11) for the small timescales (up to a few seconds in our case). The effect of rate limitation is to shift the properties of the flows towards larger timescales, and to create time-homogeneous multiscaling over the small timescales. [71] did show that the small timescales of congested TCP traffic was multifractal. Here we show that such models as multiscaling and multifractals are even applicable for flow arrivals, not only for traffic volume at the packet-level.

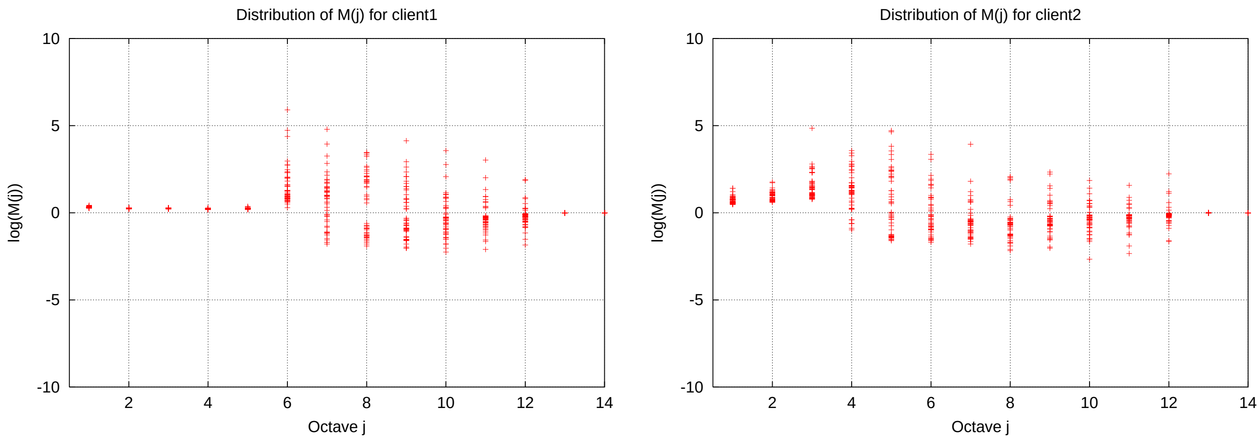



Figure 1.14: M(j) for largest PSC clients.

Finally, Figure 1.15 compares the behavior of the flows of the two largest clients, by splitting their flows into `http` (port 80) and P2P (ports 1214 and 6346). We can see on this figure that P2P traffic does not suffer from rate limitation, with clients one and two having a similar behavior of the $log(M(j))$, while `http` flows show a different behavior for the two clients. `http` flows for client one exhibit a graph of the $log(M(j))$ close to the one of all flows, while client two seems to be less impacted by client 1's rate limitation, although its behavior is still slightly different from normal `http` traffic like UCB's for instance (middle left of Figure 1.12).

## 1.5 Evaluation

In this section, we discuss the implications of the findings of this chapter on the roots of scaling in Internet traffic. Firstly, [51, 127] have shown the influence of the distributional properties of web traffic and file sizes on the statistical self-similarity of the traffic. This self-similarity was a second-order one, not strict self-similarity as used in this chapter, and it involved a single control parameter $H$ for all timescales. [163] also proved that an aggregation process with independent ON/OFF sources and heavy-tailed ON/OFF distribution times gives rise to a second-order self-similar process. Secondly, papers dealing with scaling in Internet traffic have convincingly shown the influence of the TCP protocol and its dynamics on the traffic scaling properties [128, 72, 82, 71].

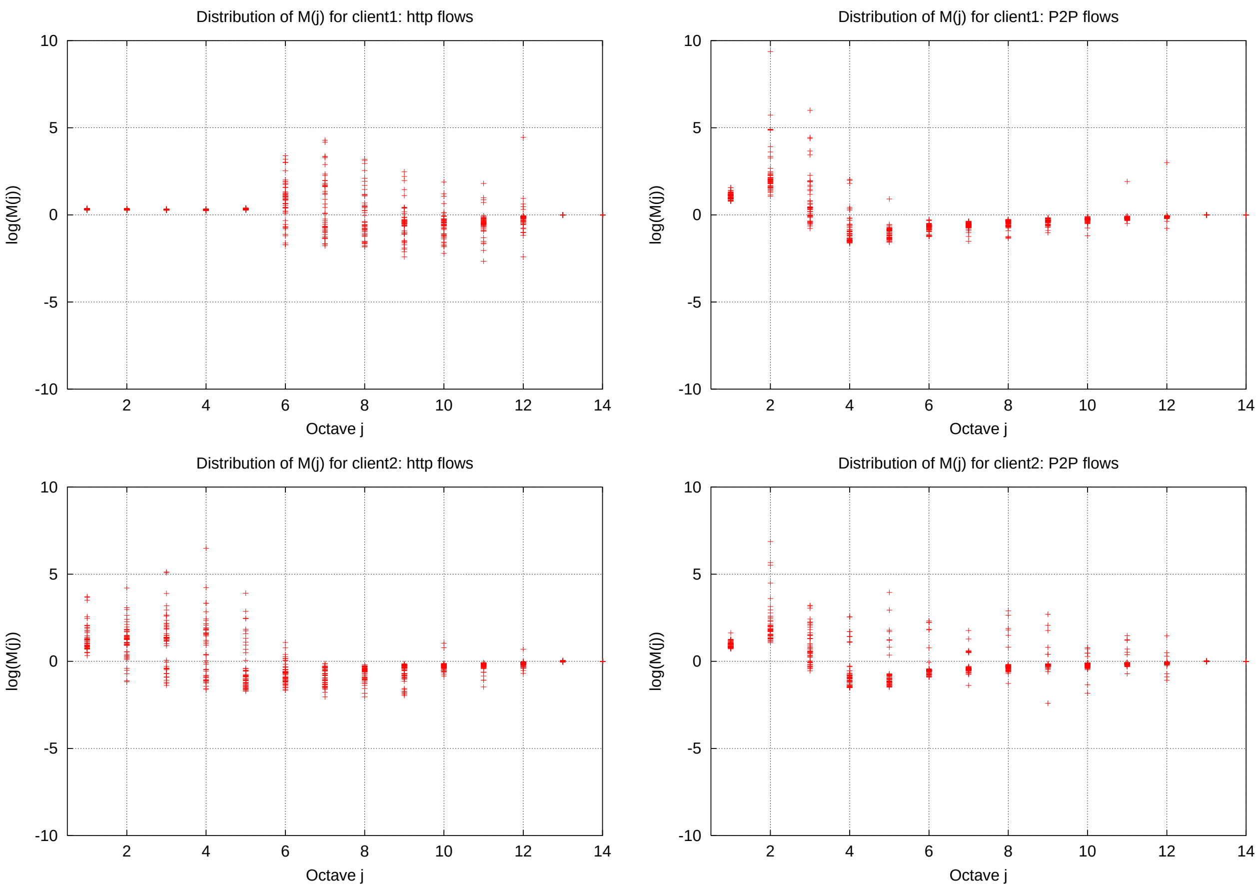



Figure 1.15: Comparison of scaling for PSC clients.

In section 1.4, we took a third perspective: the flow arrivals. We studied the process of the flow arrivals, and have shown not only that there is second-order scaling in this process, confirming [70, 177]; but that in addition higher-order scaling was also necessary to properly describe its dynamics. We showed that a wide range of scaling behaviors are present in this process, from multifractality in the small timescales of the `http` flows, to LRD, statistical dependence or no scaling in the medium timescales and finally exact self-similarity or LRD in the largest timescales. This chapter therefore points out the importance of the user's behavior through the process of the flow arrivals as another possible cause for the scaling in Internet traffic. While flow size distribution [51] in web traffic and network mechanisms [72, 82, 71] are undoubtedly partly involved in the scaling of network traffic, flow arrivals were shown in the previous sections to contain almost all types of scaling behaviors.

This chapter consequently asks for investigating the relationships between scaling in network traffic, users and applications behavior, through the process of the flow arrivals. We showed in this chapter that our understanding of the process of the flow arrivals is still limited and that interesting information could probably be gathered by thoroughly studying this process and the relationships between the network conditions and user's behavior. The high-order moments properties of the flow arrivals provide still largely unexploited information about the dynamics of network traffic.

In this chapter we studied the total traffic over timescales larger than minutes and the TCP flows arrivals for timescales ranging from milliseconds to days. Another interesting scaling property of the traffic is caused by the way TCP breaks the traffic of the flows into IP packets. This process is well modeled by a *conservative cascade* [82]. A *conservative cascade* is a cascade that accommodates the two competing objectives of deterministic and random cascades: 1) preservation of the total mass of the process at each step of the cascade and 2) randomness of the distribution of the mass among the subintervals. These two properties are intuitively valid for TCP traffic, where there is a random distribution of the mass of the traffic among the different traffic flows aggregated in the total traffic and a deterministic breaking of the IP packets within a flow due to TCP. The distribution of the packets is hence a mix between the deterministic way with which the TCP protocol distributes the mass of the traffic within a flow, and the randomness induced by the behavior of the network and its users. We studied packet-level traffic traces and showed in [170] that TCP traffic was consistent with a conservative cascade whose parameters' values were time-invariant, for timescales between a few milliseconds and a few hundreds of milliseconds. [170] showed that while the cascade model seemed to be an invariant of the traffic, the time-varying network conditions affected the second-order properties of the process as well as its multifractality.

On the one hand, conservative cascades are an invariant of the traffic for what concerns the timescales where TCP really matters,from milliseconds up to a few hundreds of milliseconds. Timescales of milliseconds correspond to the smallest timetick for the computer to send IP packets. With gigabit local area networks, these smaller timescales where the cascade model applies will go to microseconds. A few hundreds of milliseconds correspond more or less to a few round trip times, or the timescale at which TCP computes its estimate of the available bandwidth on the end-to-end path. The cascade model will therefore become more and more important as the bandwidth used by TCP flows increases since the the range of timescales over which TCP will drive the traffic dynamics will also increase.

On the other hand, the cascade model cannot capture what lies outside the scope of the behavior of TCP, timescales larger than a few RTT's. The time-varying network capacity that influences how TCP chooses to send data is not captured by the parameters of the cascade model. This is why we showed in [170] that the changes in network conditions imply second-order non-stationarity that changes the variability of the traffic differently on different timescales, to which the cascade model is insensitive. [170] showed that only traffic *received* by a stub was affected by this second-order non-stationarity, not traffic *emitted* by the stub. Only the time-varying network conditions like available bandwidth and end-to-end delays can explain this phenomenon. To conclude this section, we will just say that much remains to be done on the traffic dynamics side before we can state that we understand the traffic dynamics, non-stationarity must be taken into account to fill the gap between mathematical models and the real traffic dynamics.

## 1.6 Conclusion

Besides the various technical implications of network traffic self-similarity on the understanding of the traffic dynamics mentioned in the previous section, we conclude this chapter by discussing the implications on interdomain traffic engineering.

Self-similarity by itself is not an issue. The main issue concerns the implications of the properties of a self-similar signal on traffic prediction. From a statistical viewpoint, self-similarity implies that the extent of the variations in the signal scale linearly with the considered timescale. The aspect related to long-range dependence is secondary for what concerns traffic engineering. Correlations in the signal are of interest for applications leveraging the sample path properties of the process. For traffic engineering on the other hand, we are rather interested in knowing whether the process's variability is "stable" for control purposes. The analysis of the scaling properties of the traffic are important in this thesis because we need to know on which timescales variability in the interdomain traffic becomes sufficiently small not to be an issue for traffic engineering. We showed in this chapter that from the viewpoint of a stub AS, interdomain traffic variability becomes limited over timescales larger than one hour or so. Hence from a control perspective, interdomain traffic engineering should preferably work on timescales larger than hours.

# Chapter 2

# Topological distribution of interdomain traffic

## 2.1 Introduction

The behavior of Internet traffic has been analyzed by researchers almost since the creation of the first computer networks [99]. Many studies have tried to better understand the microscopic or packet-level behavior of the Internet. In this case, the analysis is usually performed on the basis of a capture of all the packets that flow through a given link for some period of time. This type of analysis has been popular thanks to the availability of powerful packet capture tools ([38, 94, 50] among others). By analyzing the packets in the recorded trace, researchers have identified scaling (see section 1.2.1) in network traffic [106, 131] and studied aspects like the packet size distribution, the application mix or the number of concurrent TCP connections [48, 165].

On the other hand, fewer studies have analyzed the macroscopic behavior of the Internet. By macroscopic behavior, we mean the topological distribution of the traffic throughout the global Internet. One of the early papers on this topic is probably [99] that analyzed the traffic distribution on the ARPANET in 1974. An important finding from this paper was that a few sites were responsible for a large part of the traffic. Similar studies were conducted in 1993 on the NSFNet backbone [47] on the basis of packet-level traces and SNMP statistics from backbone routers. They confirmed the findings of [99]. More recently, [66] analyzed several one hour packet level traces from universities and a commercial backbone to evaluate the impact of aggregating flows at the Autonomous System level.

In parallel to the measurements discussed in the literature, many network providers have deployed measurement systems to better understand the dynamics of the IP traffic inside their network notably for traffic engineering purposes. Several papers have described the tools deployed by these network providers [116, 37, 74, 15, 78], but few papers have used the output of such tools to analyze the macroscopic behavior of Internet traffic.

In this chapter, we present a detailed analysis of the characteristics of interdomain traffic by studying two one week long traces of the entire interdomain traffic received by two non-transit providers.

The structure of this chapter is the following. Section 2.2 first describes the two considered providers. Section 2.3 studies the aggregation provided by different notions of the traffic flows. Then, section 2.4 analyzes the topological distribution of the traffic. This topological analysis is refined in section 2.5 that studies the distribution of the active traffic sources.

## 2.2 Measurement environment

### 2.2.1 The studied providers

The only characteristic common to both providers is that they do not offer transit service. Besides, they serve different customers and it can be expected that these customers have different requirements on the network. Due to technical reasons, it was unfortunately impossible to obtain traces from the two studied providers covering the same period of time.

The first trace was collected in December 1999 and covers six successive days of all the interdomain traffic received by BELNET. BELNET provides connectivity for the research and education institutions located in Belgium. At that time, the BELNET network was composed of a 34 Mbps star-shaped backbone linking the major universities. Its interdomain connectivity was mainly provided through 34 and 45 Mbps links with two large tier-1 providers. In addition, BELNET had a 45 Mbps link to the European research network, TEN-155, and was present at the BNIX and AMS-IX interconnection points with a total of 63 peering agreements in operation. Although some universities provided dialup access for their students, the typical BELNET user had a 10 Mbps access link to the BELNET network through their university LAN. During the six days period, BELNET received 2.1 TBytes of data. BELNET is representative of research networks and could also be representative of a provider providing services to high bandwidth users with cable modem or ADSL.

The left part of Figure 2.1 shows the evolution of total traffic for BELNET during the period of the measurements. While the global evolution of the total traffic exhibits a daily periodicity, with peak hours located during the day, there are important deviations around the average traffic evolution throughout the day. The mean traffic over the six days period was slightly larger than 32 Mbps, with a one-minute maximum peak at 126 Mbps and a standard deviation of 21 Mbps. The trace begins around 1 AM on a Sunday and finishes six days later around 1 AM also.

The second trace was collected in April 2001 and covers a little less than five consecutive days of all the interdomain traffic received by Yucom. Yucom is a commercial provider that provides Internet access to dialup users through regular modem pools. At that time, the interdomain connectivity of Yucom was mainly provided through high bandwidth links to two transit providers. In addition to this transit service, Yucom was also present at the BNIX interconnection point with 15 peering agreements in operation. During the five days of the trace, Yucom received 1.1 TBytes of data. Yucom is representative of providers composed of low bandwidth users.

The right part of Figure 2.1 presents the total traffic evolution for Yucom during the measurements. The trace starts around 8:30 AM on a Tuesday and finishes almost 5 days later at midnight. The total traffic also exhibits a daily periodicity with peak hours located during the evening, in accordance with the typical user profile, a dialup user. It had an average total traffic of about 23 Mbps over the measurements, with a one-minute maximum peak at 64 Mbps and a standard deviation of 12 Mbps.

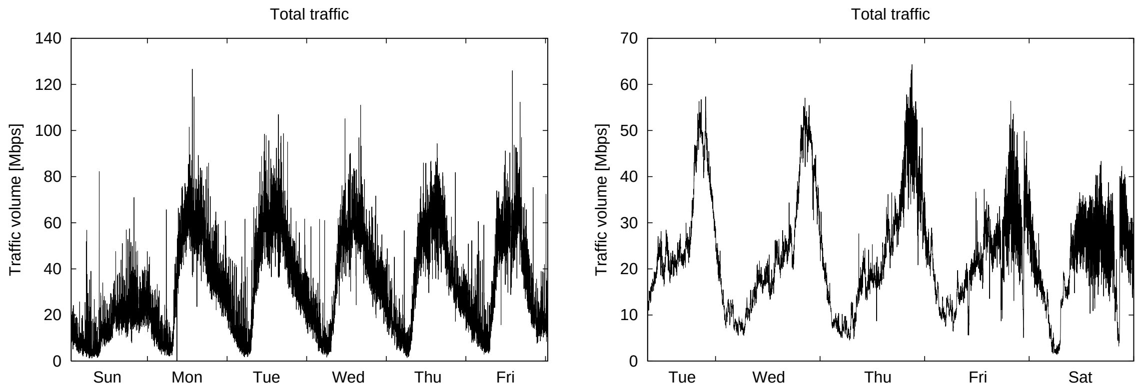



Figure 2.1: Total traffic evolution, BELNET (left) and Yucom (right).

### 2.2.2 Interdomain Topology

Before analyzing in details the collected traffic statistics, it is useful to have a first look at the BGP table of the studied providers. In this chapter, we assume that the BGP table of both providers was stable during the period of the measurements and perform all our analysis based on a single BGP table for each provider. We assume in this chapter that the influence of the changes in the BGP routes is negligible. Chapter 3 will confirm that relying on a single BGP routing table is not a bad approximation.

The routing table of the Yucom contained 102,345 prefixes, covering about 26 percent of the total IPv4 address space. This coverage of the total IPv4 address space is similar for BELNET, with about 24 percent but for 68,609 prefixes only. Form late 1999 to mid-2001, thirty percent more prefixes are necessary to cover a similar percentage of the IPv4 address space. Although having different numbers of prefixes in their BGP routing table, the two providers cover a similar percentage of the IPv4 address space. The average address span per prefix for each provider, about 15,200 IP addresses for BELNET and about 11,000 IP addresses for Yucom, explains this increase in the number of prefixes necessary to cover a similar percentage of the IPv4 space. Yucom knew 10,560 distinct ASes while BELNET 6298. This difference is mainly due to the large increase in the number of multi-homed sites during the last few years [93].

Figure 2.2 compares the distribution of the potentially reachable IP addresses at each AS hop distance for the BGP routing tables of the two providers. For each BGP prefix, we attributed the number of IP addresses according to its prefix length to the AS hop distance equal to its unprepended AS path length. While this does not measure the actual number of IP addresses at each AS hop distance, it reflects the potential reachability of IP hosts for each AS hop distance seen by BGP routes. The main difference between the two ISPs is the more compact distribution for Yucom around a distance of three AS hops. BELNET has its reachable address space more spread over distances of three and four AS hops. The first three AS hops for the Yucom provide almost eighty percent of the reachable address space while only about sixty percent for the research ISP. The difference between the distribution of the reachable IP prefixes seen from the two providers can be due to the way the two providers are connected to the Internet as well as the sixteen months delay between the two traces.

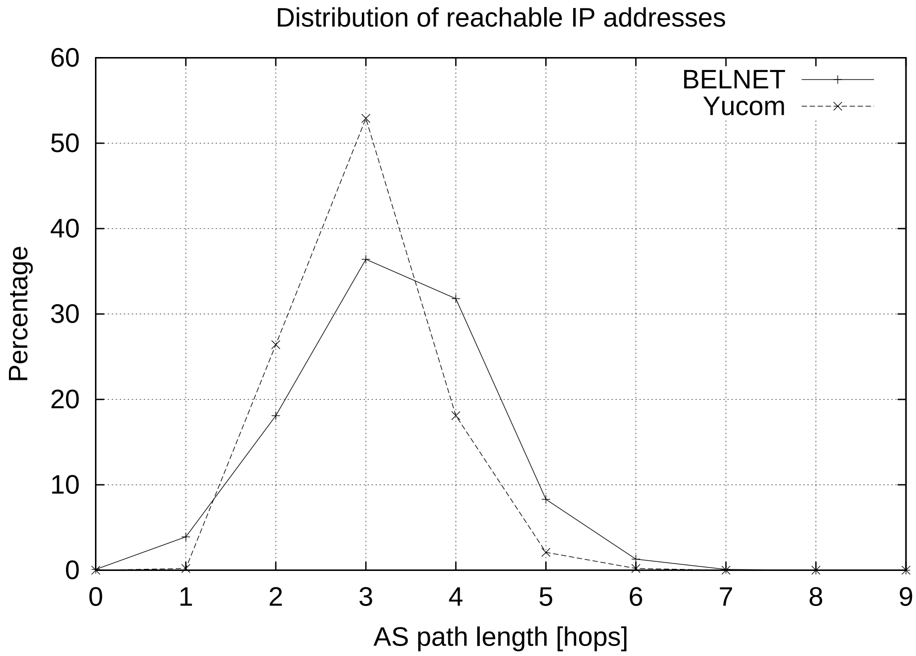



Figure 2.2: Distribution of reachable IP addresses.

## 2.3 Aggregation of interdomain flows

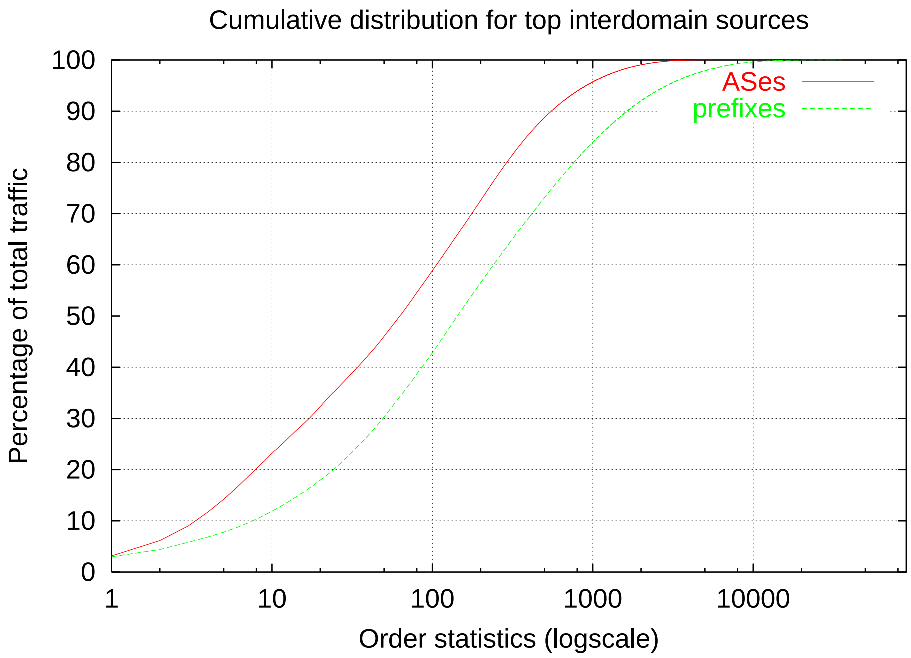
To understand the topological variability of interdomain traffic and the possible levels of aggregation, we consider in this chapter two different types of interdomain flows. Generally, a flow is defined as a set of IP packets that share a common characteristic. For example, a micro-flow is usually defined as the set of IP packets that belong to the same TCP connection, namely the IP packets that share the same source address, destination address, IP protocol field, source and destination ports. In this chapter, we consider two different types of network-layer flows. A *prefix flow* is the set of IP packets whose source addresses belong to a given network prefix as seen from the BGP table of the studied provider. An *AS flow* is defined as the set of IP packets whose source addresses belong to a given AS as seen from the BGP table of the studied provider. We do not use explicitly throughout this chapter the term "flow" to designate traffic coming from a traffic source, but mostly the terms "prefix" and "AS" (or "source AS") to denote a *prefix flow* and *AS flow* respectively.

We study in this section the amount of aggregation provided by the AS and prefix flows. Figure 2.3 shows the cumulative percentage of the traffic or *order statistics*, for AS and prefix flows. On this figure, we ordered the prefixes and ASes by decreasing order of the total traffic volume sent by them over the whole measurements, and we computed their cumulative contribution to the total traffic over the measurements. The $x$-axis uses a logarithmic scale to emphasize the low *order statistics*. Both providers have a similar cumulative traffic distribution for their most important interdomain traffic sources. The largest one hundred AS flows capture about 72 percent of the total traffic for Yucom and a little less than 60 percent for BELNET. The largest one hundred prefix flows capture about 52 percent of the total traffic for Yucom and a little more than 40 percent for BELNET. 90 percent of the total traffic volume is captured by only 4.7 percent of the ASes and by 4.1 percent of the prefixes present in the BGP table of Yucom. BELNET required 9.8 percent of the ASes and 4.5 percent of the prefixes present in its BGP routing table to capture 90 percent of the total traffic volume. These results are similar to the findings of earlier studies [99, 47] on the research Internet of the 1970's and the early 1990's. [67] in 2001 showed that over a one day period, 10 percent of the destination prefixes captured 95 percent of the total outbound traffic of a border BGP router at AT&T. On the other hand, some ASes and prefixes contributed to a small fraction of the total traffic. For Yucom, more than 4000 different ASes sent each less than one megabyte of data during the measurement period and some ASes only sent a single packet during this period. For BELNET, 719 ASes sent less than one megabyte of data during the six days measurement period.

Over the measurement period, BELNET received IP packets from 5606 ASes and 35,688 prefixes, corresponding to 89 percent of the ASes present in its BGP table. Yucom received IP packets from 7668 ASes and 35,693 prefixes, corresponding to 72 percent of the ASes present in its BGP table. These figures show that even relatively small providers receive traffic from a large portion of the Internet during a one week period, even though some sources only send a few packets.

## 2.4 Topological distribution of the traffic

The amount of traffic aggregation provided by the interdomain flows is not the only important aspect to be considered when studying the macroscopic behavior of Internet traffic. Another important aspect concerns the topological distribution of the traffic. By topological distribution, we mean the distance between the traffic



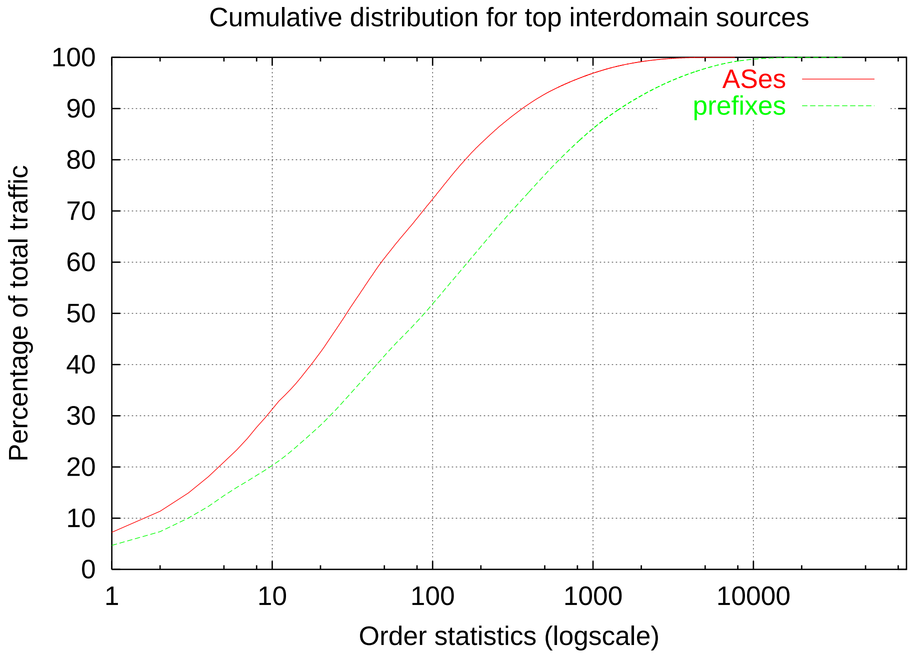
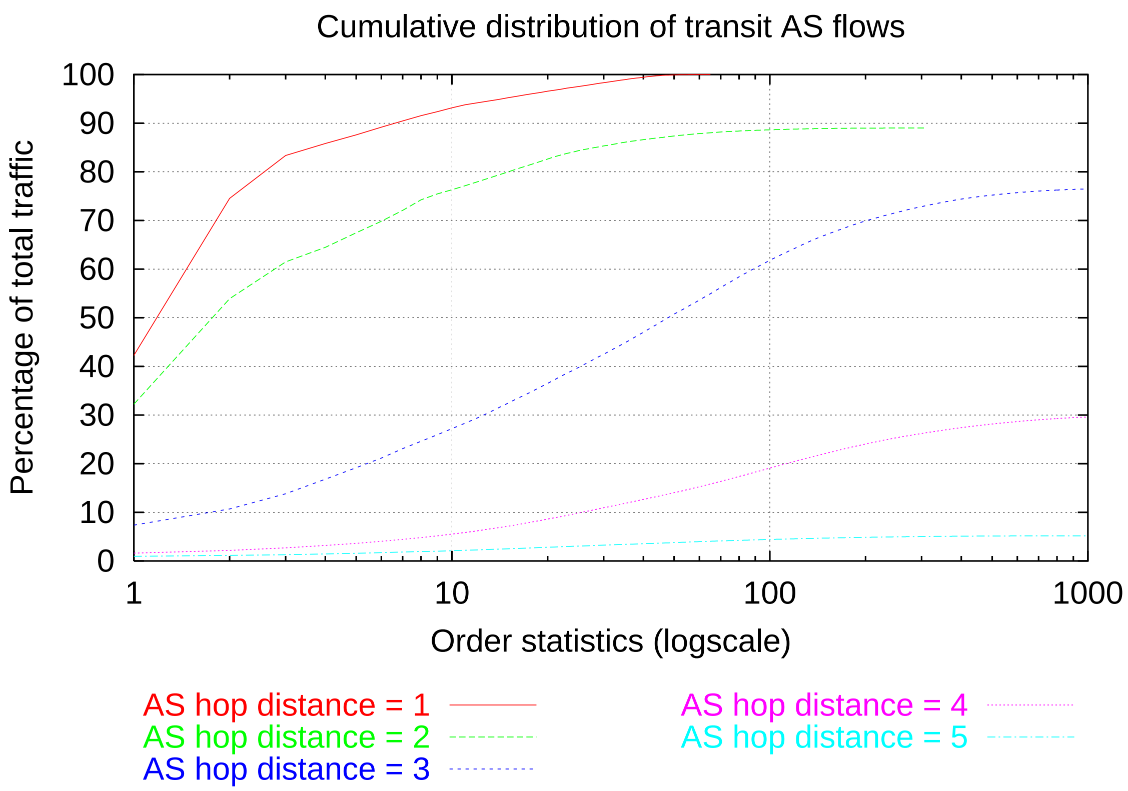




Figure 2.3: Traffic aggregation for interdomain flows, BELNET (top) and Yucom (bottom).

sources and the studied ISP. This distance is important for two reasons. First, [117] has shown that the performance of an Internet path decreases with the distance between the source and destination AS, even if [92] showed more mitigated results. Second, if the distance between the source and the destination AS is large, it will be difficult for either the source or the destination to apply mechanisms to control the traffic flow in order to perform interdomain traffic engineering [17] (see also Chapter I).

Figure 2.4 presents the cumulative traffic distribution for the order statistics of the ASes for each AS hop distance. Considering AS flows does not show the traffic aggregation occurring on the AS-level topology. We thus computed the traffic volume sent by each prefix of the BGP routing table. Then, based on the `AS path` attribute of the BGP routes, we attributed to each AS appearing in the `AS path` attribute of the route the total traffic received from the prefix of the BGP route. Doing this amounts to consider each AS not as a traffic source but as a transit. The objective of this procedure is to attribute to each AS of the Internet the traffic volume it has seen for the two studied providers during the period of the measurements. An AS located at an AS hop distance of $n$ is seen as the source of the traffic it generates as well as of all the traffic it forwards when considering the unprepended `AS path` information of the BGP routing table. Because each AS hop distance does not contribute evenly to the total traffic, we have plotted the cumulative traffic percentage for every AS hop distance with respect to the total traffic seen during the measurements, thus independent of the locality of the traffic.

Each curve of Figure 2.4 shows the cumulative traffic percentage seen by the largest ASes ordered by decreasing amount of the traffic volume seen by them over the whole measurements. A point of a curve shows the traffic percentage seen by the largest ASes at some AS hop distance. For example, the first curves on top of both graphs of Figure 2.4 show the cumulative percentage of traffic captured by the ASes located at one AS hop form the two considered providers. The AS located at one AS hop are the BGP peers of the two studied providers. Both curves end up at one hundred percent of the total traffic, indicating that all traffic was seen by the ASes located at one AS hop. The curves for larger AS hop distances on the other hand do not attain one hundred percent of the total traffic. The difference between the largest percentage of traffic seen at two successive AS hop distances tell the percentage of the traffic that was generated by ASes located at the first AS hop distance. For instance, the difference between the largest traffic percentages for the one AS hop distance and two AS hop distances on Figure 2.4 give the percentage of traffic generated by ASes located at one AS hop. In this particular case, the ASes at one AS hop generated 11 percent of the total traffic volume for BELNET and a little more than 6 percent for Yucom. ASes at two AS hops generated a little more than 12 percent of the total traffic volume for BELNET and about 42 percent for Yucom. Then, ASes at three AS hops generated about 46 percent of the traffic for BELNET and about 44 for Yucom. The last AS hop distance for which a significant percentage of the traffic was received is for four AS hops with about 25 percent for BELNET and about 7 percent for Yucom. Subsequent AS hop distances did not send a significant percentage of the total traffic for none of the two providers.



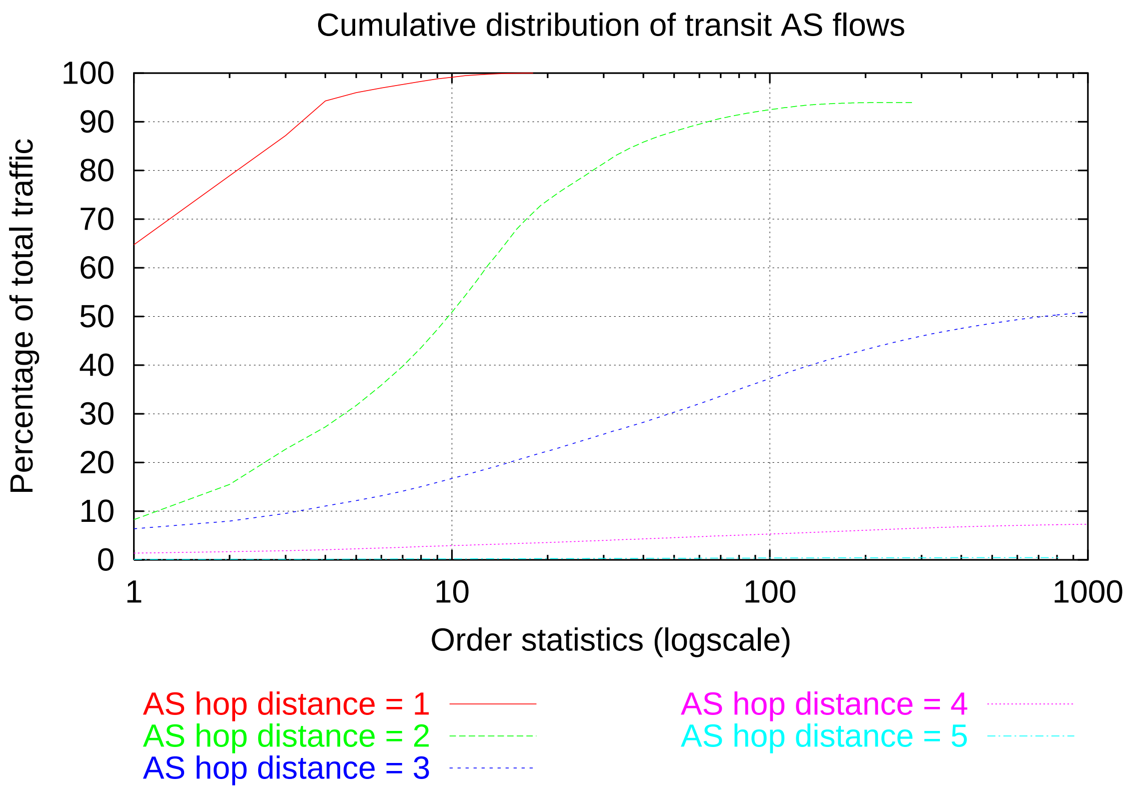




Figure 2.4: Topological traffic distribution, BELNET (top) and Yucom (bottom).

Now if we focus our attention on one particular curve of Figure 2.4, we can study how many ASes must be considered to capture some percentage of the total traffic. Take for instance the curve for a distance of one AS hop. On this curve, we see that the AS located at one AS hop that sees the largest amount of traffic for BELNET sees 64 percent of the traffic while 42 percent for Yucom. The three ASes at an AS hop distance of one and seeing the largest amount of traffic for BELNET see 83 percent of the traffic for BELNET, while for Yucom the largest three ASes at one AS hop see 87 percent of the traffic. Since ASes located at one AS hop are either transit providers used by the local AS for its Internet connectivity or peer ASes with whom local traffic is exchanged, the normal situation is the one where a very small number of ASes at one AS hop see a large fraction of the total traffic sent or received by a non-transit provider.

At a distance of two AS hops, a significant difference between BELNET and Yucom appears. AS two AS hops, the largest ten ASes for BELNET see about 75 percent of the traffic, while for Yucom the largest ten ASes at two AS hops see only about 50 percent of the traffic. However, the largest one hundred ASes at two AS hops capture a similar percentage of the total traffic of BELNET and Yucom, about 90 percent. The number of ASes required to capture a given percentage of the total traffic at some AS hop distance indicates how many ASes the incoming traffic is splitted on the AS-level topology. Suppose that we wanted to know for each AS hop distance the least number of ASes that see 40 percent of the total traffic. Then we take Figure 2.4 and find when the curve for each AS hop distance crosses the horizontal line indicating 40 percent of the total traffic. For BELNET, it gives one AS at one AS hop, two ASes at two AS hops, and less than 30 at three AS hops. For Yucom, it gives one AS at one AS hop, eight ASes at two AS hops, and more than one hundred ASes at three AS hops. The incoming traffic of Yucom is hence splitted over more ASes on the interdomain topology than the one of BELNET.

## 2.5 Activity of the Traffic Sources

This section studies the *activity* of the interdomain traffic sources, hence the ASes that generate traffic. We say that a traffic source is *active* during a given time interval whenever there is at least one byte of traffic that originates from it during this time interval. Note that this notion of *activity* pays no respect to the variations in time of the amount traffic for the traffic flows. We look at the number of AS flows that are *active* over some time interval as well as the number of *active* AS flows for different timescales.

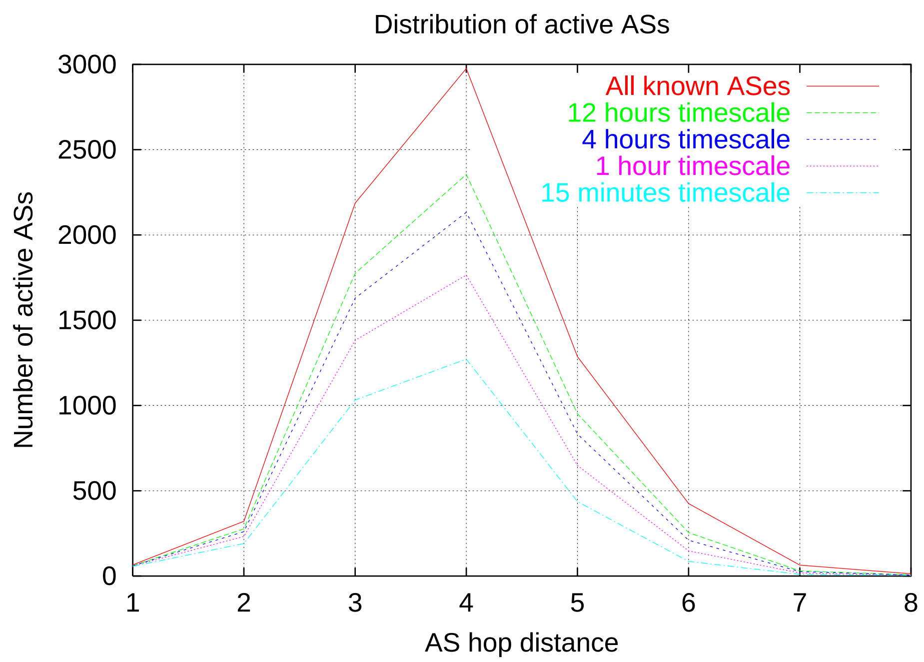
We first study the activity of the AS flows in Figure 2.5. Figure 2.5 shows, over the measurement period, the average number of *active* AS flows for each AS hop count for timescales of fifteen minutes, one hour, four hours and twelve hours. The number of *active* AS flows is compared with the total number of ASes known from the BGP table. If we compare the number of AS known from the BGP table for each AS hop distance, we realize that it differs from the distribution of the reachable address space for each AS hop distance shown on Figure 2.2. For BELNET, most ASes are located at four, three and then five AS hops. The distribution of the reachable space of Figure 2.2 on the other hand tells that there are more potential sources at three, two, then four AS hops. For Yucom, most ASes are located at three, four and then five AS hops for a distribution of the reachable space with more potential sources at three, two and four AS hops. This shows that the distribution of the reachable address space for each AS hop distance is a poor heuristics to guess the distribution of the ASes with whom traffic is exchanged.

The different curves of Figure 2.5 also show that the distribution of the average number of *active* ASes does not change fundamentally for the different timescales considered. The look of the distribution is similar for all timescales, what changes is the total number of *active* ASes that varies by a factor of less than ten between the smallest and the largest timescales considered. During a fifteen minutes period, BELNET receives packets from an average of 3088 different ASes. For the twelve hours timescale, the number increases to 5711 ASes. This should be compared to the 6298 ASes that appear in the BGP table of BELNET. Yucom for its part receives on average packets from 1632 ASes during a fifteen minutes interval and from 5721 ASes for a twelve hours interval.

The ASes that are direct peers of the studied providers are the most *active* source ASes. During any fifteen minutes period, an average of 16 ASes at a distance of one AS hop are *active* for Yucom. During a period twelve hours, this number increases to 17 different ASes. This corresponds to all the ASes known by Yucom at a distance of one AS hop. For BELNET, 90 percent of the ASes at one AS hop are *active* on average during any fifteen minutes interval and this percentage increases to 96 percent when considering twelve hours intervals. The percentage of *active* ASes at a distance of two AS hops ranges between 58 percent for the fifteen minutes intervals and 85 percent for the twelve hours intervals for BELNET. The same numbers for Yucom are of respectively of 62 percent and 90 percent. BELNET has a percentage of *active* ASes that slowly decreases with the AS hop distance. Yucom experiences an abrupt decrease of the percentage of its *active* ASes at an AS hop distance of three due to the large fraction of the ASes known at this AS hop distance.

Let us turn to *active* prefixes. Figure 2.6 shows the activity of prefixes, which follow a distribution similar to the one of the ASes. For BELNET, there are on average 7029 different *active* prefixes during a fifteen minutes interval, about ten percent of the prefixes contained in its BGP table. For Yucom, the average number of *active* prefixes during a fifteen minutes interval is 3719, representing only about four percent of the prefixes of its BGP table. The average number of *active* prefixes for Yucom increases to 18,948 for twelve hours intervals, to be compared with the 102,345 prefixes known from its BGP table.

The distribution of the traffic sources in terms of AS hop distance is hence not always consistent with the reachability information of the BGP table. Considering AS flows instead of prefixes allows for a reduction in the number of *active* sources to be taken into account, but this reduction is limited given that the number of *active* prefixes is already fairly low compared to the number of known prefixes.



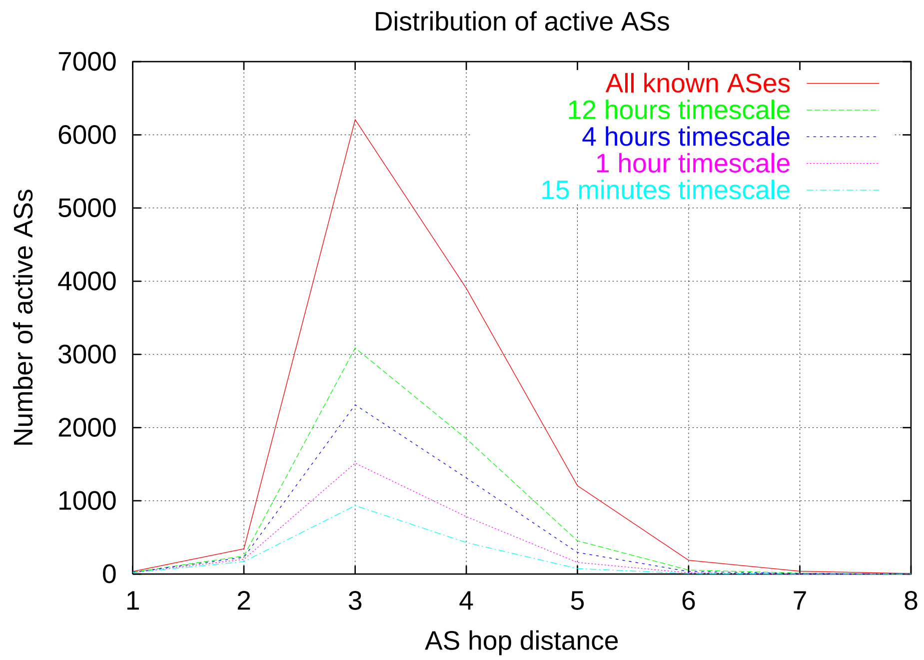




Figure 2.5: Number of *active* ASes, BELNET (top) and Yucom (bottom).

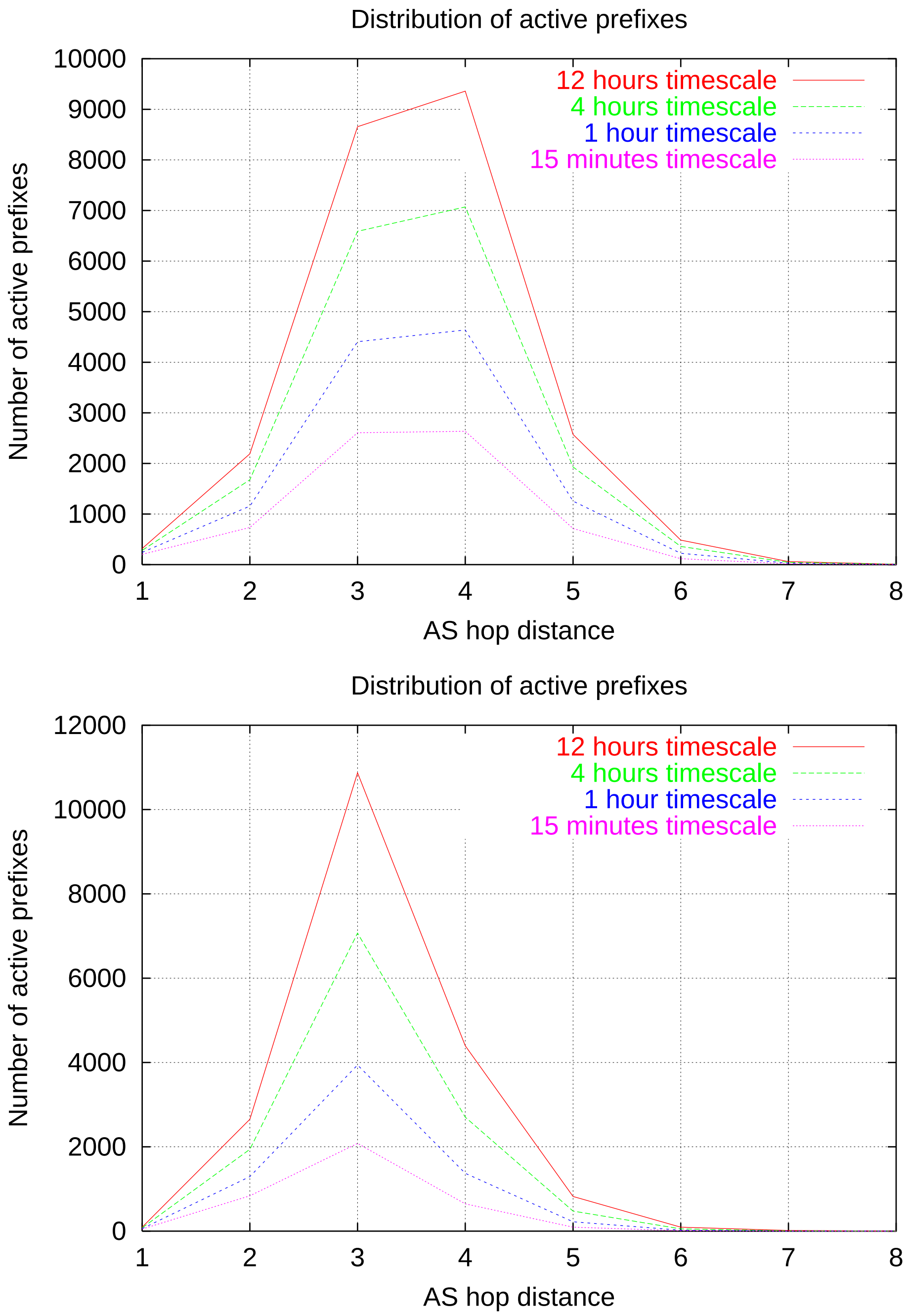



Figure 2.6: Number of *active* prefixes, BELNET (left) and Yucom (right).

## 2.6 Conclusion

In this chapter, we have analyzed this behavior on the basis of two one week traces covering all the interdomain traffic of two different medium size non-transit providers at two different moments. Our study is probably applicable to many small to medium ISPs that constitute the majority of the ISPs in the Internet, even if the distribution in terms of AS hop distance will depend strongly on the particular provider. The topological distribution of the interdomain traffic however mainly depends on the way the considered provider is connected to the Internet as well as the most popular ASes with whom traffic is exchanged.

The main results of our analysis are the following. First, the distribution of the reachable address space is limited in terms of AS hops. Second, we confirm that a small number of interdomain traffic sources account for an important part of the total traffic received by the studied providers, with a relatively small difference between the prefix and the AS aggregation levels. Third, the most important part of the traffic is produced by sources located at a distance of a few AS hops, but the distribution of the traffic differs from the distribution of the reachable address space. Fourth, the distribution of the average number of interdomain traffic sources is also limited in terms of the AS hop distance and can also differ from the distribution of the reachable address space as well as of the distribution of the traffic. On the other hand, the distribution (not the total number of sources) of the average number of interdomain traffic sources does not change much with the considered timescale. Fifth, the studied providers, despite their size, receive traffic from a large fraction of the global Internet.

# Chapter 3

# Topological dynamics of interdomain traffic

## 3.1 Introduction

The common thread in papers dealing with interdomain traffic [99, 47, 66, 69, 174, 137] is that a limited number of ASes are responsible for a large fraction of the interdomain traffic. The main limitation of these studies is that they all consider the traffic distribution for a given time period, assuming that the timescales smaller than the whole time study does not contain significant traffic variability.

The aim of this chapter is to study the dynamics of the traffic on the interdomain topology, as seen by two stub ASes. Compared to [139, 149, 100, 147] that focus on a subset of the applications in the whole traffic, our goal is to study the dynamics of all traffic sent by stub ASes, independently of the underlying applications. This chapter focuses exclusively on these stub ASes because they constitute the vast majority of all ASes [157, 9]. More than 60 % of all ASes have at least two BGP peers [157]. At the date of December 20$^{\text{th}}$ 2003 more than 86 % (14221 over 16359) of the ASes were considered as stub-ASes by APNIC's BGP routing table [151].

In section 3.2 we explain the context of this chapter. Section 3.3 presents the traffic traces as well as the procedure used to merge the traffic and BGP routing information. Section 3.4 studies the stability of the BGP routing for the interdomain traffic. Section 3.5 analyzes the topological properties of the interdomain traffic over the whole traces while section 3.6 looks at the dynamics of the properties of the interdomain topology for the traffic. Note that most of the content of this chapter will appear as [178].

## 3.2 Studying the interdomain topology

In order to understand the topological distribution of interdomain traffic we rely on an AS-level graph. This graph is built from the AS path information contained in the BGP advertisements received by the studied stub ASes. More precisely, each

AS is one node of the interdomain graph and an edge correspond to the peerings between two ASes. A node of our AS-level graph may correspond to several distinct prefixes. For example, the node corresponding to a large provider will correspond to the domain of this provider and to several of its smaller clients that do not have an AS number. Furthermore, an edge in the interdomain graph may correspond to several distinct physical links or peerings. Since BGP only advertises one path to each destination, the considered interdomain graph does not show all interconnections between domains.

On this interdomain graph, we are interested in the paths followed by IP packets sent by the monitored stub AS to the global Internet. Those paths are the one selected by the BGP decision process [156] based on the advertisements received from each of its BGP peers. Since BGP is a dynamic routing protocol, these paths may change with time. Furthermore, since an AS may advertise several prefixes and the AS Path information is associated with one prefix, there might be several distinct paths between the studied stub AS and a given destination AS.

In this paper, we call an "edge" an AS pair appearing as two consecutive and distinct ASes in the AS path of the *best BGP route* for some traffic destination. With the BGP routes, we know how the interdomain traffic gets forwarded across the AS-level topology without respect to physical links or multiple peering points between two ASes. With the AS path information of the BGP routes, only "edges" of the AS-level topology are known. An example will better illustrate this notion of an "edge". In the left part of Figure 3.1, the stub AS (AS A) has two BGP peers (AS B and AS C) and each BGP peer announces one route towards each of the two destination ASes (AS F and AS G). In this topology, the stub AS receives two routes towards destination AS F from its two peers. The route advertised by peer 1 has AS path *B-F* while the route advertised by peer 2 has AS path *C-B-F*. Assuming that the stub chooses as its best route towards AS F the route advertised by AS B having the shortest AS path, we have two edges in the interdomain topology for this AS path, namely edge <A,B> and edge <B,F>. Our stub AS receives two routes for destination AS 2 having the same AS path length of 3 from its two peers, *B-D-G* for peer 1 and *C-E-G* from peer 2. If we also assume that our stub chooses the route advertised by peer 2 we have three new edges; <A,C>, <C,E> and <E,G>.

In the AS-level graph, we are interested in studying and understanding the topological distribution of the interdomain traffic. For this, we need to associate each "edge" with the amount of traffic it carries. We thus count all network traffic that transits between two ASes connected by that edge. An edge does not represent a physical link or a single peering between two ASes, but all peerings between two ASes. From a topological characterization perspective, "edges" represent an aggregate of links or peerings. From a traffic engineering perspective on the other hand, "edges" represent a meaningful traffic aggregate that could be controlled through BGP tweaking like AS path prepending, MED, or redistribution communities [137].

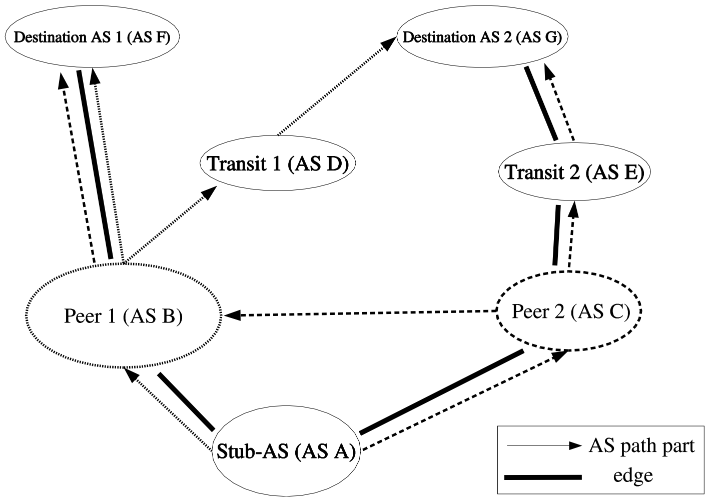



Figure 3.1: Example AS-level topology.

## 3.3 Measurements

In this chapter, we rely on two traffic traces along with the associated BGP routing information. The first trace is a one entire month trace of the outbound traffic from the Université catholique de Louvain (UCL), Belgium. The BGP data was gathered from a session established with the associated network provider, BELNET. To the best of our knowledge, this trace is the longest and most recent interdomain traffic trace of all the interdomain traffic reported in the literature. This trace was collected by the author. The second trace, mainly used for validation and comparison purposes, was gathered over 24 hours from the outbound commodity traffic from the Pittsburgh Supercomputing Center (PSC), Pennsylvania, USA. This trace was captured by Chris Rapier from the Pittsburgh Supercomputing Center. Its associated BGP information was gathered from an on-site route server.

Capturing such interdomain traffic traces is particularly difficult because of the anonymization issues involved in analyzing the whole interdomain traffic of some network. Most ISPs at least require that all IP addresses be anonymized so that most traces available on the web have all IP addresses related information lost, making them useless for studying the topological properties of the traffic that require that all flows or packets be routed with BGP routing tables to know where the sources and destinations of the traffic are located and to determine the actual path between them when the traffic crossed the Internet.

Given that the aim of this chapter is to study macroscopic properties of interdomain traffic the decision was made to use a granularity of one hour for the timescale. The reason for this choice is mainly practical, in that for a trace of one month duration a finer granularity would prove to be cumbersome. In addition, Chapter 1 has shown that at timescales smaller than one hour, LRD and self-similarity will make the dynamics of the traffic difficult to understand. This choice of one hour is therefore the smallest timescale at which we can expect a relatively smooth behavior of the traffic on the AS-level topology along with a reasonable time granularity of the study for practical purposes.

### 3.3.1 The UCL trace

The primary data set used is a month long trace of all the outbound traffic from the University of Louvain between March $19^{th}$ 2003 0:00 and April $18^{th}$ 2003 23:59 CET. During this period, 8.4 TBytes of traffic were observed with an average bit rate of 25.1 Mbps. The University's connectivity is a heavily loaded full duplex fast Ethernet link to the service provider BELNET (AS 2611).

Inbound and outbound flows were tapped and sniffed using separate fast Ethernet NICs in a single host. The host used to capture the traffic was a P4 2.4 GHz with 256 MB of RAM running FreeBSD 4.7. Two fast Ethernet NICs were used to gather traffic information while a third was used to place the host on the University's internal network. To gather the traffic statistics, we relied on `nProbe` [59], a flow-level collection software that captures packets arriving at a NIC and emulates the behavior of `Netflow` by exporting flow-level statistics. Maximum flow lifetime and timeout period were forced to be 30 seconds. No packets were reported as dropped by `nProbe` during the one month period. An eBGP session was established between the sniffing machine and the BELNET router serving the university using the Zebra routing daemon [2].

The majority of the university's user base consists of about 25,000 students and 5000 additional staff using the internal university network of 10 Mbps and 100 Mbps Ethernet links. The university also provides ADSL and cable modem access to some students and staff. The university's Internet connectivity is provided by the Belgian research network, BELNET (AS2611, `http://www.belnet.be`) which is connected to two tier-1 providers and the European research network GEANT, in addition to more than 100 peers at several national interconnection points (BNIX, AMS-IX, and SFINX). The internal network of BELNET consists of all Belgian universities and research institutions, linked by a 2.5 Gbps star configuration between its main routers in Brussels and each university. To our knowledge, BELNET did not perform any kind of traffic engineering on its commercial providers but relied on the shortest AS path towards each destination. All academic traffic was sent and received through the GEANT network. A limit of 45 Mbps was enforced by BELNET for commodity traffic of the university in the incoming direction only, and no traffic limitation was applied for traffic among the BELNET sites nor for traffic exchanged with universities in Europe or elsewhere through the GEANT network.

### 3.3.2 The PSC trace

The Pittsburgh Supercomputing Center (PSC) is a regional aggregation point of presence located in western Pennsylvania, USA. PSC provides commodity and Internet2 access to local universities and organizations including Carnegie Mellon University, The University of Pittsburgh, Pennsylvania State University, West Virginia

University, The City of Pittsburgh, and others. Currently PSC has a maximum capacity of 395 Mbps of commodity access though AT&T at 145 Mbps and Verio with 250 Mbps via an OC12. Additionally PSC has a full OC48 of Internet2 connectivity through the Abilene network.

The PSC trace used in this study is composed of all the outbound commodity traffic from 8:47 AM 18 April 2003 to 8:45 AM 19 April 2003. During this period 1.7 TBytes of traffic were observed for an average rate of about 164 Mbps.

The trace was taken by inserting an optical splitter on the interior Gigabit Ethernet commodity interface of a Juniper M40. The optical split was then attached to a PIII 800 dual processor system with 1 GB RAM running FreeBSD 4.6 with 2 Syskonnect 9843 Gigabit Ethernet cards on a 64bit PCI bus. The flow data was captured using CoralReef's [13] `crl_flow` with all flows being expired in 30 second intervals. Each trace interval captured more than one million packets and one hundred thousand flows. Packet loss during the captures averaged under ten thousand packets per interval or less than one percent.

BGP data was collected by establishing a session with the local route server using the Zebra [2] BGP routing daemon. A full dump of the table was taken on 17 April 2003 and all BGP updates stored for later integration. This method should provide highly accurate BGP information for any arbitrary time during the capture session.

The user community consists of about 100,000 students and an additional 25,000 thousand faculty and staff members. While all university members do have access to the connectivity provided by PSC, some member universities have purchased independent connectivity for the residence halls.

### 3.3.3 Post-processing

Having relied on flow-level collection tools, it was necessary to route the flows using the BGP data so that the destination IP's were correctly attributed to the BGP best-matching prefix. To accomplish this, we read the traffic flow statistics and maintained the BGP routing information synchronized with the starting timestamps of the flows. The goal is to determine at each flow if the starting timestamp of the next flow is after the next BGP update. If the next BGP update happens before the commencement of that flow we apply this update and iterate until a BGP update is encountered that has occurred after the start of the next flow. Once the BGP information is updated, we find the best matching prefix in the BGP routing table and add this information to the flow data in addition to the corresponding AS path information.

Because the BGP routing information only deals with outbound traffic, we used the best-matching prefix and AS path information only for outbound traffic. Due to routing policy AS paths are often asymmetrical in that the path followed by incoming IP packets need not be the same as the AS path for this IP destination [159, 80].

## 3.4 Stability of the interdomain paths

Any study of the topological stability of Internet traffic depends on the premise that the BGP advertised interdomain topology itself be stable. Therefore we first need to explore the stability of the AS paths of the BGP routes. To do this we rely exclusively on the traces and BGP data collected from UCL. The month long trace provides the necessary depth of information which the PSC trace of 24 hours simply cannot.

Figure 3.2 compares the number of days any distinct AS path was during the UCL trace for all AS paths and paths where at least one byte of data was seen over that month. During this period 103,853 distinct AS paths were seen in the BGP advertisements but only 31,151 carried any traffic at all. Due to BGP dynamics more than fifty percent of the distinct AS paths were present in the BGP routing table for less than nine minutes. On the other hand, more than 42 percent of the AS paths over which traffic was sent were found in the BGP routing table for 99 percent of the duration of the month. These stable AS paths having traffic represented slightly more than 13 percent of all AS paths seen over the month. These results are similar to the [139] study which showed that most BGP update events occur for a small fraction of the prefixes. [150] studied the lifetime of triples <prefix,source AS, destination AS> also confirms our observations by demonstrating that the majority of these triples have a single AS path present in the routing tables over long periods. [182] also showed that the primary AS path for a DNS A root server lasted for long time periods.

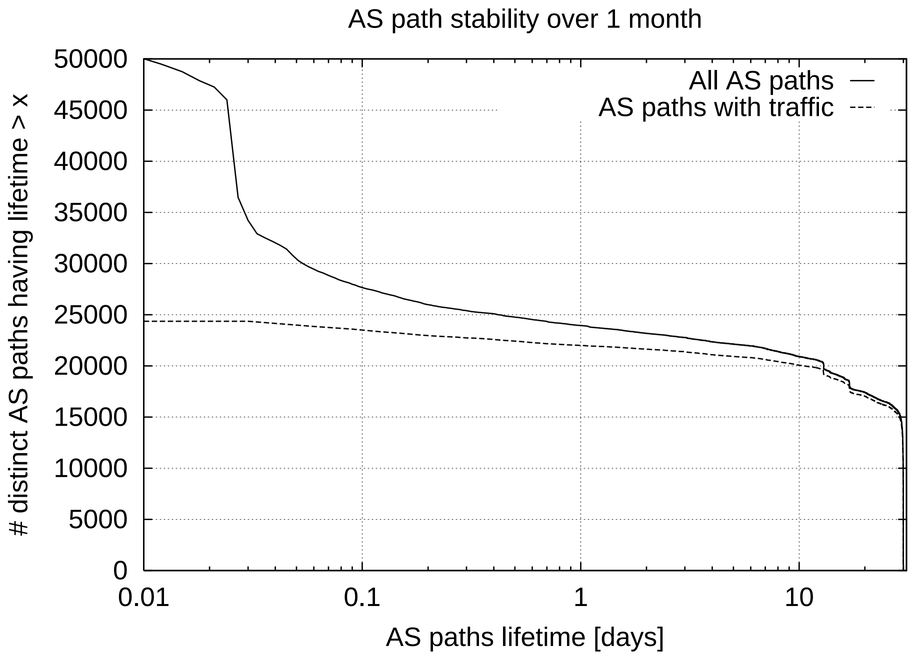



Figure 3.2: Stability of AS paths.

Figure 3.2 provides an overview of the stability of the AS paths but without respect to the traffic they see. We are more interested in whether the AS paths which carry the majority of the traffic are stable or not. Figure 3.3 compares the longevity in presence of the AS paths and their associated traffic over time. We define “presence” as the number of seconds during which an AS path was found among the BGP best routes over the one month of the trace. The curve labeled “AS paths” on Figure 3.3 gives the percentage of the total time AS paths were present in the BGP routing table. The curve labeled “traffic” shows the percentage of traffic for AS paths present for more than some percentage of the one month trace period. The latter curve shows that more than 95 percent of the total traffic was carried by AS paths that were present in the BGP routing table more than 99 percent of the time. In contrast, only 42 percent of the AS paths having traffic were present for more than 99 percent of the time in the BGP routing table. Our results are similar to those of [139] that showed that a few popular web sites for AT&T traffic had stable BGP routes. Our results however are more precise and detailed than [139] since we consider all prefixes in the traffic, not a subset of them. The prefixes for the largest fraction of the traffic of our stub AS have a single AS path that lasts for more than 99 percent of the one month period of our trace.

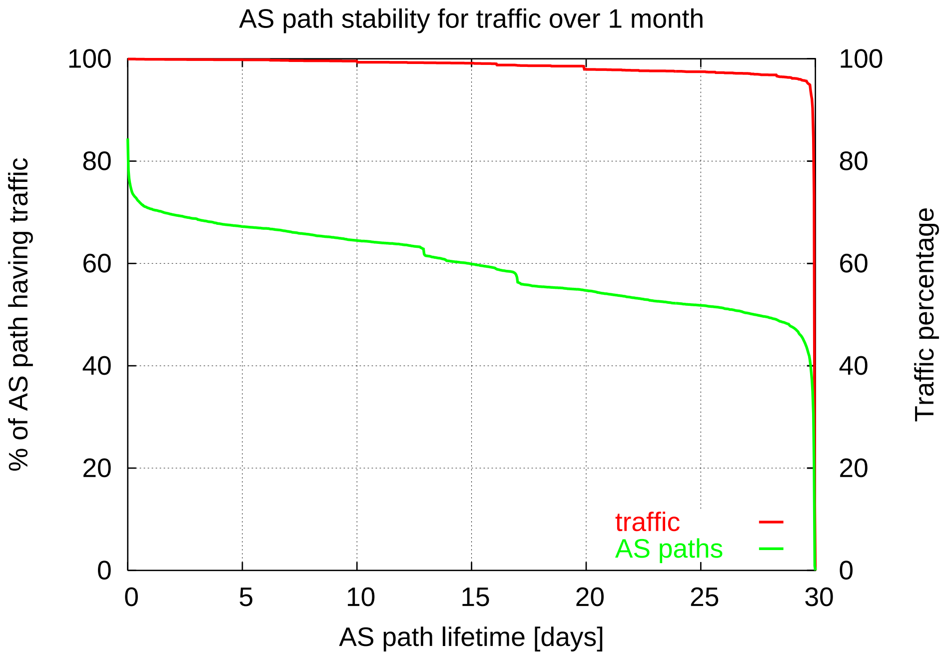



Figure 3.3: Stability of AS paths with traffic.

BGP hence does not significantly interfere with the traffic dynamics. The interdomain topology used to carry outbound traffic can thus be considered as stable even over a long time period as one month.

## 3.5 Topological distribution of interdomain traffic

Before looking at the dynamics of the traffic on the AS-level topology, it is good to build an understanding of how does the traffic gets aggregated on the AS-level topology. The definition of an "edge" serves this particular purpose. In traffic engineering, a primary concern is the size of the interdomain topology for the majority of the traffic. The question we pose in this section is the following: how large is the the AS-level graph which represents an arbitrary percentage of the total traffic exchanged with remote ASes ?

First, let us look at how much traffic traverses an AS path over the one month for UCL and the studied day for PSC. Figure 3.4 presents the cumulative traffic percentage carried by an increasing number of AS paths over the whole of the UCL and PSC traces. Figure 3.4 shows that the 20 largest AS paths carry slightly more than 50 percent of the total traffic for UCL and 40 percent for PSC. Additionally, we see that for the $90^{th}$ percentile of the UCL traffic requires about 200 AS paths while PSC needs more than 400 AS paths. The cumulative distribution of the traffic for AS paths shown here is similar to the results of [174, 137] for interdomain traffic sources and destinations (prefixes and ASes).

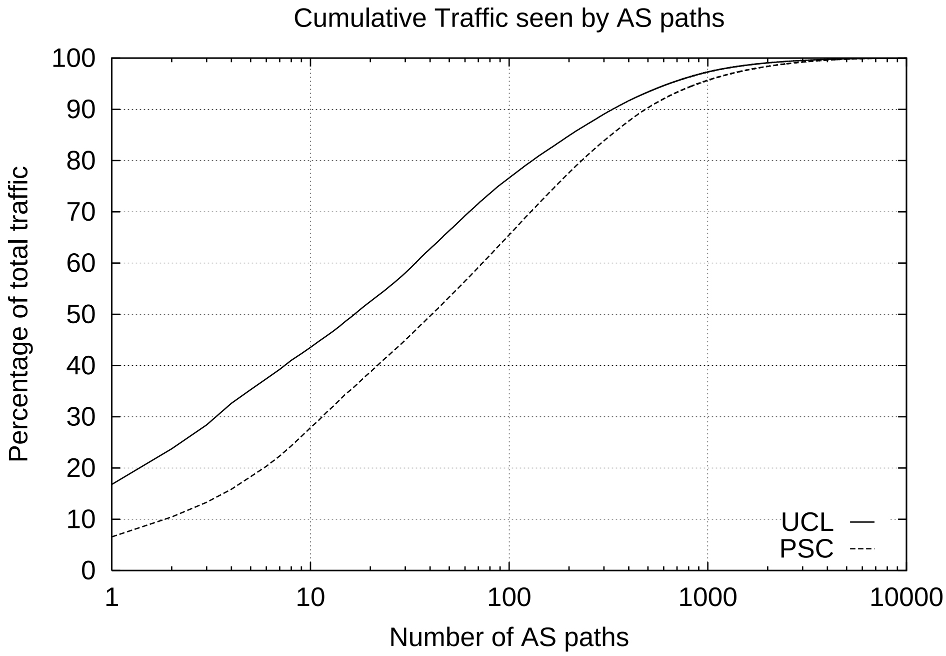



Figure 3.4: Cumulative traffic captured by AS paths.

Now let us determine how many "edges" carry the greatest portion of the observed traffic. For this, we start from the complete AS-level topology for all AS paths that carried traffic, and remove all edges that fall below as specified traffic percentage. Figure 3.5 provides the pseudo-code to compute the number of edges with more than $x$ percent of the total traffic. In the pseudo-code of Figure 3.5, we first attribute the traffic to the appropriate edges for each AS path (lines 2 to 12) in the `Edge`

```
1 // Sum up traffic for transit edges
2 foreach AS_PATH k {
3   // Sum up traffic to know the total
4   Tot_traffic+=k.traffic
5   // Add the traffic for each edge of the AS path
6   i ← 1
7   while (i < (k.as_path_length-1)) {
8       // Add AS path traffic for the current edge
9       Edge[k.as[i]][k.as[i+1]]+= k.traffic
10      i++
11  } // end while
12 } // end foreach
13 // Sort transit edges by increasing amount of traffic
14 Sorted_edges_traffic ← sort_increase(Edge)
15 // Determine how many edges are left for a trigger of x% of the traffic
16 // Initialize edge counter
16 left_edges ← size(Sorted_edges_traffic)
19 while ((Sorted_edges[left_edges]*100/Tot_traffic) < x) {
20  // This edge has less than x % of traffic, prune it
21  left_edges-
22 } // end while
23 // left_edges transit edges seeing at least x % of traffic are left
```

Figure 3.5: Pseudo-code to remove edges seeing less than $x$ percent of the total traffic volume.

array. We then sort the transit edges by byte count in ascending order (line 14). This result is stored in the array `Sorted_edges_traffic` and finally we exclude the edges having less traffic than the $x$ percent threshold. The resulting value is the number of transit edges with more than $x$ percent of the traffic and is assigned to `left_edges`.

Figure 3.6 gives the results of the above code for both the UCL and PSC datasets. The curves of Figure 3.6 show the number of edges and AS paths remaining in the AS-level topology, after imposing the constraint on the amount of traffic they carry. A total of 22,352 edges and 31,151 AS paths were seen for UCL and 15,961 edges and 17,263 AS paths for PSC. The effect of increasing the traffic percentage threshold reduces the number of edges and AS paths in what seems to follow a long-tailed distribution, namely a distribution whose tail does not fall like an exponential for large values of the threshold. Only for the largest fifty or so edges and AS paths in terms of the carried traffic do we see a discontinuous behavior of the curves, for threshold values larger than about half a percent of the total traffic. For less than fifty transit edges and AS paths, the horizontal distance between the two curves increases. An increase in the horizontal distance between the curves of edges and AS paths means that for that particular number of remaining edges and AS paths, one of two traffic aggregates requires a larger threshold than the other. Because edges aggregate the traffic from potentially several AS paths, the "edges" curve is always on the right of the "AS paths" curve on Figure 3.6. We know from Figure 3.4 that these fifty largest AS paths are indeed carrying the majority of the traffic, fifty percent for PSC and sixty-six percent for UCL.

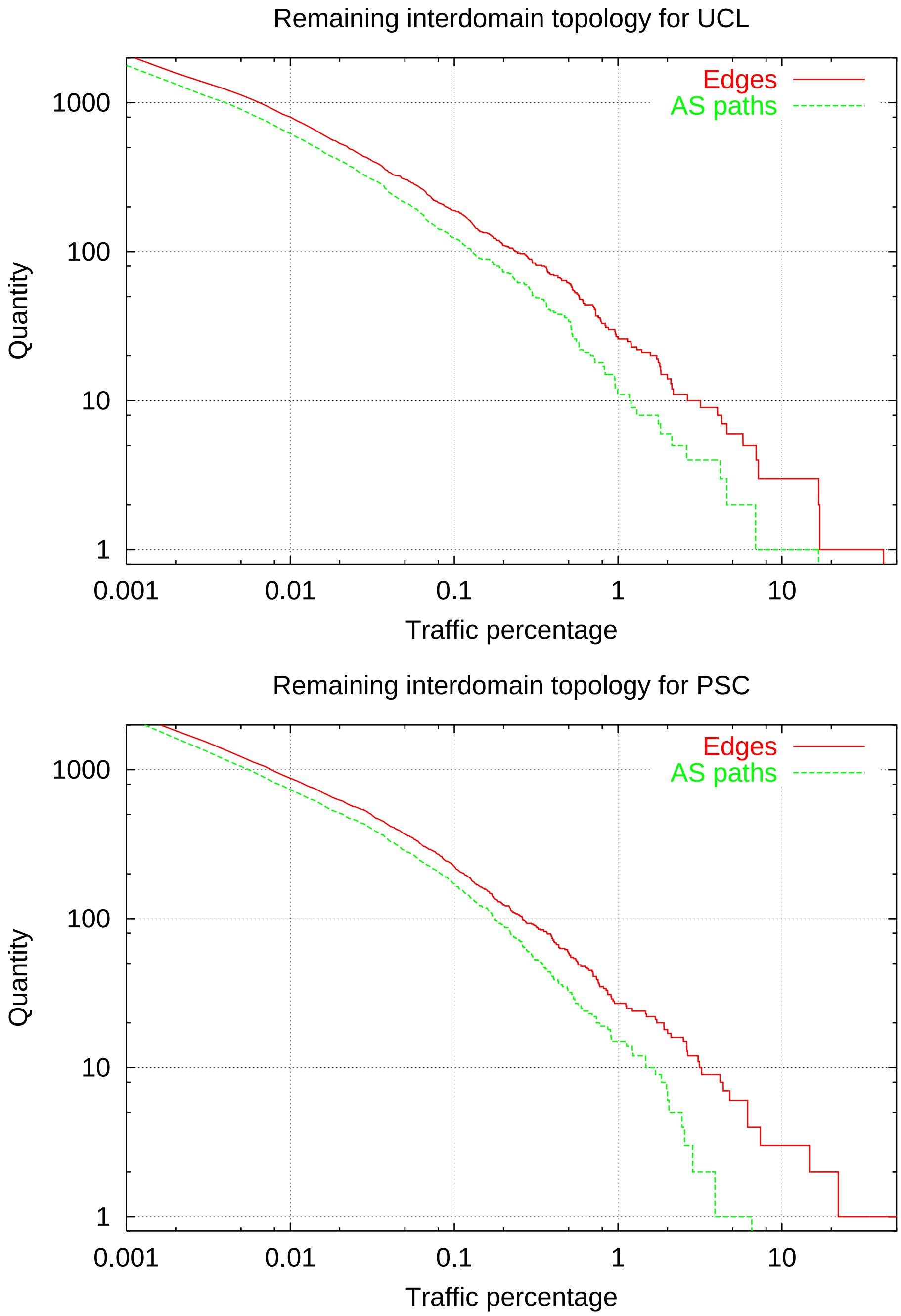



Figure 3.6: Effect of trigger for pruning the interdomain topology: UCL (top) and PSC (bottom).

Looking closely at the tail of the curves of Figure 3.4 shows a marked difference between UCL and PSC in that the horizontal distance between the UCL's AS paths and edges curves does not grow as rapidly or as much as PSC's. The reason lies in the way traffic aggregation occurs for these largest edges for these two stubs. UCL has a significant percentage of its traffic with direct peers of its provider, while all the PSC traffic has all its traffic that crosses tier-1 providers. A significant traffic aggregation occurs for the few largest edges of PSC. Figure 3.4 proves this point by providing the cumulative traffic percentage carried by edges at each AS hop distance. So we take all edges at some AS hop distance and count how many bytes have been carried by these edges. We then sort the edges at this AS hop distance by decreasing byte count and plot the cumulative percentage of traffic carried by the largest $n$ edges at this AS hop distance from the local AS. The curves for an AS hop distance of one on Figure 3.7 illustrate the main difference between UCL and PSC. The largest direct peer of UCL's provider sees only a little more than forty percent of the total traffic, while the largest peer of PSC sees a little less than eighty percent of the total traffic. UCL's provider has many direct peers with whom it exchanges traffic at local interconnection points. The PSC trace on the other hand shows that the two largest direct peers see all the traffic, simply because the PSC trace contains only commodity traffic that is sent through its commercial providers.

The curves of Figure 3.7 for an AS hop distance of two and three provide us with the interesting information about traffic aggregation on the AS-level topology. Obviously, traffic aggregation occurs on the edges with the direct peers. However, traffic aggregation is desirable farther in the topology because traffic engineering techniques are due to perform finer operations than moving all the traffic from one peering to another. Traffic engineering will have to work on traffic aggregates to allow an AS to control the amount of traffic that crosses the links with its direct peers. This is why it is important to know how traffic is splitted a few AS hops away in the AS-level topology. Figure 3.7 tells us that if we were to influence the largest ten edges, then we would be able to control about 35 percent of the total traffic for UCL and a little less than 60 percent for PSC. This information is interesting for traffic engineering since we would be able to influence a large fraction of the total traffic by tweaking a few edges. If we need to control for instance ninety percent of the total traffic to perform fine-grained traffic engineering, then we need to rely on edges at two AS hops or more because working on direct peers will move too large amounts of traffic. This means that we would then need to tweak in the order of hundreds of edges due to the lack of traffic aggregation beyond direct peers. For PSC, controlling 90 percent of the traffic by tweaking edges at two AS hops requires to influence more than one hundred edges. For UCL, there is a lot of traffic with direct peers so that edges two AS hops away see only more than 70 percent of the total traffic. Had UCL not that much traffic with peers at public interconnection points, it would show a similar topological aggregation to the one of PSC.

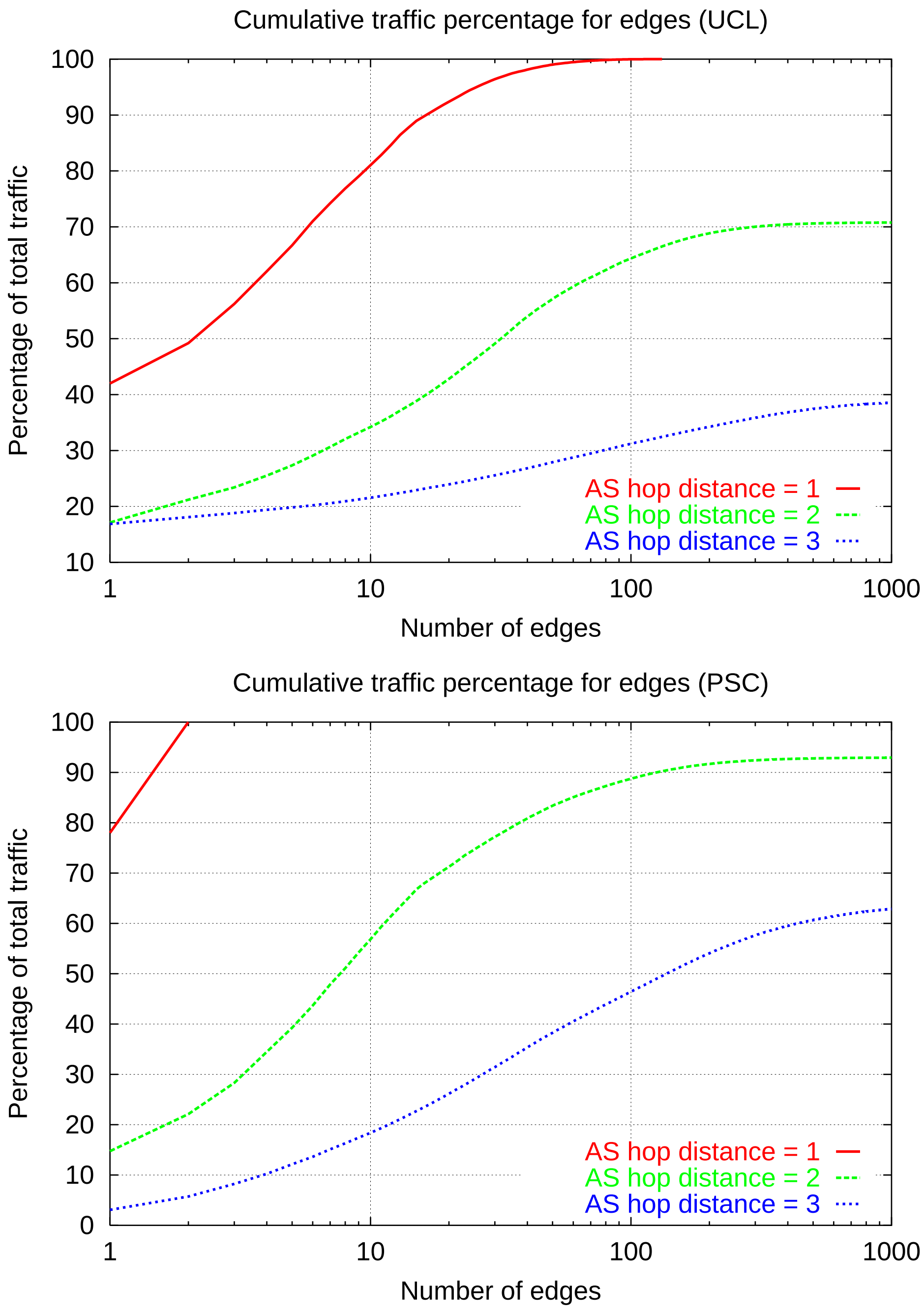



Figure 3.7: Cumulative traffic carried by edges.

## 3.6 Short-term stability of interdomain traffic

The previous two sections examined the properties of the traffic on the AS-level topology over the entire duration of the measurements. Another important aspect is the stability of the traffic over smaller timescales. Understanding at which timescale the AS paths are stable is of interest for interdomain traffic engineering. Therefore in this section we outline the connection between the large timescale features of the traffic on the AS-level topology and smaller timescales. The analysis in this section is based solely on the UCL dataset and we rely on the PSC dataset for confirmation of our results. Furthermore, we will not use edges in the rest of this chapter. The reason for using AS paths only is that there is no substantial difference between AS paths and edges as far as traffic aggregation is concerned (see section 3.5). We will therefore use the more familiar notion of AS path.

### 3.6.1 Short-term evolution of top AS paths

We start with an analysis of the dynamics of what we call the top AS paths. The "top AS paths" are nothing more than the AS paths sorted by decreasing amount of traffic over the considered time period (hour, day or month). For instance, the top fifty percent AS paths for some hour is the smallest set of AS paths that carry fifty percent of the total traffic during that particular hour.

Figure 3.8 shows the hourly evolution of the number of AS paths present in the top $x$ percent AS paths for four different values of $x$. To create this graph we computed the amount of traffic seen by each AS path for each hour, sorted them by decreasing amount of the total traffic, computed the cumulative traffic distribution and finally plotted the hourly evolution of the number of AS paths for various values of the traffic percentage every hour. Each point of the curves shows how many AS paths are necessary to carry a specified percentage of the total traffic during a given hour.

The use of a logarithmic scale for the y-axis reduces the visual importance of the dramatic increase in the number of AS paths needed as the percentage of traffic increases. Capturing fifty percent of the traffic every hour requires between five and ten AS paths, while ninety percent requires one hundred AS paths, and capturing all the traffic demands as many as five thousand AS paths. These numbers are smaller than the ones we have seen on Figure 3.4 for the cumulative percentage carried by the largest AS paths. On the timescale of hours, large AS paths thus see proportionately a larger fraction of the traffic than over one month, the latter needing twenty AS paths for fifty percent of the traffic and more than two hundred AS paths for ninety percent of the traffic. Large AS paths are thus larger as seen over smaller timescales than over larger timescales.

The hourly evolution of the number of AS paths for the traffic percentage seems to follow a daily periodicity similar to the one of the total traffic evolution. The larger number of AS paths seen during the busiest hours of the day is likely to be due to

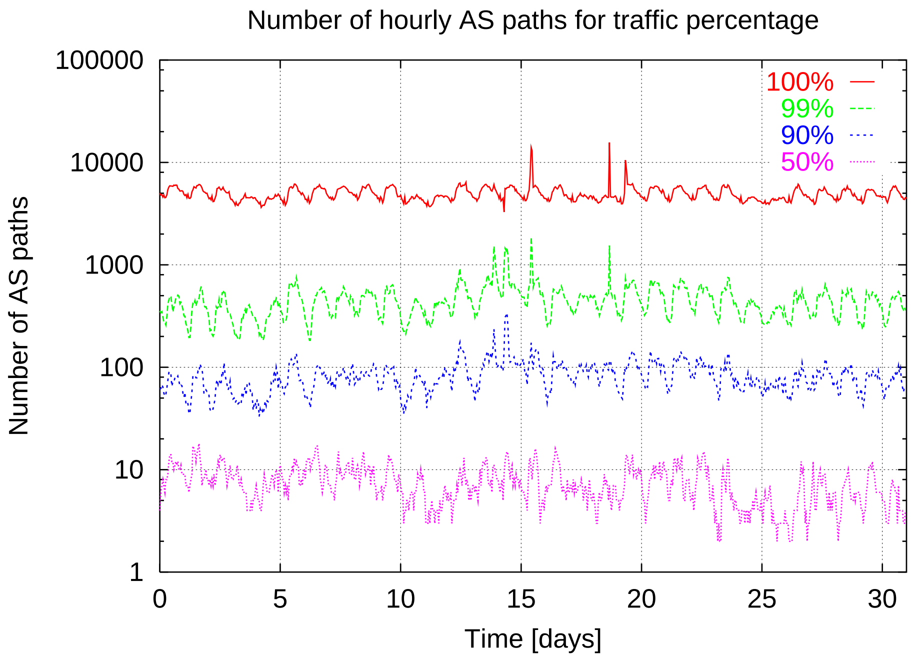



Figure 3.8: Hourly evolution of the number of top AS paths.

a larger number of IP addresses. Note however that for fifty percent of the hourly traffic, this daily periodicity is less evident because the number of AS paths is small.

### 3.6.2 Intersection of hourly and monthly top AS paths

What the above results do not tell is whether the AS paths seen each hour are the same ones that are seen over the whole month. Without this information, it is impossible to determine the actual stability of the traffic on the AS-level topology. We must therefore compare the individual AS paths over both the monthly and hourly timescales to ascertain any degree of overlap they might have.

The overlap between the hourly and monthly AS path statistics can be seen in Figure 3.9. Figure 3.9 shows three curves. The first one, labeled "# top 90 % AS paths (over 1 month)", provides the number of top AS paths that carry ninety percent of the traffic over the whole month. 330 AS paths are required to capture ninety percent of the traffic over the whole month. The second curve of Figure 3.9, labeled "# top 90 % AS paths (hourly)", provides the hourly evolution of the number of AS paths representing ninety percent of the traffic every hour. This curve is the same as the curve labeled "90 %" on Figure 3.8. The third curve, labeled "intersection", shows the number of AS paths among the top ninety percent hourly AS paths that are also in the top ninety percent monthly AS paths. The absence of significant difference between the second and the third curve indicates that, at least in terms of the number of AS paths, there is an important overlap between important AS paths in the short and the long term.

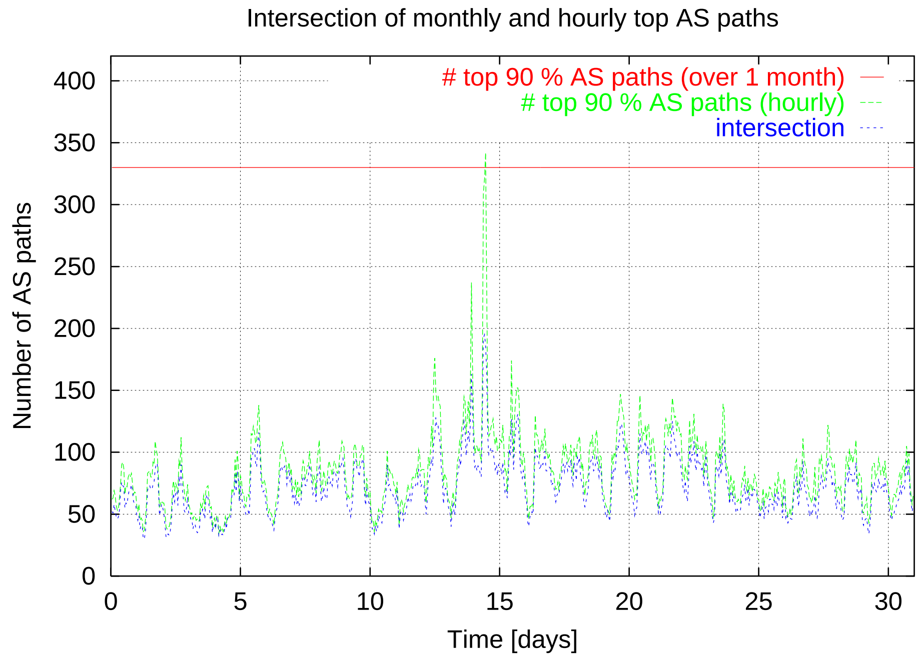



Figure 3.9: Evolution of intersection between hourly and monthly top AS paths.

An important overlap in number between the top AS paths over the month and the top AS paths over hours does not mean that the overlap in traffic carried is important. The top curve of Figure 3.10, labeled “hourly capture of monthly top 90 % AS paths”, shows the hourly evolution of the largest AS paths that carry ninety percent of the total monthly traffic. This curve shows that if we could guess what are the 330 most important AS paths over the forthcoming month, then we could control about ninety percent of the hourly traffic. The middle curve of Figure 3.10, labeled “hourly capture of 90 % intersection”, shows the hourly percentage of traffic carried by the AS paths that are among the top ninety percent AS paths both monthly and hourly. This second curve shows that the top monthly AS paths capture a fair amount of traffic on a hourly basis. Finally, the bottom curve of Figure 3.10, labeled “monthly capture of 90 % intersection”, shows the percentage of the monthly traffic carried by the hourly top 90 % AS paths. This bottom curve shows that the AS paths that are among the hourly top 90 % AS paths capture only between fifty and seventy percent of the monthly traffic. Hence the largest AS paths on a hourly basis are not the same as the largest AS paths on a monthly basis.

Figure 3.10 tells us that assuming that we could know which AS paths are important over long time intervals in advance, we could include them and have a good certainty that this set of AS paths will capture a fair amount of the traffic in the future. However, we cannot make this assumption. Even though 330 AS paths capture ninety percent of the traffic over a one month period, less than one hundred are active on a hourly basis. Therefore, to capture ninety percent of traffic with a large probability of success, we would have to take into account more than three times as many AS paths as required.

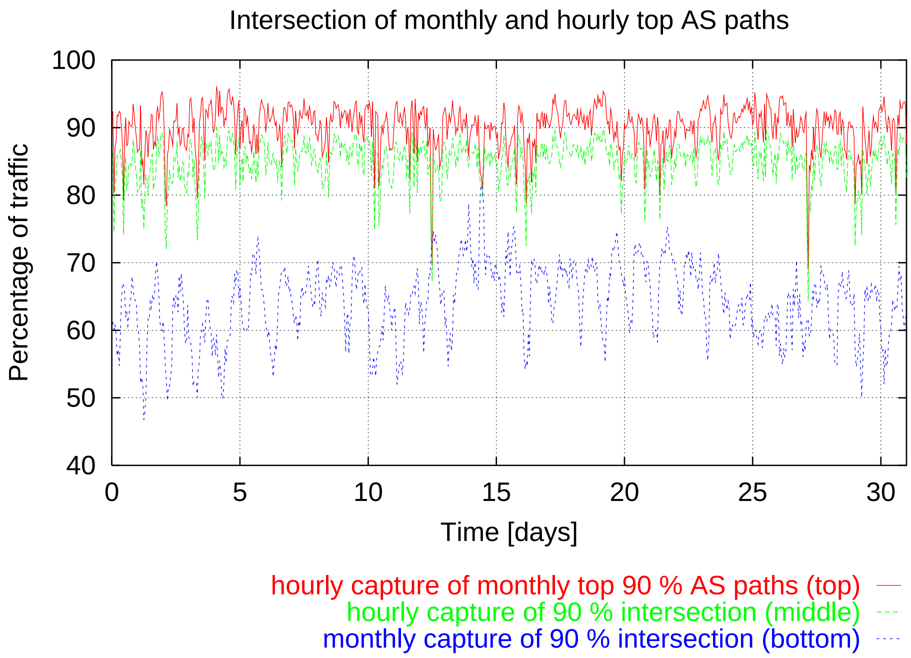



Figure 3.10: Evolution of traffic captured by intersection of top AS paths.

### 3.6.3 Presence of short-term top AS paths

The previous section showed that important hourly AS paths have much in common with the monthly top AS paths. In practice we do not have such knowledge of which AS paths will carry large amounts of traffic in the future. This section studies for how much time most AS paths are among the hourly top AS paths. For this, we compute for each hour the top ninety percent AS paths and count how many times the AS paths reappear in the top ninety percent over the length of the trace.

Figure 3.11 counts the number of hours over the month that each AS path appears in the top ninety percent AS paths. We also compute for each of these AS paths the amount of traffic it captures during the hours it is among the top ninety percent AS paths. The curve labeled "AS paths percentage" gives the cumulative percentage of the AS paths which have ever appeared among the top ninety percent in terms of the number of hours it was present, hence the name "presence". The other curve labeled "traffic percentage", gives the corresponding cumulative percentage of the total traffic carried by the previous AS paths during a given number of hours over the month. A total of 2139 AS paths were seen among the hourly top ninety percent AS paths. 626 of them were seen only once while just six were seen during all time periods. The curve labeled "AS paths percentage" shows that most AS paths were seen for a small fraction of the time. More than 96 percent of these AS paths were seen for less than one third of the time and represent about twenty percent of the total traffic during the whole month. Additionally, fifty percent of the total traffic appears for AS paths seen among the hourly top ninety percent for about eighty percent of the time. The six AS paths that are present during all time periods carry about 36 percent of the total traffic which is a significant portion. However, more

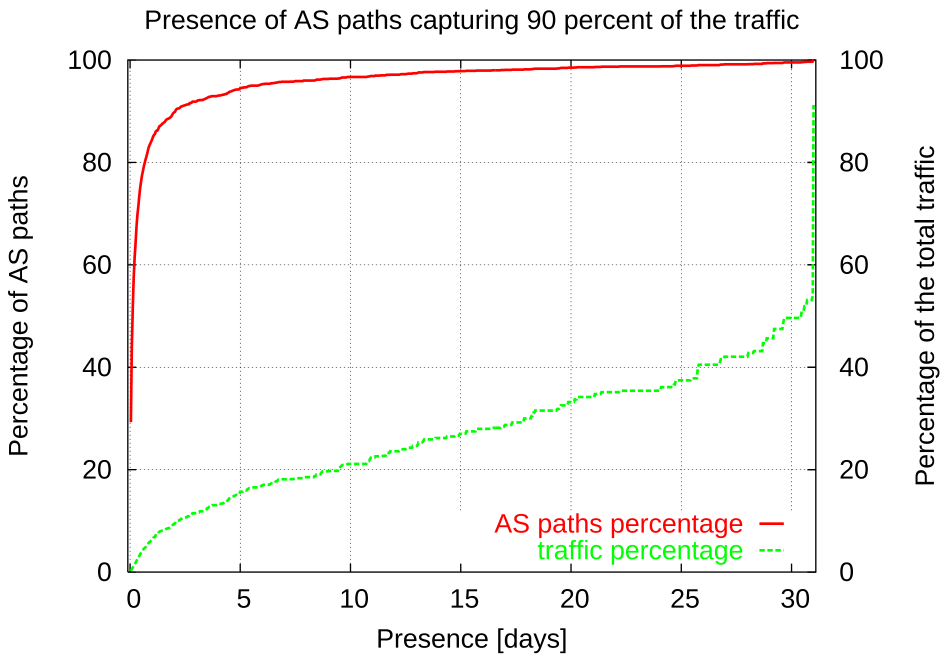



Figure 3.11: Presence of hourly top ninety percent AS paths.

than twenty percent of the traffic appears for AS paths that are seen in less than one third of the time periods during the month. This leaves about 25 percent of the traffic being carried by AS paths that are moderately but not entirely stable.

The relatively limited presence of the top ninety AS paths might be due to the fact that most of them carry too limited an amount of traffic. The presence of larger AS paths might be better than what Figure 3.11 showed. Figure 3.12 shows the same information as Figure 3.11 but for the top fifty percent top AS paths. 360 AS paths were seen among the hourly top fifty percent AS paths. One third of these AS paths were present during only two hour intervals over the one month among the hourly top fifty percent top AS paths. Slightly less than 90 percent of these 360 AS paths were present during 24 one hour intervals over the month. Among these 360 AS paths, those that are present for less than one third of the hourly intervals carry a little less than 20 percent of the total traffic. AS paths that are present more than half of the hourly intervals hence carry 30 percent of the total traffic. This shows that even the largest AS paths in terms of the hourly traffic have a high variability in presence. This variability in presence of the AS paths thus concerns all AS paths, irrespective of the amount of traffic they carry.

This section showed that while a few AS paths can be considered as stable over the whole month, there is a significant fraction of the total traffic for unstable AS paths among the largest AS paths in terms of traffic over the short term.

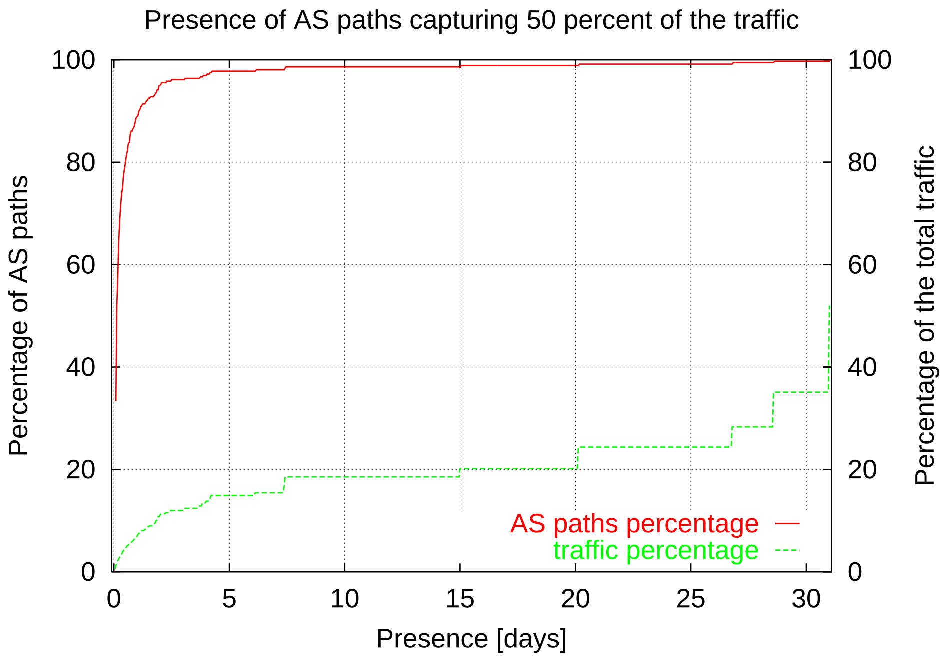



Figure 3.12: Presence of hourly top fifty percent AS paths.

### 3.6.4 Traffic Capture for top AS paths

To this point we have assumed foreknowledge of which AS paths carry the largest portion of the traffic. Under that assumption it is easy to show that a significant percentage of the hourly traffic travels AS paths that are prominent in the monthly traffic statistics. However, we have shown in Figure 3.11 that while some of the hourly top ninety percent AS paths are stable the majority are moderately to highly unstable and do account for a significant part of the traffic. This means that knowledge of AS paths with a large amount of traffic over short times scales will not necessarily give an indication of the AS-level topology for the traffic over longer time scales.

Figure 3.13 shows the hourly percentage of traffic captured by the top thirty, fifty, and ninety percent AS paths over the length of the one month of the UCL trace. We first determined the AS paths that carried the most traffic over the month and then plotted the time evolution of the hourly traffic that these top AS paths represent. If these paths were stable we would expect to see a relatively flat line with little variation over time. However we see that the AS paths that carry thirty percent of the monthly traffic rarely carry thirty percent of the hourly traffic. Instead it displays large variations in the hourly traffic, generally between twenty and forty percent. We see similar variation as we include more AS paths to increase the percentage of traffic although the absolute level of variation decreases as we include more paths. Figure 3.13 shows that even though the largest AS paths in traffic over the one month capture a large fraction of the traffic on a hourly basis, the traffic percentage these AS paths capture every hour varies a lot, particularly for the largest AS paths in traffic.

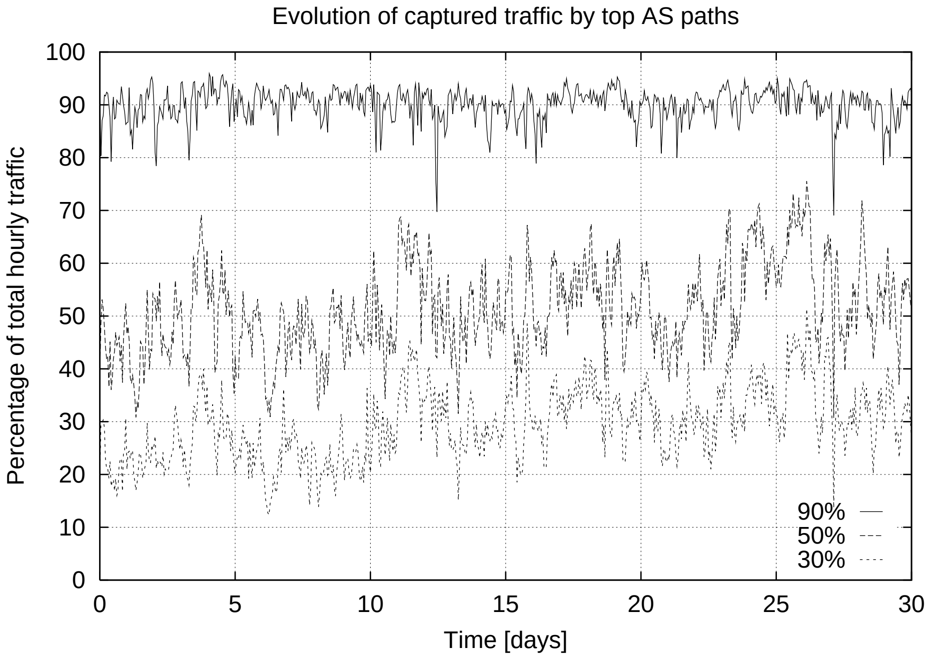



Figure 3.13: Evolution of percentage of traffic captured by top AS paths.

It is possible that windowing the previous traffic periods might give some insight into the stability of AS paths in subsequent time periods. The results of this inquiry can be seen in Figure 3.14 where the average percentage of traffic seen by fifty percent and ninety percent AS paths during time windows of hours and days is given. For each hour we first determined the AS paths that carry fifty percent and ninety percent of the traffic during the previous time window, and computed how much traffic these ASes carry over the next hour. Finally, the average over the length of the trace of the percentage of traffic seen for each time window length is determined. The same process was then repeated for time windows of days. The results of this can be seen in the top graph of Figure 3.14 along with the standard deviation (bottom of Figure 3.14).

The top of Figure 3.14 shows that increasing the length of the the time window increases the average percentage of traffic captured. However, the gains for larger time windows are limited. This is especially true for increasing the time window to periods of multiple days. However, if we compare the standard deviations of the hours window and the days window we discover a notable difference. The standard deviation for all time windows and the AS paths capturing fifty percent over the time window is larger than for the ninety percent AS paths. The smaller number of AS paths for the fifty percent capture are responsible for this since the smaller set of AS paths accounts for a comparatively larger portion of the total traffic. The smaller sets of AS paths are thus more vulnerable to random traffic surges.

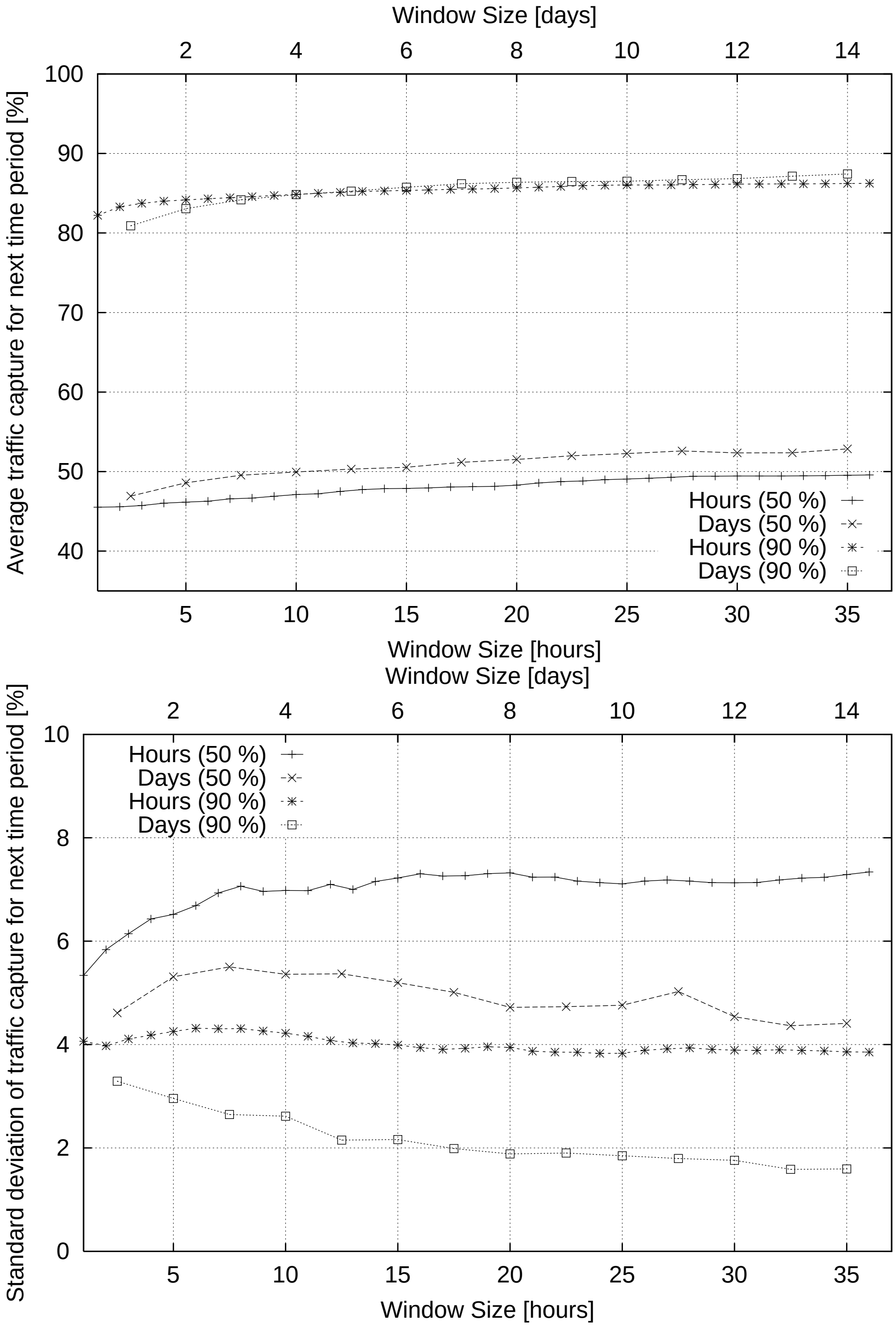



Figure 3.14: Traffic capture for windowed 50 % and 90 % top AS paths: average (top) and standard deviation (bottom).

## 3.7 Implications on interdomain traffic engineering

Most studies on interdomain traffic insist that a few sources and destinations generate most of the interdomain traffic. This optimistic news is misleading because it hides an equally important facet of the network topology: influencing most of the interdomain traffic requires that a significant number of ASes be considered – far more than previously believed. For instance, in section 3.6.1 it was shown that to control 50 % of its traffic UCL would have to tweak about 10 AS paths every hour, or about 100 AS paths every hour to control 90 % of its traffic. While these numbers might not seem particularly large, making the necessary adjustments to BGP on a nearly continuous basis is not a trivial task.

This is not to say it is impossible though. For outbound traffic, tweaking BGP is much easier and does not impact BGP routing outside the local domain of the stub AS. For inbound traffic things are a little more intricate. First, the AS path information contained in the BGP routing table of the local stub does not provide the right information concerning the inbound path followed by IP packets [159, 80]. Second, even if the local stub had the whole map of the AS-level Internet, which is a very strong assumption indeed, there is no guarantee that influencing remote ASes located a few AS hops away would be possible given that policy routing is shaped independently by each AS. Third, even if there was some manner in which this could be done, it would mean that a stub would need to frequently change its BGP announcements for potentially hundreds of ASes. If a significant percentage of the stub ASes begin to perform that kind of BGP tweaking, it is difficult to predict the global effect on BGP routing, which has already been demonstrated to suffer from convergence problems [14, 103, 86].

Additionally, we showed in section 3.6 that the variability of the AS paths that carry the largest amounts of traffic is quite important. The uncertainty due to the traffic dynamics implies that controlling the interdomain traffic requires to influence more AS paths than what is believed based on an a priori knowledge of the important AS paths in the traffic. The topological dynamics of the interdomain traffic indicates that the only way to improve the certainty of influencing a large fraction of the traffic is to tweak more AS paths. To do this, one needs to influence more BGP routes, which in turn will potentially affect traffic that spans a larger fraction of the topology of the Internet.

Finally, tweaking BGP will impact on the intradomain traffic distribution of large ISPs [8]. Solutions that rely on intradomain routing to optimize the intradomain traffic of large ISPs [76] can be affected by the tweaking of BGP routing. The interactions between intradomain and interdomain traffic engineering thus need to be studied before deploying interdomain traffic engineering tools.

## 3.8 Conclusion

In this chapter, we have contributed to the understanding of the dynamics of the interdomain traffic on the AS-level topology for stub ASes. This was reached through the analysis of two interdomain traffic traces from two stub ASes so as to study the dynamics of the interdomain traffic on the AS-level topology.

This analysis revealed several properties of the interdomain traffic. First we showed that interdomain paths are stable from a routing viewpoint. Second, we showed that while a few AS paths were responsible for a large fraction of the interdomain traffic, capturing a large fraction of the total traffic requires several hundred. We then identified a transit core that aggregated large amounts of traffic on the interdomain topology and that a small number of transit edges carry a large fraction of the interdomain traffic. Next, a study of the dynamics of AS paths needed to carry a given percentage of traffic demonstrated that the behavior of high traffic top AS paths is significant. However, it also showed that the stability of top AS paths over long time scales did not correlate to stability of these same paths on shorter time scales. In fact, we showed that the commonly expected stability of the AS paths did not actually closely coincide with observation. Lastly, the problem of capturing a given amount of traffic based on the previous behavior of specific AS paths was shown to require observational timescales on the order of days in order to limit short term variability. The only solution which limits this short term variability is to consider larger percentages of the total traffic on shorter timescales and a larger portion of the interdomain topology.

# Part III

# Interdomain traffic optimization with BGP

In this third part of the thesis, we wish to evaluate the feasibility of interdomain traffic engineering based on the tweaking of the BGP protocol. Chapter 4 is mainly introductory, in that it provides a first step into the evaluation of the problem of a multi-homed stub AS wishing to minimize the billing cost of its outgoing traffic. Chapter 4 also introduces the evolutionary meta-heuristics. Then, Chapter 5 takes the problem introduced in Chapter 4 but where the objective is to limit the number of BGP filters to be used to attain an improvement in a single traffic objective. Chapter 6 studies the on-line optimization of a traffic balancing objective under constraint of minimizing the burden on BGP. Chapter 7 finally considers two traffic objectives defined on different timescales, also under contraint of minimizing the number of changes in BGP. A large part of Chapter 5 has been published in [176], Chapter 7 has been submitted as [171] and Chapter 6 as [175].

# Chapter 4

# Interdomain traffic engineering with BGP: an Evolutionary Perspective

In this chapter, we propose to utilize evolutionary optimization to perform traffic engineering and evaluate interdomain traffic engineering with BGP. We look at the the problem of optimizing a daily cost function defined on the interdomain traffic to understand the issues of tweaking BGP to controlling the flow of the interdomain traffic.

This chapter is structured as follows. Section 4.1 introduces the problem of interdomain traffic engineering for stubs. Section 4.2 then details the nature of the single objective function to be optimized. Section 4.3 presents the different combinatorial views of the interdomain traffic engineering problem. Section 4.4 then introduces the multi-objective interdomain traffic engineering problem. Section 4.5 presents evolutionary algorithms. Section 4.6 describes our evolutionary algorithm for the single-objective problem. Section 4.8 presents the datasets we use to evaluate our algorithms. Section 4.7 shortly presents a few numerical results in the single-objective case. Section 4.9 explains how to extend the evolutionary algorithm used for the single-objective problem to the multi-objective case. Section 4.10 goes on to evaluate the evolutionary algorithm for the multi-objective problem. Finally, section 4.11 evaluates the simulation results.

## 4.1 The "simple" combinatorial optimization problem

To introduce the problem that we intend to solve, let us first provide an example illustrating the nature of the interdomain traffic engineering problem with BGP. Assume a stub AS having one link per provider. Our stub AS receives a BGP routing table from each of its providers. To control the flow of its interdomain traffic, our AS can only rely on the information found in these BGP routing tables.

Figure 4.1 provides an example topology for a stub AS connected to three providers. Each provider advertises one route towards each of the three considered destinations.

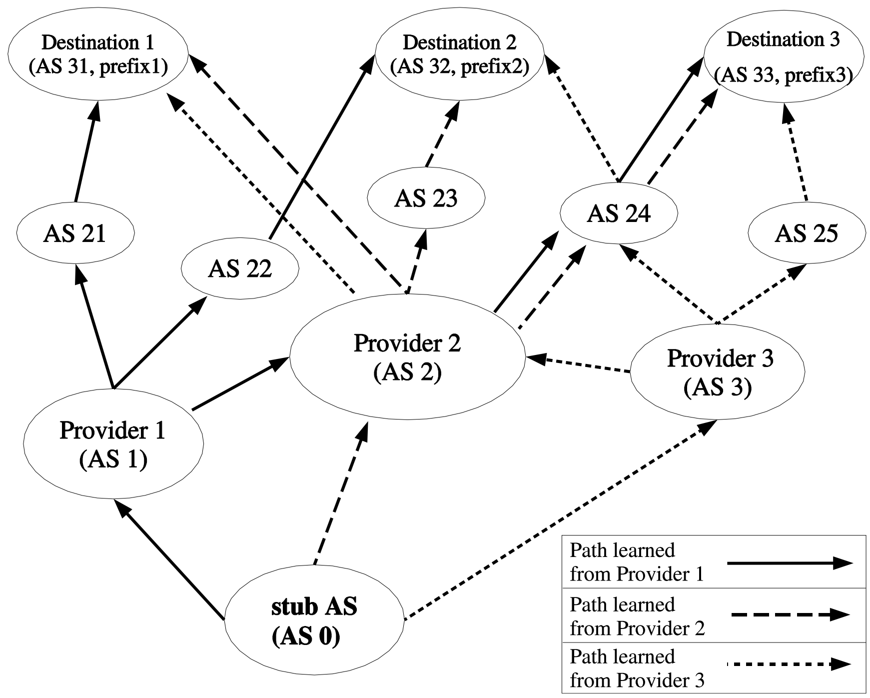



Figure 4.1: Example AS-level topology.

In this figure, an arrow from AS X to AS Y indicates that the pattern "X Y" appears in the AS path for some destination. In addition, we use three different arrow styles to identify each provider's routes, so that if part of the AS path of some routes that our AS received through several providers are identical, then we shall have as many arrows as there are routes from these different providers. On Figure 4.1, the links between two ASes learned through routes advertised by provider 1 are represented by continuous arrows, while the ones from provider 2 are represented with medium dashed arrows and those from provider 3 with fine dashed arrows.

With the BGP routes received from each provider, our AS knows through which providers it can reach a particular destination. For the example provided on Figure 4.1, all three destinations can be reached through any of the three providers. Assume that each of these three destinations represents one third of the outgoing traffic for our AS. We could choose to use one different provider for each destination so that its outgoing traffic is well balanced.

Table 4.1 provides the hypothetical (and simplified) BGP routing entries corresponding to the routes for the topology given in Figure 4.1. Each destination AS shown on the topology of Figure 4.1 has a network prefix corresponding to the IP addresses that can be reached in this network (first column of Table 4.1). The second column of Table 4.1 provides the AS path that the IP packets sent towards these prefixes will follow. Next is the next BGP hop used to attain the destination. The next hop is the IP address of the BGP router of the peer AS with whom the BGP session is established, not a logical name like the one we used in Table 4.1 for the sake of clarity. Finally, the fourth column of Table 4.1 provides the value of the `local-pref` attribute of the routes.

Table 4.1: Example BGP routing table.

| Prefix | AS path | next-hop | local-pref |
|---|---|---|---|
| prefix1 | 1-21-31 | provider 1 | 100 |
| prefix1 | 2-31 | provider 2 | 100 |
| prefix1 | 3-2-31 | provider 3 | 100 |
| prefix2 | 1-22-32 | provider 1 | 100 |
| prefix2 | 2-23-32 | provider 2 | 100 |
| prefix2 | 3-24-33 | provider 3 | 100 |
| prefix3 | 1-2-24-33 | provider 1 | 100 |
| prefix3 | 2-24-33 | provider 2 | 100 |
| prefix3 | 3-25-33 | provider 3 | 100 |

Now we get back to our problem of our AS wishing to distribute evenly its outgoing traffic among the three providers. Upon receiving a BGP route, an AS may modify the value of some of the BGP attributes of this route through BGP filters. It is possible to apply a BGP filter that changes the value of the `local-pref` attribute of some route so that one route for a given destination received through one particular BGP peer be preferred over other routes for the same destination received through other BGP peers. The reason why we choose to rely on the `local-pref` attribute is that this attribute is local to the AS. It is not advertised outside the domain. This means that this kind of BGP tweaking has no perverse effect on the behavior of BGP outside the local domain, at least for stub ASes. As can be seen on Table 4.1, the local AS sets to 100 the value of the `local-pref` attribute[1] of all routes through input filters. In that case of all routes having the same (and lowest possible) preference, the first rule of the BGP decision process does not play any role in the choice of the best route.

If the stub AS shown on Figure 4.1 wants to achieve a good balance of its outgoing traffic, it can rely on the `local-pref` attribute to prefer some routes over others. Recall that we assume that each destination prefix receives one third of the traffic. Suppose that the restricted BGP decision process with only `local-pref` and the AS path length is considered. If the routes towards the three destinations have a default `local-pref` value of 100, then given the fact that the shortest AS path route would be chosen by BGP to reach a destination, the traffic towards destination 1 will be forwarded through provider 2 (AS path length of 2), the traffic towards destination 2 will be forwarded through providers 1, 2 or 3 (AS path length of 3), while the traffic towards destination 3 through providers 2 or 3 (AS path length of 3). This would mean that the routes for destinations 2 and 3 are non-deterministic (for this restricted BGP decision process) since there are several "best routes" among which BGP can choose. In practice, the BGP decision process ensures that only one route is used but this would not automatically lead to a good balance of the traffic. So to get back to our problem a solution is to put a higher value of the `local-pref` attribute

[1] Another value could be chosen but this value of 100 is the default one according to [118]

in one of the routes learned from destinations 2 and 3 to ensure that only one best route remains and that the traffic balance be good among the three providers. For that purpose, our stub can attach a `local-pref` value of 110 to the route towards destination 2 learned from provider 1 and also attach a `local-pref` value of 110 to the route towards destination 3 learned from provider 3. This configuration would ensure that each provider gets one third of the outgoing traffic if the traffic is evenly distributed over the three prefixes.

## 4.2 Single-objective optimization

Let $x_i(t), i = 1, .., n$ be the amount of traffic to be sent towards $ASi$ at time $t$, and let $C_i(y), i = 1, .., p$ be the cost of sending $y$ bytes of traffic over link $i$. The objective is to find $min \sum_1^p C_i(l_i)$ under the constraints that $y_i \leq Cap_i$ and $\sum_1^p y_i = \sum_1^n x_j$ where $Cap_i$ denotes the maximum amount of bytes that can be sent over link $i$. This maximum can be the limit of the physical link capacity or the limit imposed by policing. In other words, the objective is to distribute the traffic to be sent towards all ASes over the available links so that the total cost of sending it is minimum. Note that we do not pose the problem as a combinatorial optimization problem here because we are not aimed at finding the optimal solution to the problem, but to understand the problem of tweaking BGP to optimize a cost function.

The problem of choosing what fraction of the traffic should be sent over which link under minimal global cost is not difficult to solve, whenever the cost functions are sufficiently "well-behaved". For example, if the cost to send traffic towards a destination only depends on the provider and the amount sent to that provider, not on the total amount sent for all traffic on a particular provider, then the cost function $C_i$ can be expressed as weights $c_{ij}$ so that the cost function is linear. In such a case, the problem is an instance of the assignment problem which can be solved through the Hungarian method in time complexity $O(n^3)$ where $n$ denotes the number of destinations, see chapter 11 of [125] for a detailed presentation of weighted matching problems.

In order to have an idea of the pricing schemes currently used for billing providers, we asked the question of how providers are billed for interdomain capacity on the NANOG mailing list (`nanog@merit.edu`). Here we provide a summary of the answers we received. First, most pricing schemes rely on the following procedure:

- collect samples of the traffic volume every $t$ minutes (5 and 15 minutes are common);
- combine these $t$ minutes samples into one combined sample;
- at the end of a billing cycle, compute the $95^{th}$[2] percentile of the combined samples, this number corresponds to the bandwidth $L$ which will be used for the price.

[2]The $95^{th}$ percentile is the most commonly used in practice but any other percentile could be used as well.

The computation of the $95^{th}$ percentile can be based on:

- $95^{th}$(incoming)
- $95^{th}$(incoming + outgoing)
- $95^{th}$(max(incoming,outgoing))
- max($95^{th}$(incoming),$95^{th}$(outgoing))

where `incoming` means the traffic entering the provider and `outgoing` the traffic leaving the provider. Now that we have introduced the percentile-based computation, let us describe the way billing is actually done:

1. percentile-based : $x$ USD per $y$ Mbps ($n^{th}$ percentile) with a commitment of `c` Mbps. The price per Mbps can be different for the commitment and for the traffic above the commitment (also called "burstable").
2. average-based : same as point 1 but using an average (not $50^{th}$ percentile) instead of a percentile.
3. volume-based : $x$ USD per $y$ bytes.
4. destination-based : $x$ USD per Mbps for "local" traffic (national for instance) and $y$ USD per Mbps for "non-local" traffic (international for instance).
5. max-based : flat rate based on the maximum available bandwidth, independent of how many bits are used.

It must be noted that the dynamics of the traffic can influence the value computed because for a given amount of bytes exchanged over some link, different values for the $n^{th}$ percentile can be found. We do not take into account such aspects in this chapter. Chapter 7 shall deal with more realistic traffic objectives. However, it must be noted that except for the volume-based billing scheme without any commitment and rate limit, all other common billing schemes are not linear. Solving this problem as an instance of a classical combinatorial problem is not forcibly the best way due to the complexity of the actual cost functions.

## 4.3 To group or to assign

The problem of interdomain traffic engineering as presented in section 4.1 can be seen as an instance of different combinatorial problems.

One way to view the previous problem is as an assignment problem. This problem consists in assigning the traffic $t_i$ towards each destination $i$ through one and only one provider $j$ in order to minimize the total cost of the assignment $\sum_{j=1}^{P} C_j(\sum_{i=1}^{N} x_{ji}t_i)$, where $C_j$ represents the total cost of sending traffic through provider $j$ and $x_{ji}$ tells through which provider $j$ the traffic towards destination $i$ is sent. Without constraints on how many changes to be made in the assignment and a linear cost function, this problem can be efficiently solved [125]. However, if one is interested at solving the assignment problem with constraints on the number of changes with respect to the default assignment found by BGP, then one deals with the generalized assignment problem which is known to be NP-hard [143].

The problem of interdomain traffic engineering can also be seen as an instance of a more general class of problems called "grouping" problems, also called partitioning or bin packing problems in the literature [64, 65]. These problems consist in grouping the items (the $x_i$ of our problem) of a set $U$ into mutually disjoint subsets $U_i$ in such a manner that $\cup U_i = U$ and $U_i \cap U_j = \emptyset, i \neq j$. In addition, not all groupings are allowed, but hard constraints must also be met, like the sizes of the links $U_i \leq C_i$. A hard constraint is one that cannot be violated unless the solution is unfeasible, in opposition to a soft constraint that is not mandatory. A cost function defined over the groupings must also be optimized. In our case, we want to minimize the total cost of the grouping which correspond to the traffic sent over all links. These grouping problems are known to be NP-hard in the strong sense [81]. Henceforth, there is little hope to find an exact algorithm that will find an answer in polynomial time. This is why we rely in this chapter on an evolutionary algorithm [90, 83, 120] to find an approximation to the optimal solution.

In this chapter, we shall often refer to the problem of interdomain traffic engineering as a grouping problem. Viewing it as an assignment problem is as appropriate. It is only the way we developed our algorithm that explains that we see the problem in that way.

## 4.4 Multi-objective optimization

The assignment's view of interdomain traffic engineering does not consider the number of destinations for which tweaking their BGP routes is necessary. In practice, one is not willing at obtaining a minimal cost of its traffic if it requires changing the best route found by BGP for almost all destinations. What one really wants is to get the best improvement in the cost function for a given value of the changes made in BGP. In addition, traffic demands change in time so that the optimal assignment will change each day. If the objective of the algorithm is to find the set of BGP changes that provide the best improvement within a given number of BGP changes, then the strategy of re-optimizing each day might not be the optimal one. If one wishes to reduce as much as possible to number of BGP changes on the long run, perhaps that limiting the daily improvement in the cost function each day will allow to spare many changes to be performed in BGP for consecutive days on the long-run.

So the new problem problem is now a multi-objective one, in that we try to find the minimum of the cost as well as the minimum of the BGP configurations necessary for this cost. These two objectives are conflicting since improving the cost function can only be done by applying changes in BGP. In this chapter, we assimilate the term "BGP configuration change" to mean that a given interdomain destination (be it a prefix or an AS) has to use another link than the one it had to use before the change.

There is no trivial way known in the literature to solve the conflicting objectives of multi-objective optimization problems [75]. There are however two main approaches. The first is to explicitly combine the two objectives into a single one, by a linear combination of the two cost functions. This solution consists in forming a new objective function $F$ that is a weighted sum of the individual objectives functions $f_i, i = 1, ..., N$

$$F = \sum_{i=1}^{N} \alpha_i f_i, \alpha_i \in \mathbb{R}^+. \tag{4.1}$$

This approach unfortunately forces one to choose the relative importance of each criterion. If there exists a solution that succeeds to optimize each $f_i$ then this approach can be satisfactory. In general however, some of the $f_i$ compete with one another, requiring a degradation in terms of one objective in order to improve another. In some cases, trade-offs between the different objectives are meaningless, then the $f_i$ cannot scalarized and the objectives are said to be non-commensurate.

The other approach relies on the concept of Pareto-optimality [126]: one now searches for solutions with respect to one objective, under the constraint that the solution cannot be worse in terms of the other objectives. This approach allows to find the region in the multi-objective plane that corresponds to values of the other objective that are less or equal (in the case of minimization) than a given value.

**Pareto-optimality** The concept of Pareto-optimality is related to the set of solutions whose components cannot be improved in terms of one objective without getting worse in at least one of the other components. More formally, a multiobjective search space is partially ordered in the sense that two solutions are related to each other in two possible ways : either one dominates or neither dominates. Consider the multiobjective minimization problem:

$$\begin{array}{ll} \textit{Minimize} & y = f(x) = (f_1(x), ..., f_n(x)) \\ \textit{where} & x = (x_1, ..., x_m) \in X \\ & y = (y_1, ..., y_n) \in Y \end{array}$$

and where $x$ is called the decision vector, $y$ the objective vector, $X$ the parameter space and $Y$ the objective space.

In the context of this chapter, the decision vector $x = (x_1, ..., x_m)$ represents the allocation of each destination to a particular link where $m$ is the number of destinations and each $x_i$ is an integer representing the link over which the traffic of destination $i$ is forwarded. The objective vector $y = (y_1, y_2)$ represents the pair of objectives to be minimized: the cost of the traffic and the number of "BGP configuration changes".

A decision vector $x_1 \in X$ is said to dominate another decision vector $x_2 \in X$ $(x_1 \prec x_2)$, iff

$$\begin{array}{l} \forall i \in 1, .., n : f_i(x_1) \leq f_i(x_2) \quad \wedge \\ \exists j \in 1, .., n : f_j(x_1) < f_j(x_2). \end{array}$$

Domination is an important notion because it determines the result of the comparison of two decision vectors. A decision vector $x$ is said Pareto-optimal *iff* $x$ is non-dominated regarding $X$, i.e. $\nexists x' \in X : x' \prec x$

A Pareto-optimal decision vector cannot be improved in any objective without degrading at least one of the other objectives. These are global optimal points. In our context however, we are not interested in global optima but optimal points in some neighborhood of some of the objectives. More precisely, we aim at finding the Pareto-optimal points with respect to the cost of sending traffic over the available links in a neighborhood of the default allocation of the destination among the links found by the BGP routing protocol and having a distance of at most $\epsilon$ BGP configuration changes compared to this default BGP routing solution. Hence we do not search for globally Pareto-optimal decision vectors but locally Pareto-optimal decision vectors [55]:

*Consider a set of decision vectors* $X' \subseteq X$.
1. The set X' is denoted as a local Pareto-optimal set *iff*
$\forall x' \in X' : \nexists x \in X : x \prec x' \wedge ||x - x'|| < \epsilon \wedge ||f(x) - f(x')|| < \delta$
where $||.||$ denotes a distance metric, $\epsilon > 0$ and $\delta > 0$.
2. The set $X'$ is called a global Pareto-optimal set *iff*
$\forall x' \in X' : \nexists x \in X : x \prec x'$.

Note that a global Pareto-optimal set does not necessarily contain all Pareto-optimal decision vectors. In the remainder of this thesis, we sometimes call non-dominated points Pareto-optimal while we have no explicit proof of their optimality due to the too large search spaces. To be rigorous, such points should be said non-dominated.

## 4.5 Evolutionary algorithms

In this section, we describe our evolutionary algorithm that searches for the Pareto-optimal solutions for the multi-objective minimization problem presented in section 4.4. One of the main reasons for relying on evolutionary algorithms when dealing with multiobjective optimization problems lies in that the search for solutions in terms of conflicting objectives makes life difficult for non-population based methods that search one solution after another in order to find an optimum [49]. In order to improve one objective, it is often required to degrade the performance in terms of another objective. It is therefore necessary to explore worse regions of the search space in terms of one or several objectives to find a subset of the Pareto-optimal front. This is why population-based methods are more likely to "better" sample the Pareto-optimal front by exploring several regions of the search space in parallel. In addition, evolutionary algorithms are due to be less sensitive to the shape of the Pareto-optimal front than mathematical programming techniques. Furthermore, population-based search techniques can search the Pareto-optimal front in a single run.

The code developed in this chapter is based on the GENESIS (version 5.0) system developed by J. Grefenstette [84] available at `http://www.aic.nrl.navy.mil/galist/src/#C` which we modified for our particular needs.

Genetic algorithms [90, 83, 120] are stochastic search algorithms based on the mechanism of natural selection. They rely on a population of individuals who evolve with the successive iterations of the algorithm, and stronger individuals of the population are allowed to survive through selection mechanisms in a competing environment. The basic features of a typical genetic algorithm is as follows :

- <u>population :</u> a population of individuals stochastically searches the search space.
- <u>generations :</u> from one generation to the next, only the fittest individuals survive to ensure improvement with respect to the objective function.
- <u>fitness :</u> improvement in the search is measured by the "fitness" of any individual.
- <u>selection :</u> individuals of the current population are selected to survive to the next generation based on their fitness, inducing competition among individuals.
- <u>crossover :</u> pairs of individuals generate offspring that inherit from the "genetic" material of their parents.
- <u>mutation :</u> random changes in the individuals are performed to ensure a stochastic search.

The previous elements constitute the "core" of genetic algorithms. GAs are weak search methods in that the assumptions they make about the problem at hand are limited. Genetic algorithms are a paradigm, not instances of solutions. The issue is that certain optimization problems work in search spaces that can be quite large. In practice, each problem requires that the genetic algorithm be transformed into an evolutionary algorithm [120, 158] which uses problem specific information allowing the algorithm to perform better than random search. Specific representation for the individuals together with genetic operators leverage the problem-specific knowledge in order to improve the search.

Figure 4.2 illustrates the basic working of a typical evolutionary algorithm. First, an initial population is generated. This step is often carried through a random sampling of the individuals. Depending on the purpose of the algorithm as well as the feasibility of the sampling, random sampling is carried either in the objective space or the decision space. All individuals of the initial population might also be the same if the algorithm has to start with some default solution. Second, the genetic operators are applied on the individuals in search for improvement in the objective space. Third, all individuals of the population are evaluated and their fitness is computed. Fourth, the individuals are selected according to their fitness to constitute the population for the next iteration. Finally, optional mutations are performed on the individuals to provide an additional randomness into the search.

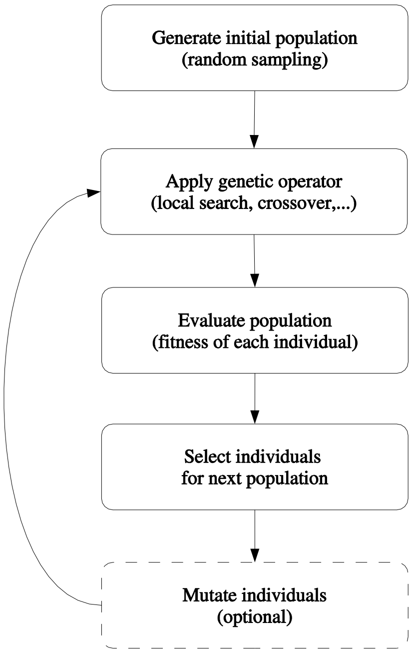



Figure 4.2: General working of an evolutionary algorithm

## 4.6 Algorithm for single-objective problem

Evolutionary algorithms are made of two important parts: encoding of the problem (data structures for individuals) and specialized operators (crossover, mutation, local search,...).

We encode individuals as integer vectors: each value $v_i$ of an element $i$ of the vector $v$ represents the link over which traffic for this destination is sent, $1 \leq v_i \leq l$ where $l$ is the number of available links. This allows an explicit representation of the assignment of each destination to a particular upstream link.

The operators we use are particular versions of the classical crossover and mutation operators. Indeed, we combine these two operators into a single one that tries to provide the intuitive advantage of both operators in the context of our problem. Each crossover phase works in the following way (illustrated in Figure 4.3):

1. select two parents (called mom and dad) to be used to generate two new offsprings;
2. copy the values of the elements of the parents into the offspring (offspring1←mom, offspring2←dad);
3. for each item i and both offspring:
   a) compare each offspring with itself where item i has been replaced by the one of the other offspring, and take the individual among the original offspring and the modified one that has the smallest cost;

b) if the cost of the minimum cost individual found in $a$ makes the total of some link larger than its capacity, then generate a random integer value between 1 and l for the value of item i, i.e. `offspring[i] = randint(1,l)`;

c) else put the value that minimizes the cost of the offspring found in a) in the element offspring[i].

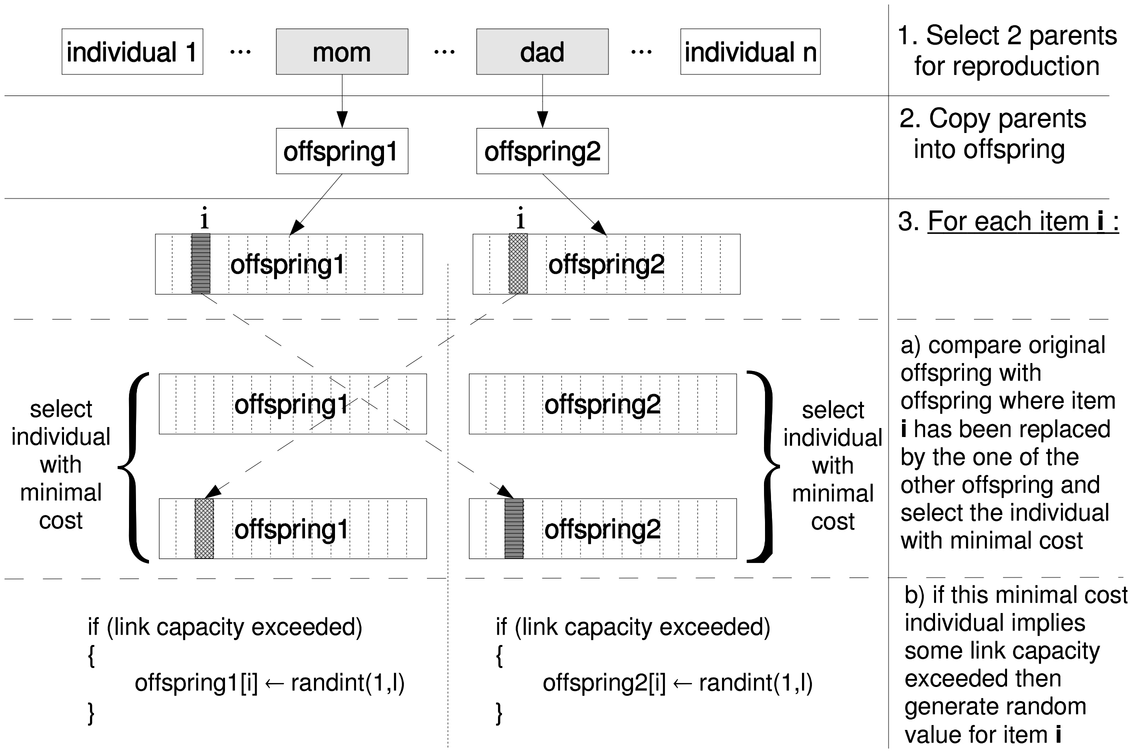



Figure 4.3: Description of single objective EA.

This procedure ensures that individuals improve by benefiting from the good groupings of the parents, and whenever the parents contain bad groupings (the capacity of some link is exceeded) a new random choice is made for the grouping of this item. The functions of both classical crossover (inheritance from the well performing groupings of the parents) and mutation (providing of new genetic material from random changes) are fulfilled. Whenever the parents allow to improve the offspring, then these changes are selected. When these changes make the solution unfeasible, then the item is mutated and its value is made random. The key of the performance of the algorithm lies in the search for improvement when evaluating each single element of the offspring.

Classical genetic crossover works on bitstrings, by injecting contiguous sub-parts of the parents in the offspring, on the grounds of the "building block hypothesis" [90, 83]. This is certainly a good approach when optimizing numerical functions, but not in the case of grouping or assignment problems [65]. The reason is that the semantics of grouping problems is to assign destinations into groups, hence the object of interest is the group, not each destination. When performing classical crossover over our individual's representation, we would inject into the offspring a group of items that belong to different groups. The reason why an individual is better than another lies in the way items are attributed to groups, not because a group of items is attributed to some set of groups. To be more concrete, improving an individual can be performed by choosing a different group (link or provider) for some item. If one changes the grouping for many items that belong to many groups, one does not know the global effect on the resulting fitness of the new individual, because changing many groupings at the same time can have as well a positive as a negative impact on the global fitness of the individual. This is why we chose our crossover to work on individual items, so that we can be sure what the effect on the fitness of the individual of changing the group of this particular item. This ensures that offspring are better than parents on every run, when paying no respect to the hard constraints.

The crossover we perform is indeed quite "greedy", in that we do it on each item of the vector. Changing potentially each item means that we perform a local search but whose depth is relatively important. This strategy increases the probability that the offspring have a better fitness than the parents by trying to improve each offspring as much as possible. Our procedure does not preclude that the algorithm be stuck in a local optimum, but it tries to find the best grouping given the initial population. The larger the population, the more chances we have to find a better minimum (or the global optimum) when applying our crossover on the individuals. When the population is large compared to the number of groupings, it is likely that our crossover operator allows all groupings to be tested for each item. Note that we talked about "groupings" mostly because we allow several changes in the assignment of the destinations to some upstream provider. If one feels more comfortable with thinking about assignments, then this view is equivalent.

## 4.7 Numerical results

Having described the working of the evolutionary algorithm for a single traffic demand, let us provide some numerical simulations to show how well it performs. Assume a network has four upstream providers that bill the traffic differently :

- provider 1 uses flat rate billing: sending any quantity of traffic on that link costs fifty monetary units (whatever the quantity) and the maximal link capacity is fifty traffic units,
- provider 2 uses volume-based billing: sending traffic on that link costs 1.4 monetary units per traffic unit and the maximal link capacity is of twenty traffic units,
- provider 3 uses volume-based billing: sending traffic on this link costs two monetary units per traffic unit and the maximal link capacity is of twenty traffic units,

- provider 4 uses "burstable" billing: sending any quantity of traffic on that link below the commitment of twenty traffic units costs twenty monetary units, and any additional traffic exceeding the commitment of twenty traffic units costs 1.5 monetary units per traffic unit.

For one hundred traffic units, the optimal global cost should be around 113 with the following traffic allocation over the links: fifty traffic units for link 1, twenty traffic units for link 2, ten traffic units for link 3 and twenty traffic units for link 4. The optimal solution that was the largest among all runs of the algorithm (thirty different traffic demands were evaluated) found after twenty evaluations (19 generations) was 113.21. Note that we used a population of fifty individuals.

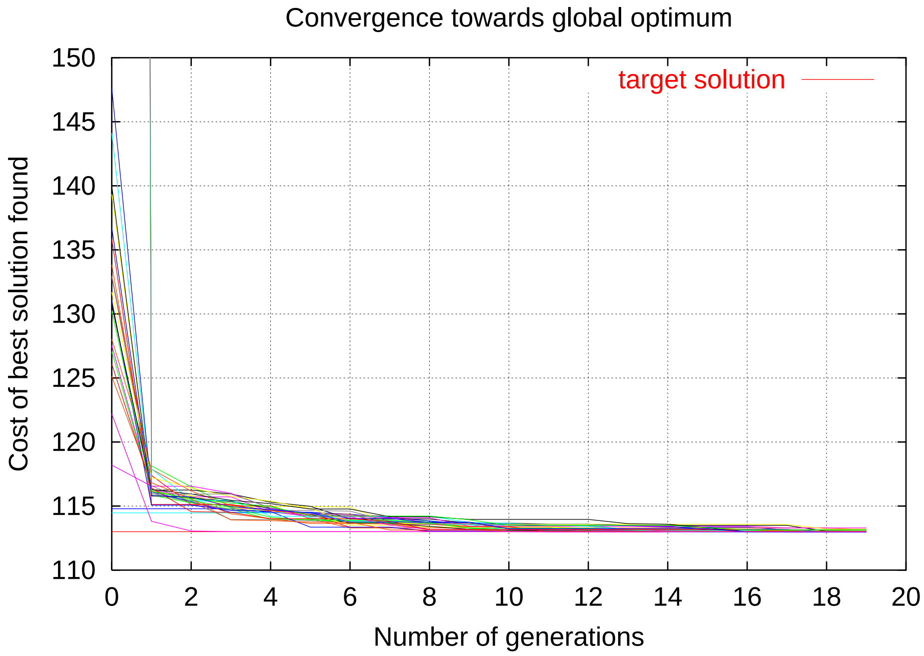



Figure 4.4: Convergence of the single-objective EA for 30 days of traffic demands.

Figure 4.4 presents the convergence speed of the evolutionary algorithm for one month worth of traffic demands. These traffic demands correspond to the outgoing traffic (ASes) from Abilene for each day of the month of September 2002 [1]. In this simulation run, we simulated a provider that had the Abilene traffic demands each day and wanted to minimize its total cost of sending traffic each day. Except for the first day where the choice of the upstream link for each destination was random (uniformly among the four links), the global optimum of each day was used to start the algorithm for the next day. The cost of the solution after the first evaluation can be above 1000 because we used a penalty function to prevent solutions to exceed the maximal capacity on any link. In case of any solution having more traffic than the maximal capacity for some link, the evaluation function returned a cost of 1000 for this link. Note that the part of the "crossover" that attributes a random value for the value of an item has been seen to occur only during the first generation of the algorithm when having to adapt from one traffic demand to another. Once the largest destinations have been re-attributed onto least-loaded links, the algorithm spends time exclusively on minimizing the cost, not on trying to find feasible solutions. This shows that not caring about the traffic demands from one day to another while relying on an optimal solution could seriously impact on the next day's solution. The results of Figure 4.4 for generation zero show that yesterday's best solution can be far from optimal on the next day if no re-optimization is performed. Hence, using an optimal solution for the next might very well be a bad idea. In particular, we used a random attribution of the destinations for the initialization of the algorithm, which is optimistic compared to the way the "tie-breaking" rules of the BGP decision process will choose the default allocation of the traffic among the upstream providers in practice. The next chapter will discuss the inoptimality of BGP in that respect.

As illustrated on Figure 4.4, the convergence towards the optimal solution is fast at the beginning. Further approaching the global optimum on the other hand appears more difficult, and more and more comparisons between different groupings are required to get closer to the optimal value. This convergence illustrates the nature of this combinatorial optimization problem, where it is not very difficult to get close from the optimum, but harder and harder to get closer to it. The running time of each day's optimization (twenty evaluations) requires in the order of one minute on a PIII 1 GHz running Linux.

## 4.8 Abilene Netflow statistics

To evaluate our algorithm, we use in this chapter the data available on the web from Abilene [1]. Abilene collects Netflow [45] statistics every day from its POPs and allow these statistics to be queried from the web at `http://www.itec.oar.net/abilene-netflow/`. For this chapter, we used four month of data, starting from June 1$^{st}$ 2002 to September 30$^{th}$ 2002. To evaluate our algorithm, we relied on the "destination AS" reports (percentage without names) from September 2002. For the algorithm with time-varying demands, we use the whole "source-destination AS" traffic matrix because it provides with total byte counts rather than percentages for each AS. Relying on byte counts instead of percentages is necessary since each day's traffic level matters for the computation of the expected long-term gain of the optimization.

To give an idea of the variability of the traffic demands from day to day, Figure 4.5 presents the amount of traffic in bytes for the largest destination ASes of Abilene. We have ordered the destination ASes seen during the four months of the trace (x-axis) and we plotted the amount of bytes (in logarithmic scale) sent to the one thousand ASes with the largest amount of traffic for each day of the trace. Most ASes received daily an amount of bytes that varied by a factor of ten or more. This shows that although ASes that receive a lot of traffic often do so over large time periods, the absolute amount of traffic can vary a lot between successive days.

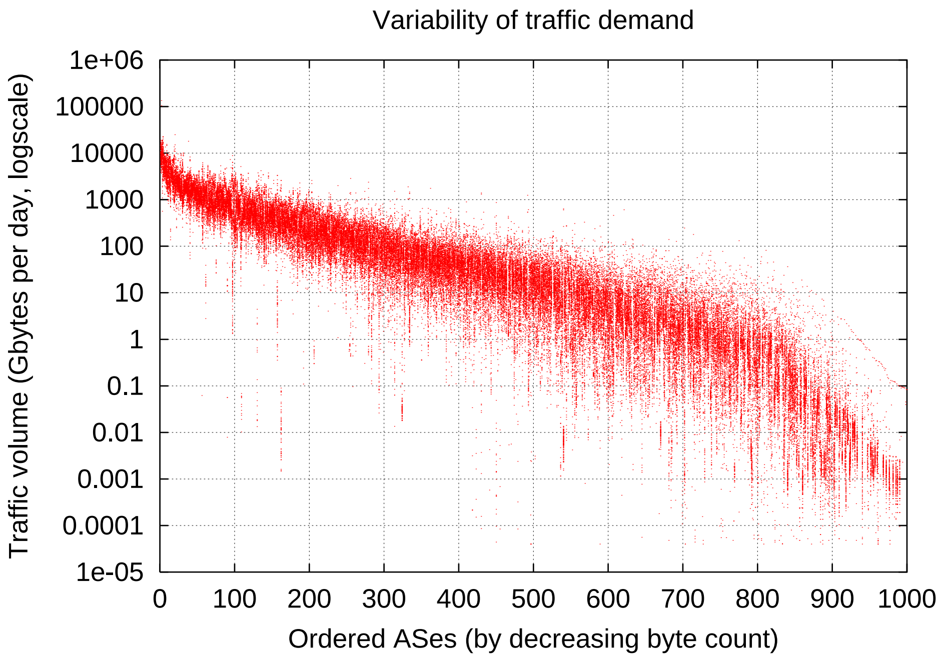



Figure 4.5: Traffic demands of Abilene destination ASes.

## 4.9 Minimizing BGP configuration changes

The previous section has provided a sketch of our solution, we need tough to explain how to modify the evolutionary algorithm to adapt to changing traffic demands and minimizing the number of BGP configuration changes when re-optimizing the attribution of the destinations from one period to another. Because we have no way to explicitly provide a trade-off between the cost of sending traffic over links and the non-peculiar cost of changing the BGP configuration in order to force the traffic of one interdomain destination to be forwarded over some upstream provider, we must rely on Pareto-optimality introduced in section 4.4. Henceforth, we modify the evolutionary algorithm described in section 4.6 by adding the constraint of trying to improve the new objective while minimizing BGP configuration changes. Recall that we use the term “BGP configuration change” to mean changing the allocation of one interdomain destination to an access link other than the default one chosen by BGP, although we know that in practice the two things are not forcibly equivalent.

Due to the large size of the search space, we cannot realistically search for the Pareto-optimal front blindly. The knowledge of the number of BGP changes required for the single-objective problem should guide us for deciding a strategy to find a good trade-off between minimizing both the BGP changes and the cost of sending traffic. Our experience with the single-objective optimization showed us that many BGP changes are required for getting close to the global optimum (a few hundreds), while the pay-off for a large fraction of these BGP changes is tiny. This means that we cannot rely on the cost of sending traffic to find the Pareto-optimal front since it would over-sample the region of the front that requires many BGP changes. Nevertheless, we need to guide the evolutionary algorithm to improve in terms of the cost of sending traffic while minimizing the number of BGP changes. Our approach hence consists in starting from zero BGP changes (the solution found by BGP) and searching for the best solutions in terms of the cost of sending traffic and we allow a given number of BGP configuration changes at each generation. So what we basically do is the same as the single-objective algorithms except that at each generation, we only allow a given number of randomly chosen items to be changed.

The pseudo-code of the new evolutionary algorithm for a particular traffic demand is the following:

1. Begin with the solution given by last iteration (previous BGP configuration).
2. For each generation and each individual of the population: select randomly $n$ destinations among the destinations active during the forthcoming period and for each destination find the assignment that minimizes the cost of this solution.
3. If the number of generations is larger than the number of changes allowed by the provider: stop.

Point 2 requires some further description. We cannot for some given value of $n$ compare all feasible solutions implying at most $n$ BGP configuration changes. The reason is once more the size of the search space that would represent about $Popsize \times l^n$ points at each generation: given that we use $l = 4$, $Popsize = 100$ and $n = 10$, this would require $100 \times 4^{10} \sim 10^8$ evaluations per generation. Each evaluation requires as many additions as there are items in the vector (about 800) so even if the computer executes one addition per $10^{-9}$ second, one generation would require eighty seconds which is still feasible. In practice, the time needed to process the population during one generation is more than that because we need to access to the individuals in memory, write to memory the best element of each generation to record the individuals of the Pareto-optimal front, without mentioning other overheads due to the evolutionary algorithm.

Figure 4.6 presents the pseudo-code for point 2 above. For each individual and for each generation, we generate $n$ times one item among the destinations that have traffic for the considered period. For this item, we determine the link that implies the smallest cost for this individual. This means that we perform $n$ evaluations for each individual and each generation. The total number of evaluations for each generation is thus $Popsize \times l \times n$ instead of $Popsize \times l^n$, this reduces the number of evaluations a lot. One could ask why we do not you evaluate all the $l^n$ solutions to be certain to find the best solution requiring at most $n$ BGP configuration changes. The answer is quite simple. Assume you have the best solution for the first $n$ changes with respect to the default BGP solution. If $n$ is not too large, then we have seen in the previous paragraph that it is possible to compute all the possible solutions. This would provide all points on the Pareto-optimal front for the first $n$ values of the BGP configuration changes. The issue there, is that you know you cannot do this for large values of $n$: even for $n = 20$ (which is not that large), it would require about $4^{20} \sim 10^{12}$ evaluations in our case, about more than 9 days (with the

```
foreach individual k {
    foreach item = 1 to n {
        // selecting one item randomly among the active destinations
        selected_item = active[randint(1,active_destinations)]
        // finding the link giving minimal cost for individual k
        best_link = min_cost(k,selected_item)
        // changing the link for item iff cost has decreased
        if (cost(k,selected_item,best_link) < current_cost[k]){
            // updating the value of this item
            k[selected_item] = best_link
            // updating the cost of the individual
            current_cost[k] = cost(k,selected_item,best_link))
        }
    }
    // computing the number of BGP configuration changes for this solution
    // compared to default BGP solution
    BGP_config[k] = required_BGP_config(k,selected_item,best_link)
}
```

Figure 4.6: Pseudo-code for searching the Pareto-optimal front.

optimistic hypothesis of one evaluation per $10^{-7}$ second) to find the first 20 points of the Pareto-optimal front. So assume we have the Pareto-optimal front up to 20 BGP configuration changes, the next generation has to start with a population made of the best individuals known up to that point. Given that we have a population of limited size−increasing the population allows a better search at the cost of more memory consumption and larger overheads for the evolutionary algorithm−which strategy is better ? Starting from the best solution for 20 BGP configuration changes or some other population ? Indeed, we do not know which strategy we should use to be sure to sample correctly the Pareto-optimal front. Even if we could evaluate all the solutions starting from the best-known point at every generation, this would not provide us with the certainty that we have got the true Pareto-optimal front. For this, we have no other choice than trying all possible solutions with a given number of BGP configuration changes. Remark that in Figure 4.6 we need to recompute the number of BGP changes for each individual after having tried the $n$ items for the current generation run because we do not know in advance how many BGP configuration changes are required for this new solution. This is due to the BGP configuration changes performed during the current generation can undo changes done in a previous generation, so both the number of BGP configuration changes and the cost of a solution could decrease from one generation to the next.

While this method does not provide the whole Pareto-optimal front, it has several nice properties. Firstly, we have a certainty on the maximum number of BGP changes the solutions found require. Each generation implies at most $n$ BGP configuration changes so we can stop the algorithm when all individuals of the current population require more than the higher bound specified by the provider in terms of BGP changes. Second, because we cannot search for all changes of upstream provider for all destinations, we allow the algorithm to select the fittest solutions at each generation. This allows the search to focus on the BGP changes that decrease the most the cost of sending traffic. Hence we try at each generation to decrease by the largest amount possible the cost of the solution, under the constraint that the number of additional BGP changes is smaller than $n$. This heuristic does not give any assurance that we shall find the Pareto-optimal front but at least we try to get the best short-term improvement for a few number of BGP configuration changes. There is no other choice anyway to find good solutions when limiting the number of evaluations we allow ourselves.

Our approach is not the best one for searching the Pareto-optimal front in terms of both objectives. However, using a method that does not rely on the number of BGP changes to search for the multi-objective solutions has a high risk of not being able to find good trade-offs between the two conflicting objectives. We actually oversample the regions with the least number of BGP configuration changes because we know that we are not able to search the whole search space. The advantage of our solution is that it has a built-in upper bound on the number of BGP changes that some solution will require. In addition, we do not have to find a strategy for searching the "cost-BGP changes" plane while spending most of our time trying to decrease the cost while minimizing the number of BGP changes. To make a link between this section and section 4.4, $X$ representing the set of feasible solutions $x$, what we are after here is the global Pareto-optimal set $X'$ of non-dominated points $x'$ that have the smallest cost of their groupings within a distance of $\epsilon$ ($\in \mathbb{N}$) BGP configuration changes with respect to the solution $x^*$ found during the last period. If the function $cost(y)$ represents the cost of grouping $y$ and $bgp(y)$ represents the smallest number of BGP configuration changes required to get from grouping $x^*$ to grouping $y$, then
$\forall x' \in X' : \nexists x \in X : cost(x) < cost(x') \wedge bgp(x) < bgp(x')$.

The reader could have noticed that this new evolutionary algorithm does not contain any crossover. We do not exchange information between parents and offspring, but each individual evolves (or dies) depending on its relative cost. There is a simple reason for that. The single-objective algorithm has shown that the effective part of crossover was the possibility of trying to improve each solution by changing as many items (destinations) as possible. The effective part of the algorithm is not the crossover. The choice among the two different groupings of the parents is not critical in the search. Allowing many items to be changed at each generation on the other hand ensures the largest improvements in the search. So for searching for the Pareto-optimal front of a grouping problem, we have found that the most effective algorithm is the one that allows as many changes of grouping for each individual. Restricting the number of BGP changes thus automatically provides a trade-off between the effectiveness of the search and the constraint on the number of BGP configuration changes.

## 4.10 Predicting Future Demands

Up to now, we have made the assumption that the traffic demands were known beforehand. This is not true in practice. The scheme we have presented in section 4.6 works on the current traffic demand, hence assuming we already know it. There is a big issue here: how to predict the forthcoming traffic demand?

The algorithm does not know two things: the future traffic demand and the BGP routes. We do not know for which destinations we shall see traffic. Chapter 3 has shown that a large fraction of the AS paths were stable but a significant fraction was relatively short-lived. Chapter 3 has also confirmed [139] concerning the stability of the BGP routes for the largest fraction of the traffic. This stability unfortunately does not help us to predict over which upstream link we should send the traffic for a particular destination. It only tells us that relying on enough measurements should provide good results for the future, but we have to find out how much data we need to take into account to obtain the best results from our knowledge of the traffic demand and BGP routing information. Furthermore, in this chapter we consider that it is better for a provider to know the total number of BGP configuration changes compared to the default BGP routing than to rely on the previous day's solution. Another technical problem of relying on the last day's solution to find the current Pareto-optimal front is that we still do not know which solution of the front the provider wants to choose, how many BGP changes are allowed. So we should have to start from yesterday's Pareto-optimal front to find today's. This is not possible because today's front could be more costly, hence requiring the search to progress in many directions towards the new front (which we do not know). The last issue of this approach would be that we start from a potentially unfeasible region, we would then spend our time trying to find feasible solutions, instead of exploring the Pareto-optimal front.

So the main question we try to answer in this section is the one of how long in the past should we go in order to be able to find the best prediction in terms of the upstream link we should use to send traffic towards a destination ? To answer this question, we do not try to build models for the various unknowns of our problem. The reason is that various providers can have different traffic types according to their users' profile as well as different characteristics of their BGP peerings. So results obtained in one particular context will differ in another. Not knowing what are the important properties of the traffic for our particular problem, modeling it is unsound. Modeling requires assumptions about the features of the modeled system, which we do not have.

Left without a model, we rely on three months of Abilene data. For each day that has to be traffic engineered, we study the Pareto-optimal front found by our algorithm given that we rely on the Pareto-optimal front for the last $t$ days to predict the future traffic demand. What we do for each day is the following:

1. based on the BGP routing tables of the provider, determine through which upstream each destination should be sent if BGP was used to forward the traffic;

2. for each value of the time window $t$, compute the Pareto-optimal front for the traffic demand of the last $t$ days;

3. compute the cost of the Pareto-optimal front found in point 2 for the forthcoming traffic demand.

We can now compute the Pareto-optimal fronts for the three months of the Abilene traffic data, one for each value of the time window. In addition to generating many figures, in practice we do not have the opportunity to choose the right time window each day so that the "best" Pareto-optimal front will be found for the next day. Instead, a provider will have to choose a value of the time window and a maximum number of BGP changes to be made every day to get one point from the Pareto-optimal front. A provider will probably prefer to have an algorithm whose parameters do not have to change every day and be provided with an uncertain long-term planned gain of sending its traffic for a choice of the maximal number of daily BGP changes than having to choose a point on the Pareto-optimal front each day.

Figure 4.7 presents the global Pareto-optimal fronts for three months of the Abilene data [1]. We used the months of June, July, August and September 2002 to predict each day of July, August and September. These traffic statistics were used together with four different BGP routing tables from four major tier-1 providers that were gathered on Oregon route-views [119] to simulate the upstream providers. We chose BGP tables from four major providers from October $3^{rd}$ 2002: AT&T, Sprint, Level3 and Verio. Using a more recent BGP routing table than the traces was to ensure that all AS numbers of the traffic traces would be known. The purpose of choosing these large tier-1 providers' BGP tables was to provide enough redundancy for the reachability of the ASes over the four links. Each figure shows the Pareto-optimal fronts for each value of the time window. We show only values of 1, 5, 10, 15, 20, 25 and 30 days on these figures. Each point represents the total cost of sending traffic using this allocation of the destinations ASes over the upstream links (compared to the standard BGP routing) and its associated maximal number of BGP configuration changes required each day. A point (x,y) gives the expected gain $(1-x)$ in terms of the cost of sending traffic compared to the classical BGP solution when at most $y$ BGP configuration changes are applied each day.

Several points concerning the way we compute the global Pareto-optimal (over several days) front need to be explained. First, for deciding which upstream to use for a given AS, we chose the upstream having the shortest AS path and a random choice of the upstream when several same length AS paths were found. This implies that there is not one initial cost for zero BGP configuration changes but many, the differences are however small compared to the difference between solutions on the Pareto-optimal front. Second, solutions on the Pareto-optimal front are monotonous in the case of one day, but not in the case of one month. This is because the sum of several monotonous functions is not monotonous anymore, generally speaking. Allowing several BGP configuration changes at each generation implies sampling the complete Pareto-optimal front for each day. One must be aware that there is a trade-off between the efficiency of the search and the sampling of the Pareto-optimal

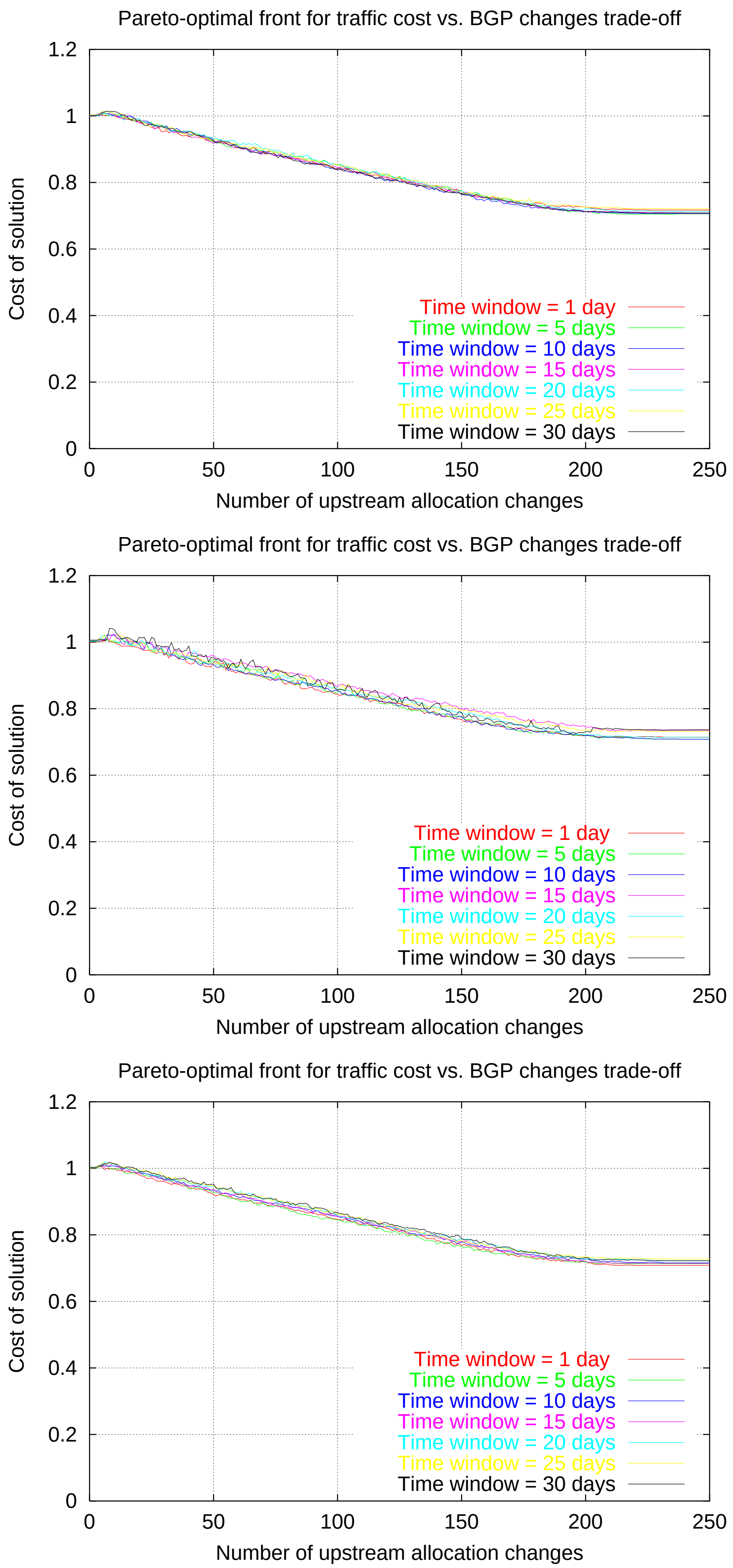



Figure 4.7: Long-term Pareto-optimal fronts for July (top), August (middle) and September (bottom) 2002.

front. Constraining the search to improve of one BGP configuration change at each generation dramatically slows down the improvement in terms of the cost.

Getting back to figure 4.7, the first thing one can notice is the absence of remarkable difference between the various Pareto-optimal fronts found by the values of the time window. This indicates that for this traffic demand and BGP tables, relying on more days in the past does not improve much the solution found on the long-term. This result is extremely valuable because it allows the algorithm not to take into account a lot of data to predict the future. Although our results could be due to particular context, it does not appear evident that using a large amount of data helps finding less costly solutions on average. The second thing one can observe is that the best BGP solutions (0 upstream allocation change) can be better than relying on a few BGP configuration changes (up to twenty). This means that if a provider does not want to rely on more than twenty BGP configuration changes, it is not worth trying to optimize. On the other hand, if a provider is ready to rely on many BGP configuration changes, then the expected gain can be significant, depending on the relative cost of its upstream links. The gain shown on figure 4.7 only reflects our choice of the relative cost of the four upstream providers we simulated for the Abilene traffic demand. Indeed this gain cannot be generalized to other situations. In our case, the cheapest upstream provider had a cost 4.5 times smaller than the most expensive one. Differences of interdomain bandwidth prices of a factor two or three are not uncommon in the current Internet (see `http://www.telegeography.com` for instance).

The optimization of each day's traffic demand (more than 850 destination ASes per day) took about 20 minutes on a PIII 1 GHz running Linux.

## 4.11 Evaluation

In this section we evaluate the advantages and drawbacks of the approach we used in this chapter to tackle the interdomain traffic engineering problem.

First, the cost objective can be replaced by any other objective that can be evaluated on each interdomain destination: quality of the routes (delay, bandwidth, hop count distance,...), load balancing, optimizing simultaneously the requirements of different traffic types,... There are many ways to easily adapt our scheme for different particular needs. For optimal load balancing for instance, the objective would be to minimize the maximum of the traffic sent on any link (or the load of any link). For that, instead of minimizing the cost of one solution, one would try to minimize the maximum traffic sent over any link (or the load). This would automatically distribute the traffic (or load) on all the available links. The main advantage of our technique is its generality with respect to the main objective. We do not claim in this chapter that the gain is worth the effort but in some cases where the relative cost among the upstream providers can be large we are confident that the long-term gain can be significant without implying a high burden on BGP. The main advantage of our solution is that it can be very easily adapted to other traffic objectives. In addition, because we change the `local-pref` attribute of the BGP routes, our solution can work even if BGP tweaking is performed on the routes before we start the algorithm. We do not require that the solution we start with be the default traffic distribution of BGP. Any default solution will do. In practice, one could imagine running this algorithm on a route reflector which whould insert the right local-pref values into the BGP routes and redistribute them to all BGP edge routers dynamically.

Second, it must be noted that the original target of this chapter are multi-homed non-transit providers. These providers could benefit from such a scheme without too much problems regarding the behavior of their BGP routing. For what concerns transit providers on the other hand, the benefits of our approach are less clear. Large transit providers do not necessarily have such large cost constraints on their interdomain links, so minimizing the cost of the traffic may not apply or be less important than intradomain issues [76]. Some benefits could be achieved with transit providers if some cost function represents some real interest, for instance if a large transit provider wants to find a better selection among its peers for each source or destination AS, evolutionary algorithms could find some better answer than the one classically found by IP protocols in a robust way.

An application of our algorithms would be to find the best peer AS (not the physical links) to send or receive packets to/from each prefix or AS. If the transit provider does peering with a few (let us say up to a few tens) large transit providers, then based on long-term traffic matrices measurements it should be possible to find the best way to exchange traffic with these transit peers for all source and destinations globally. For this, an MPLS-engineered network would constitute the best target since MPLS allows for the required traffic demands to be available and the control of the MPLS LSPs also allows the provider to decide how its traffic demand should be mapped to its physical topology. The drawback of evolutionary techniques to solve such problems in the lack of any guarantee of the optimality of the solution found. However, the main advantage of the technique is that it can comprise several aspects to be taken into account during the search for a better solution. Optimality-proof techniques on the other hand tend to be relatively difficult to handle in situations where the solution to be found is difficult to model explicitly.

Finally, on the combinatorial optimization side, we presented in this chapter an evolutionary algorithm capable of solving grouping problems efficiently. Our single-objective optimization algorithm described in section 4.6 demonstrated that this class of NP-hard problems could be solved in a greedy way, by combining crossover and mutations into a single "genetic operator". Although this operator is not suited to perform interdomain traffic engineering with BGP if the number of BGP filters is to be minimized, it exhibited interesting convergence properties towards the global optimum that make its efficiency likely on other grouping problems.

## 4.12 Conclusion

In this chapter, we provided a first evaluation of the feasibility of tweaking BGP for interdomain traffic engineering. We explained in section 4.3 the combinatorial nature of the interdomain traffic engineering problem. Then, we described our first evolutionary algorithm in section 4.6. We showed the main elements for the success of solving the single-objective problem (minimizing cost only). We then introduced in section 4.9 another evolutionary algorithm that tried to rely on as few BGP changes as possible when re-optimizing the traffic objective for the successive days. We carried out simulations on real traffic demands for both algorithms and evaluated their effectiveness. Finally, our study of the long-term gain on three months of traffic demands from Abilene under changing traffic demands showed that there was no gain at using more than the previous day to optimize the next day's traffic demand. This long-term study also showed that on the traffic demands used, a large number of interdomain sources or destinations could have to be influenced in order to get close to the global optimal value of the cost objective.

# Chapter 5

# Interdomain traffic engineering with minimal BGP configurations

## 5.1 Introduction

In this chapter, we first explain in section 5.2 the limitations of the previous chapter. Section 5.3 then presents the evolutionary algorithm aimed at solving the new interdomain traffic engineering problem, when trying to minimize the number of BGP filters to be applied on the BGP routes. Section 5.4 provides simulations that study the working of our algorithm on the problem of interdomain traffic engineering with minimal BGP configurations.

## 5.2 The multi-objective combinatorial optimization problem

In the previous chapter, we have dealt with what we called the "simple" problem. The reason why we called this problem "simple" is that it was purely combinatorial. It does not make any explicit mention to the number of `local-pref` changes we need to configure on the BGP routers is potentially as large as the number of destinations for which we need to override the choice made by BGP in order to distribute the traffic as we would like it among the available providers. Our aim however would be to utilize as few BGP filters as possible in order to achieve the "best" (according to the objective function) distribution of the traffic. This problem is thus a multi-objective combinatorial optimization problem, since we want to optimize the objective function while minimizing the number of BGP filters at the same time. So in fact we are not interested in the pure combinatorial problem since the optimal solution to the assignment problem could involve too many BGP filters. On the other hand, we do not know the cost of a BGP filter, this objective is not measurable. Hence our two objectives, cost of traffic and number of BGP filters, are *non-commensurate*.

Given that we want to find the optimal distribution of the interdomain traffic for any number $n$ of BGP filters to be configured, our objective is to find those among the best $n$ filters for the $m$ possible destinations and the $p$ available providers, but starting from the default BGP configuration. So the actual problem we have to solve is not an unrestricted search by trying to swap the destinations among the providers but we need to know how to tweak BGP and find the successive filters that allow to reduce the objective function starting from the default solution distribution found by BGP.

Now a crucial step is to get a rough idea of the size of the search space that we are going to work in. Assuming there are about one thousand ASes with which the local AS exchanges a significant amount of traffic each day among the fifteen thousand ASes that composed the Internet in 2003, and given that we have a worst case of the $p$ providers having each one route towards each destination, finding the best ten BGP filters to apply to the routes would require to search for the best ten BGP filters among one thousand times $p$ possible ones. For the case $p = 2$ and the first ten BGP filters, this amounts to $\frac{2000!}{10!1990!}$ different possibilities. Of course, the number of filters that can actually be applied is smaller since in the case of two providers there will be always one default BGP route towards any destination and only one filter can be applied to eventually change the provider used by the BGP default routing. Still, the number of possible filters is $\frac{1000!}{10!990!}$, a number with twenty-three decimal digits. Not knowing how many BGP filters will be required to find an "acceptable" solution to the multi-objective problem, we need to reduce the size of the search space in some way.

In order to reduce this number of BGP changes, we need to aggregate the interdomain destinations into coarser objects. The AS-level topology obtained from the BGP routing tables constitutes a sound basis for that, by aggregating the destinations into what we call "AS sink trees". So let us define some preliminary concepts that will be useful in the remainder of this chapter. A "terminal AS" (destination) is *an AS that appears as the last AS in the AS path of a BGP route*. A "sub-ASx path" of an AS path is *the suffix of that AS path whose first AS is the first occurrence of ASx*. Note that we cannot remove the "first occurrence" part of the previous definition because of AS path prepending [156]. Even though loops do not occur in the paths of a path vector protocol like BGP, repetitions of the same AS are allowed. The "ASx sink tree" is an instance of an "AS sink tree", defined by *the set of all ASes that appear in all sub-ASx paths in a given BGP table*. A practical, but important reason to utilize these definitions is that all existing BGP routers can be configured to attach `local-pref` values to learned routes whose AS path matches such a regular expression [87]. It must be noted that the AS-level topology as seen by a stub AS is not a tree nor a forest. So there are chances that an AS might appear in several "AS sink trees". In the case that several BGP filters defined on "AS sink trees" where some common ASes belong to them, then we do not try to solve this conflict of which BGP route will be chosen, but the BGP decision process will.

Getting back to the problem of the size of the search space, let us evaluate the gain at considering AS sink trees instead of interdomain destinations. In the simulations proposed in section 5.4 with a little less than one thousand terminal ASes, we have about more than eleven thousand AS sink trees while when taking into account only the AS sink trees for ASes located two AS hops away from the local AS (which we call "level 2" AS sink trees), this number reduces to about one hundred and fifty for two providers. This means that considering such "level 2" AS sink trees would result in a number of possible filters of $\frac{150!}{10!140!}$ points for two providers, a number with fifteen decimal digits instead of twenty-three. This is still a large number but the gain is significant. We cannot perform an exhaustive search nor a random search in such a large space. An exhaustive search would take too long while a random one would not guarantee that we find a good solution. Biasing the search by trying first the most important destinations in terms of their traffic provides good solutions for a few BGP filters only and does not provide any advantage for a few tens of filters. Using an integer program to solve this problem is quite difficult, mainly because our problem depends on the behavior of BGP when a filter is applied. An integer program would require that we be able to explicit the behavior of the BGP decision process which is by no means simple. Our algorithm simulates a subset of the real BGP decision process and takes as input real BGP routing tables, hence we do not rely on any theoretical assumption concerning the way BGP works in our algorithm, which we could hardly do with an integer program. So the "evolutionary" approach provides a simple way to search for good BGP filters in a way that is close semantically to the problem we are trying to solve. It must be noted that we cannot afford to try to find in addition to the best AS sink trees the optimal value of the `local-pref` attribute for each AS sink tree because this would make the size of the search space increase and make the search even more difficult. Although BGP encodes the `local-pref` attribute as a 32 bits integer, we only utilize two different `local-pref` values in this chapter, a default low value for all routes (100 for instance) and a single higher value (110 for instance) for the routes on which a BGP filter is applied.

It must be noted that relying on the AS path attribute to aggregate the traffic is one particular way to define BGP filters. One could think as well about working on the BGP prefixes and try to aggregate the traffic based on the IPv4 address space. The choice of the best way to define BGP filters is left as an open issue.

## 5.3 The evolutionary algorithm

Given that our problem is largely combinatorial, we rely on an evolutionary algorithm that will perform a stochastic search using a population of solutions to find among the BGP filters the most interesting ones. By most interesting, we mean those filters who improve the most the global cost of the traffic.

We encode an individual as an integer vector $v$: each value $v_i$ of the vector $v$ represents the provider to be used to attain a particular destination. An individual is hence equivalent to the output of the BGP decision process.

Our algorithm works as follows. We initialize the search by running the BGP decision process once and all individuals are then set according to the results of the BGP decision process. At each iteration and for each individual, we choose randomly (and uniformly) an AS sink tree and compute through which provider we should sent the traffic for all the destinations of the AS sink tree. For each provider, we apply a `local-pref` value higher than the default one to all the routes of this provider for these destinations and run the BGP decision process to know the new distribution of the traffic among the providers. If this change leads to improvement of the objective function we actually change the `local-pref` value of all the routes of this AS sink tree and this new individual replaces the old one. The main issue is that we cannot know in advance how the distribution of the traffic among providers will change due to both the BGP decision process and the overlap of the destinations among different AS sink trees. In theory, it should be possible to compute the actual amount of traffic that overlaps among two AS sink trees, but given the complexity of the BGP decision process it is not trivial to assess whether this overlap in the AS sink trees will help in determining the exact effect on the traffic distribution of these overlapping AS sink trees. If some destinations have the same value of `local-pref` due to different BGP filters, it will be up to the tie-breaking rules of the BGP decision process to decide about the final traffic distribution. Among these individuals not all lead to an improvement in the objective function, we do not remove the individuals which did not lead to an improvement during the current iteration because they could permit improvements during the next iterations. It is up to the selection procedure of the evolutionary algorithm to decide which individuals must survive.

Figure 5.1 provides a pseudo-code version of the search procedure used in our evolutionary algorithm. The most important parts are lines 9 to 13. In line 9, we search in the AS-level topology of provider *provider* all the destinations that are in the chosen AS sink tree. The AS-level topology from the BGP routes for interdomain traffic destinations is know to be relatively flat: most destinations are within 3 or 4 AS hops away from the local ISP [174]. So we use a recursive breadth-first search to find the destinations within a given AS sink tree. In line 11 we apply the BGP filter by putting a `local-pref` value of 110 to all routes of this AS sink tree and in line 13 we run the BGP decision process but only on the BGP routes having as last hop in their AS path an AS number with whom traffic has been exchanged, reducing the number of routes on which the BGP decision process has to be applied.

Two crucial parts of the algorithm are not shown in the pseudo-code above: the selection process and determining the population size. First, the selection process is the key to sample efficiently the search space. We use the classical stochastic universal sampling [19] to perform the selection of the individuals that survive from one generation to the next. The selection process is critical to focus the search on interesting individuals (from the viewpoint of the objective function) while still retaining a sufficiently large diversity of solutions in the population at each generation. So it must select the individuals according to their relative fitness without biasing the selection too much towards highly fit individuals.

The second important part is the size of the population. In order to ensure that all AS sink trees are tried at least once at every generation, the size of the population must be larger than the number of available AS sink trees that the current aggregation level contains. If the population size is smaller than this number of available AS sink trees, then it is possible that we do not find the best BGP filters during the

```
foreach individual k = 1 to Popsize {
  // try uniformly a random AS sink tree
  sink_tree ← randint(1,num_sink_trees)
  // try all providers for this AS sink tree
  foreach provider = 1 to p {
      // check whether this provider is connected to this AS
      if (connected(provider,sink_tree) == TRUE){
          // compute the destinations for this AS sink tree and provider
          destinations ← dest(provider,sink_tree)
          // change local-pref for the routes of the destinations
          solution ← change_local-pref(k,provider,destinations)
          // run the BGP decision process
          distribution = BGP_decision_process(solution)
          // compute objective function value
          new_cost[k] ← compute_obj_value(distribution)
          // check whether we improved in the objective function
          if (new_cost[k] < old_cost[k]){
              // update the individual k with improved solution
              k ← solution
              old_cost[k] ← new_cost[k]
          }
      }
  }
}
```

Figure 5.1: Pseudo-code for one generation of the EA.

early generations of the algorithm. So a large enough population size is important to get the best out of the early generations of the algorithm. As the number of BGP filters applied increases, the fraction of the search space sampled becomes so small. In that case, increasing the population size does not provide any interesting pay-off, hence the population size should be chosen to efficiently sample only the first few BGP filters.

## 5.4 Simulations

The purpose of this section is to study the behavior of our algorithm to get an idea of the impact of some networking related parameters. The parameters we are interested in are the number of providers, the objective function and the AS sink tree aggregation used during the search.

From the discussion of section 5.2, we have seen that increasing the number of providers dramatically increases the size of the search space. The type of objective function can also make the life harder for our algorithm, for instance if the improvements in the objective are a complex non-linear function of the amount of traffic that are switched among providers, then the algorithm might take more generations to improve the objective. Finally, we would also like to determine the "ideal" AS sink tree aggregation to perform interdomain traffic engineering. Allowing all possible AS sink trees to be tried (any AS in the AS-level topology) is due to provide the largest flexibility for the search since the algorithm could find the minimal number of BGP filters (i.e. AS sink trees) to obtain a given value of the objective function. However, allowing more AS sink trees means to increase the size of the search space, hence slowing down the search.

In our simulations we use two different objective functions. The first objective function is the maximum amount of the traffic exchanged on any of the interdomain links. This objective function tries to balance the traffic sent to the upstream providers. The second objective function is a volume-based billing cost defined on the traffic sent to the upstream providers.

The scenario of our simulations consists of a stub provider connected to two to four large tier-1 providers. The traffic demand of this would-be stub provider was gathered on the web from the Abilene Netflow statistics page [1]. We have used one aggregated demand of one month of daily traffic demands, from August $22^{nd}$ to September $21^{st}$ 2002. This traffic demand concerned 977 different destination ASes. This dataset has already been described in the previous chapter in section 4.8. We gathered four BGP routing tables from Oregon Route views [119] from four major Tier-1 providers from October 3 2002: AT&T (AS7018), Sprint (AS1239), Level3 (AS3356) and Verio (AS2914).

### 5.4.1 Traffic balance

Figure 5.2 presents the results of the optimization with the traffic balance objective with two, three and four providers. For the case with two providers, we confront AT&T and Sprint. For three providers, we confront AT&T, Sprint and Level3. The x-axis of the graphs of Figure 5.2 shows the number of generations required to obtain the smallest value of the objective function. The y-axis gives the value of the objective function *max(traffic(i))*, $i = 1, .., 4$ where *traffic(i)* denotes the amount of traffic exchanged with provider $i$. The total traffic of the one month traffic demand corresponds to about 640 units, one unit being one billion of bytes with the one over one thousand packet sampling performed by Netflow. So the unit corresponds roughly to one terabytes of traffic.

Figure 5.2 illustrates the effect of two parameters: the number of providers and the AS sink tree granularity. First let us deal with the number of providers. The default solution found by the BGP decision process appears for generation zero. As the number of providers increases (middle and bottom graphs), BGP does not provide a good traffic balance among all the available providers when the default values of the route attributes are used, with a distance between the optimum and the BGP solution does not decrease when the number of providers increases. Increasing the number of providers a stub would have thus does not ensure that the traffic balance among these providers shall be good, due to the biased choice of the tie-breaking rules of the BGP decision process. Furthermore, the more providers a stub has, the bigger this stub is generally speaking. So a bigger stub is due to have increased needs for traffic engineering. Figure 5.2 shows that a larger number of generations (hence of BGP filters) is required for the algorithm to converge towards the optimum: twenty generations for two providers, a little less than one hundred and fifty generations for three providers and more than one hundred and fifty generations for four providers. Since the number of generations provides an upper bound on the number of BGP filters, we see that a good traffic balance for a small number of BGP filters can be obtained with a few BGP filters only in the case of two providers. This larger number of BGP filters required to improve the traffic balance has two causes. The first cause is the inability of BGP to balance the traffic because of the tie-breaking rules of the BGP decision process that systematically favors one of the upstream providers. The second cause is the distribution of the traffic for the interdomain destinations. We know from Chapter 3.5 that the cumulative percentage of the traffic for interdomain destinations requires more and more destinations to capture increasing percentages of the total traffic. This means that the first few BGP filters are able to move a significant amount of traffic from the default upstream provider found by BGP to the one providing the largest gain in the objective function. Increasing BGP filters on the other hand work on smaller amounts of traffic so that the improvements in the objective function become smaller and smaller. So this behavior does not depend on the optimality of the search technique, but only on the distribution of the traffic for the interdomain destinations. Henceforth we do not expect the optimal BGP filters to provide a significant gain compared to the solutions found by our evolutionary algorithm.

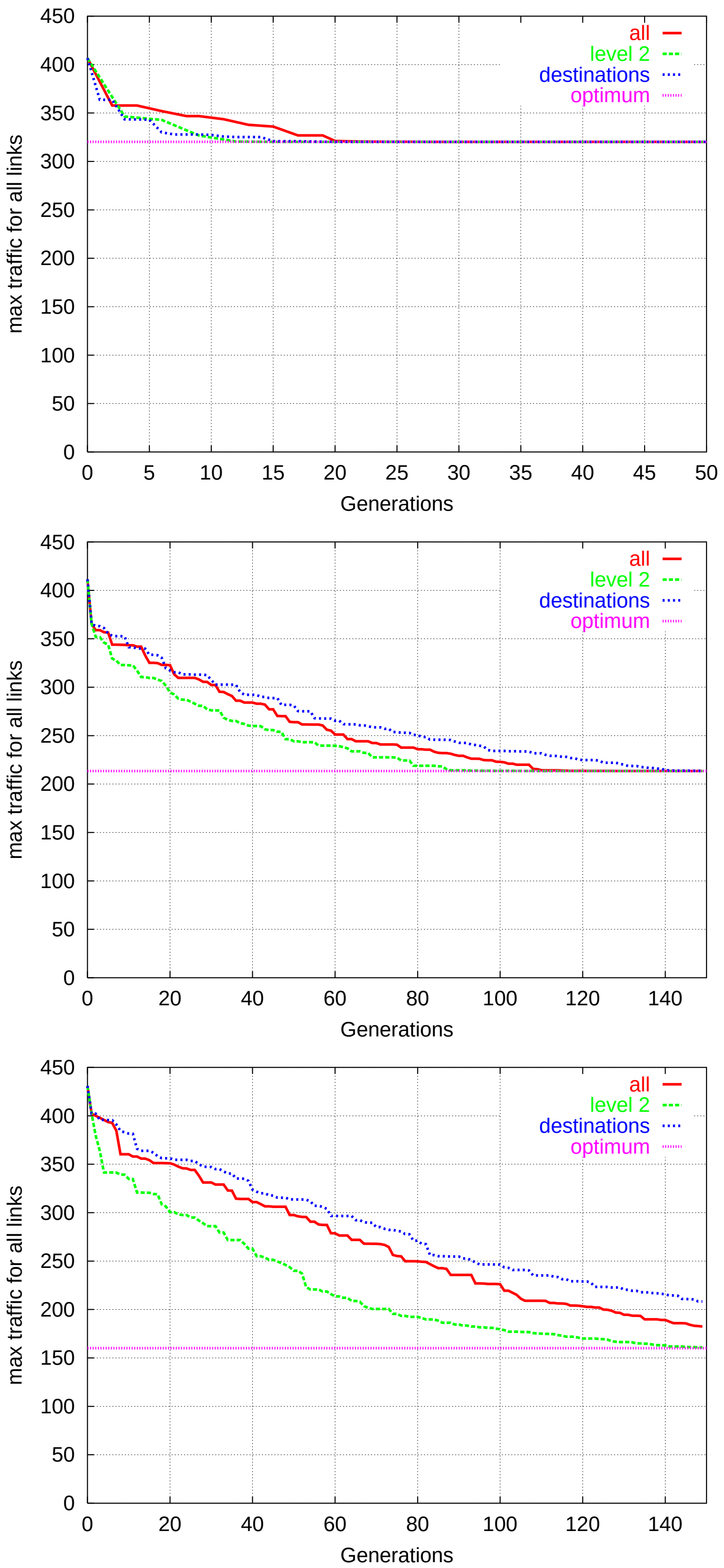



Figure 5.2: Simulations for traffic balance: two providers (top) , three providers (middle) and four providers (bottom).

The other parameter studied in Figure 5.2 is the AS sink tree granularity. Each graph of Figure 5.2 compares three different ways to leverage the aggregation of the traffic destinations with the AS sink trees. The first way consists of allowing all ASes seen in the AS paths for all traffic destinations to be an AS sink tree. This means that we allow AS sink trees to be anywhere in the AS-level topology. This corresponds to the curves labeled "all" on the graphs of Figure 5.2. There are then respectively 1075, 1112, 1118 and 1125 different AS sink trees for the simulations with one, two, three and four providers. The second case corresponds to destination ASes only, where one restricts the AS sink trees to the leaves of the AS-level topology only. There are 977 AS sink trees in that case corresponding to the curve labeled "destinations". The last traffic granularity we study is a heuristic based on our knowledge of the AS-level topology. The coarsest aggregation possible for AS sink trees is the one of the upstream providers themselves, but this is not due to provide a sufficiently fine-grained granularity at the traffic-level to allow optimal load balancing. Because most of the traffic destinations for interdomain traffic are located two to four AS hops away from a stub ISP [66, 174], aggregating traffic destinations one AS hop behind the upstream providers is due to provide an important aggregation of the destinations on the AS sink trees while still a fine traffic granularity.

Figure 5.2 confirms this intuition by showing the fastest decrease of the objective function for a given number of BGP filters to be applied. There are respectively 80, 155, 185 and 225 AS sink trees at a distance of two AS hops (curve labeled "level 2"). This search space is therefore also the smallest among the three AS sink tree aggregation levels. All simulations of Figure 5.2 start the search with the traffic distribution obtained by using the default BGP routing, the traffic distribution when no BGP filter is applied to the routes.

What we see when comparing the three curves of Figure 5.2 is that the AS sink trees located at two AS hops provide the smallest value of the objective function for few BGP filters, confirming our intuition that these AS sink trees constitute a good balance between the number of AS sink trees and their traffic granularity. The important thing to bear in mind when reading this figure is that the main difference between the three curves comes from the traffic aggregation for a limited number of BGP filters. For a relatively small number of BGP filters, it is the amount of traffic that can be moved by BGP filters that counts. We see on Figure 5.2 that the "level 2" aggregation works better than the other two most of the time. We also see that the "all" aggregation also works better than the "destinations" aggregation. The explanation is that the "level 2" aggregation is the coarsest, then comes the "all" aggregation, then the "destinations". It must be noted that the "all" and "destinations" aggregation levels both suffer from quite large search space compared to the "level 2" aggregation. Both the "all" and "destinations" aggregation levels would need a larger population size in order to be able to find better solutions for a few BGP filters, but this would take too much time when the number of filters is several times larger than for the "level 2" aggregation.

The main issue with the multi-objective problem is that using a heuristic based on the amount of traffic that the traffic aggregate is able to move provides good results for a limited number of BGP filters. The issue with biasing the search towards potential large traffic moves is that this heuristic works well when there are large differences between the amount of traffic among the upstream providers. When the largest traffic moves have been performed by the first few BGP filters on the other hand, the objective of the search is to find how the small traffic destinations can be piled up to further reduce the imbalance between the upstream providers. When one upstream provider receives substantially more traffic than the others, the algorithm has just to find the largest traffic aggregate and move it to the least loaded among the other upstream providers. While the differences in the amount of traffic received by each upstream provider become relatively small, there are many ways to move several small traffic aggregates to reduce the traffic disbalance. In that case, biasing the search towards large traffic aggregates slows down the search rather that speed it up. One must bear in mind that including domain-specific information that works well on average in the search heuristic will bias sooner or later the search. In our case, biasing towards large traffic aggregates speeds up the search for the first few BGP filters only.

### 5.4.2 Cost of traffic distribution

Now we complexify the task for our algorithm by providing a different cost function for the traffic exchanged through each provider, each traffic unit exchanged through a given provider having a fixed cost (volume-based billing). Such a cost function is more complex than the traffic balance objective because improvements depend not only on the amount of traffic that is switched from some provider to another but also on the relative cost of the traffic for the providers.

Figure 5.3 presents the simulation results with two (top), three (middle) and four (bottom) providers. Each graph of Figure 5.3 shows the best solutions found by the algorithm at each generation, expressed as a percentage of the cost of the solution found by the default BGP routing. For each graph we provide simulations for the three traffic aggregation levels and two different cost settings of the providers. The cost of each provider is provided on the legend of the graphs.

The two values of the relative cost for the two providers simulations (top of Figure 5.3) illustrate the important points about the convergence of the algorithm. First, the difference between the cost of the traffic for the two providers must be sufficiently important in order to achieve a non-negligible gain (defined as the percentage in cost reduction compared to the BGP solution) with a small number of BGP filters. For instance, with a cost of two for the first provider and one for the second provider (denoted by (2,1) on the labels of the top graph of Figure 5.3), the total gain after 140 generations is about twenty-three percent for all AS sink trees while about seventeen percent for the "level 2" aggregation. For the (4,1) cost setting, the gains are a little under forty percent for the "all" AS sink trees aggregation and a little under thirty percent for the "level 2" aggregation.

The second thing we can notice concerns the quality of the solutions found by the three AS sink tree aggregation levels. The three aggregation level exhibit a similar behavior as in the traffic balance objective, with a better solutions found by the

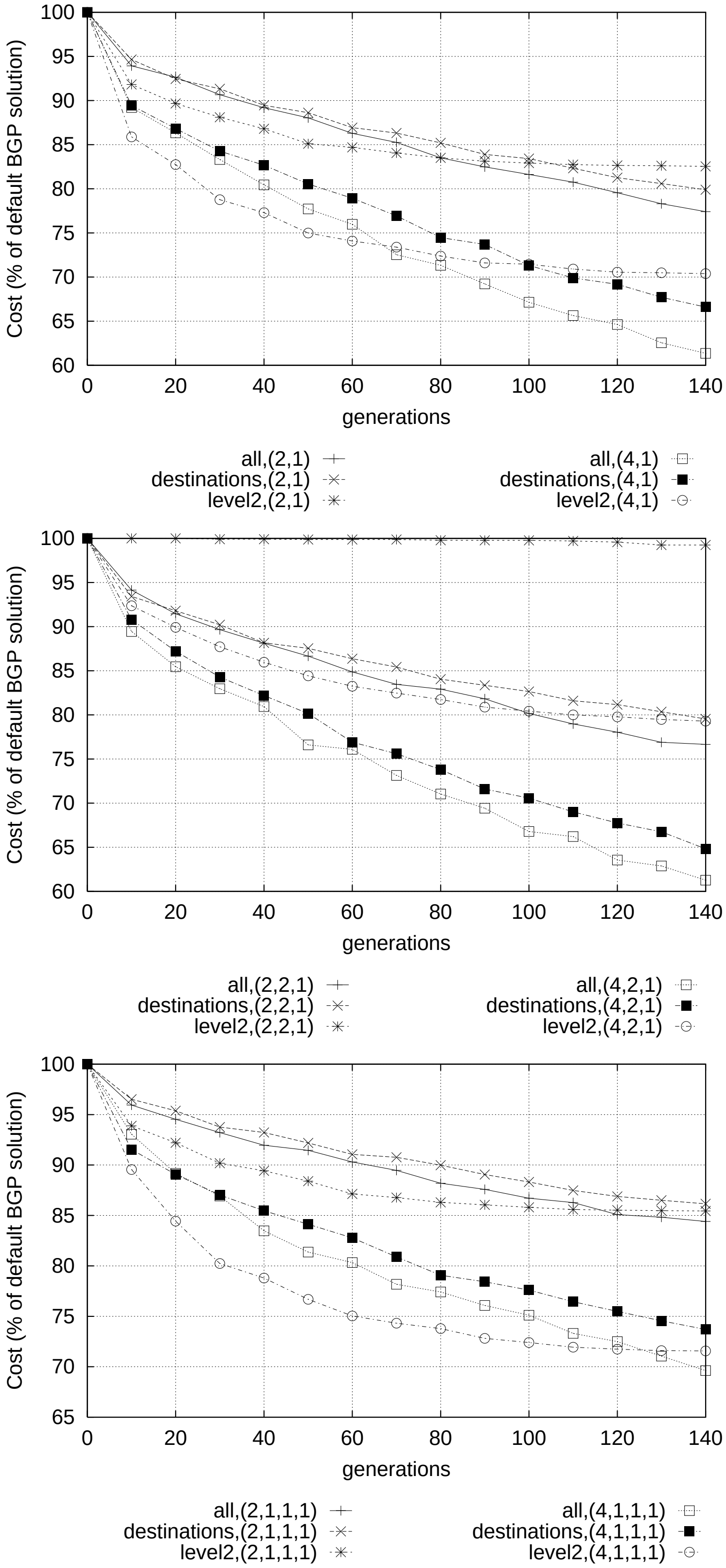



Figure 5.3: Simulation for cost function: 2 providers (top), 3 providers (middle) and 4 providers (bottom).

"level 2" aggregation for a small number of filters while for a large number of BGP filters the "destinations" aggregation provides the best results. For a small number of BGP filters, the most important factor is the amount of traffic that can be moved in one BGP filter from one upstream provider to another. For a large number of BGP filters on the other hand, the "level 2" aggregation is stuck in its search because it works with too large traffic aggregates. The "all" aggregation has a large search space than the "destinations" aggregation hence its improvements are slower than the "destinations" aggregation. The "level 2" aggregation works with larger traffic aggregates and in a smaller search space, hence it is quite efficient for a limited number of BGP filters. The other two aggregation levels on the other hand work in larger search spaces so are slower but they work on the finest traffic aggregation possible so they are not stuck in their search for a sufficiently large number of BGP filters.

As shown on the bottom graphs of Figure 5.3, the simulations with four providers give gains similar to the two providers case. When one provider has a cost significantly larger than all others, the algorithm can find improvements as in the two providers case. When the cost setting is more complex however, moving the traffic on a "level 2" AS sink tree has a non-trivial impact on the improvement in the cost function because several destinations are involved. When one provider has a larger cost than all others, switching a "level 2" AS sink tree towards a smaller cost provider is certain to reduce the total cost of the operation since some traffic goes from this most expensive provider to less costly ones while the other destinations whose traffic change between equal cost providers do not increase the cost. When the cost setting is more complex, moving some traffic to a less costly provider may also move some other traffic from a still less costly provider to a more costly one, making the net result of the change difficult to predict.

An illustration of the main disadvantage of the "level 2" AS sink trees appears in the simulation with three providers and the (2,2,1) cost setting. This simulation is shown to be stuck in a local optimum because of the difficult cost setting that prevents the algorithm to improve the search by moving a large traffic aggregate among providers. It is only if the relative costs among providers are different that one can be sure that large traffic aggregates will allow improvement in the cost function with a single BGP filter. If we had allowed the algorithm to apply several BGP filters at each generation, then this could have prevented the algorithm to be stuck for the "level 2" AS sink trees. In practice, it suffices that the largest traffic aggregates be reachable through upstream providers having the same cost to slow down the improvements of the search. Note that the other simulation with three providers and the "level 2" AS sink trees performs also significantly worse than the other two AS sink tree aggregations, confirming that the particular settings with three providers we used are responsible for the bad results of the "level 2" AS sink trees.

## 5.5 Conclusion

We presented in this chapter an evolutionary algorithm that tries to minimize an objective function by finding the successive BGP filters to be applied on the BGP routes. We evaluated this algorithm on two objective functions and a real interdomain traffic demand and real BGP routing tables. We studied the behavior of our algorithm for three parameters: the number of providers, the AS sink tree aggregation for the interdomain traffic and the type of objective function.

Our simulations showed that the default way BGP performs traffic balancing between providers is poor, and it does not improve as the number of providers increases due to the tie-breaking rules in the BGP decision process. We showed however that improving the traffic balance compared to the default BGP solution is not a difficult problem. However, when the number of upstream providers increases, more and more BGP filters are required to properly balance the traffic.

For a cost objective, our simulations showed that the gain of the optimization depends a lot on the relative costs among the upstream providers. We showed that reducing the cost of the interdomain traffic by a significant percentage can be achieved with a limited number of BGP filters.

# Chapter 6

# On-line interdomain traffic engineering with BGP

The current state-of-the-art in interdomain traffic engineering is primitive [16]. While ISPs know their internal topology and techniques exist to set the IGP costs [76], most of them rely on manual tuning to appropriately set their IGP weight. At the interdomain level, matters are even worse. Operators change their routing policies and the BGP attributes of the routes on a manual basis without a proper understanding of such changes on the flow of the traffic. Many problems arise due to misconfigurations in the routers [109]. The current practice in BGP-based traffic engineering is often "trial-and-error", i.e. an operator changes the BGP attributes of some routes that were observed to carry a large amount of traffic and observes the effect on the interdomain traffic. For large transit ISPs, interdomain traffic engineering is a complex problem even for outbound traffic due to interactions between BGP and the IGP. In the case of stub ASes on the other hand, the reason for the absence of a proper engineering of BGP is mainly a lack of understanding of the working of BGP and its effect on the traffic. The aim of this chapter is to demonstrate that properly tuning BGP to optimize the outbound traffic of a stub AS is not an issue, even on relatively short timescales of a few minutes.

### 6.0.1 Problem statement

The goal of the traffic engineering technique presented in this chapter is to track the optimal distribution of the interdomain traffic on timescales of minutes under the constraint of minimizing the number of BGP configuration changes. The two objective functions are thus:

1. minimize the number of BGP route changes,
2. optimize an objective function defined on the outbound traffic sent to the BGP neighbors.

The first objective concerns BGP routing. Given that there are many remote networks with which a stub AS exchanges traffic on timescales of minutes to days [137], one cannot require that engineering the interdomain traffic needs to influence the traffic of a too large fraction of these remote networks. A reasonable objective of any BGP-based traffic engineering technique should be to minimize as much as possible the burden on BGP. Tweaking the value of local BGP attributes has no impact on the BGP advertisements sent to BGP peers of a stub ASes.

The traffic objectives to be optimized by stub ASes might differ depending on their size and their main business. In this chapter, we use two representative objectives. The first is to equally balance the total traffic over the available providers, specified as $min(max(tr_i))$, $i = 1, ..., n$, $n$ being the number of available providers and $tr_i$ denotes the amount of traffic sent to provider $i$. Because traffic balancing might be too simple, we also evaluate the optimization of a second objective of balancing the cost-weighted traffic, specified as $min \sum_{i=1}^{n} c_i \times \frac{tr_i}{\sum_{i=1}^{n} tr_i}$ where $c_i$ denotes the cost of one unit of traffic sent to provider $i$. The minimum for the second traffic objective happens when $c_i \times tr_i = c_j \times tr_j, \forall i \neq j$. We thus call this second objective "cost-weighted" traffic balancing since at the minimum the values of the traffic of each provider weighted by its cost are equal among the providers. The second traffic objective models the volume-based billing performed by transit ISPs.

### 6.0.2 Context

The ASes we mainly focus on in this chapter are stub ASes that constitute the majority of the ASes in the Internet [157]. At the date of January 24 2004 more than 86 % (14337 over 16523) of the ASes were considered as stub-ASes by APNIC's BGP routing table [151]. Among these stub ASes, we focus on ASes connected to the Internet through several different providers. Because stub ASes do not offer transit service, they do not advertise via eBGP the routes they learn from their peers. In the case of a stub, there are no interactions between the local BGP route tweaking performed on the eBGP routes learned from the peers and the rest of the Internet.

Figure 6.1 plots the distribution of the number of providers stub ASes have according the the BGP tables gathered from several vantage points on December 12 2003 [157]. On Figure 6.1, we only consider the ASes from the gathered BGP routing tables that were not considered as provider for any of their peering according to the heuristic proposed in [157]. According to the data of [157], 14287 ASes were considered as stub ASes. Among these stubs ASes, 5671 (40 %) are single-homed and 6748 (47 %) dual-homed. The average number of providers (distinct ASes) per stub is 1.87. These multi-connected ASes need to distribute their interdomain traffic among several interdomain links. Almost all ASes have to pay large ISPs to get a global connectivity on the Internet. The cost of the traffic on these links can become important. These ASes would find interest in redistributing some traffic from more expensive providers towards less expensive ones or simply better balancing their traffic among the various available access links. Furthermore, the number of providers per stub is not expected to grow much in the future since adding providers has been shown to be of limited value [11].

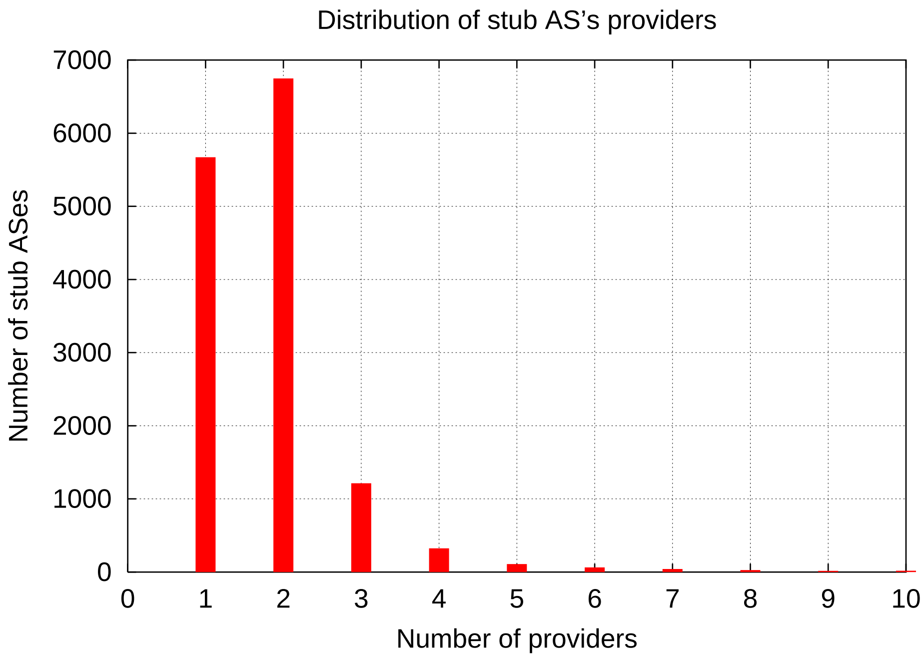



Figure 6.1: Number of providers for stub ASes.

The remainder of this chapter is structured as follows. In section 6.1 we discuss the difficulty of on-line traffic engineering with BGP. In section 6.2 we provide simulations to evaluate the feasibility of outbound on-line traffic engineering with BGP. Section 6.3 discusses extensions to our interdomain traffic engineering solution.

## 6.1 Difficulty of on-line TE

Most optimization problems dealt with in the literature are static ones [81, 125]. Dynamic optimization mainly works on continuous problems [34]. For discrete and combinatorial problems in dynamic environments, evolutionary algorithms and swarm optimization techniques are widely used [32, 39]. Unfortunately, no optimization technique is known to be able to deal with all dynamic problems. The specificities of each problem make some techniques more suited than others.

The problem we tackle in this chapter consists in controlling the traffic and its dynamics by tweaking the BGP routes. This problem is discrete and the control of the traffic is indirect, by tweaking the BGP routes. Furthermore, the dynamics of network traffic is important [129], hence tracking the traffic increases the difficulty of the problem.

In this chapter, we rely on an evolutionary algorithm (EA) to perform on-line traffic engineering. The choice of evolutionary algorithms lies in their ability to solve complex problems and particularly multi-objective optimization problems [49]. Because our problem is a dynamic multiple-objective one, a population-based heuristic makes the search easier as explained in the next sections.

The principles of our scheme are as follows. For each short-term time interval, a current BGP configuration drives the distribution of the outbound interdomain traffic. By BGP configuration, we mean through which provider the traffic towards some BGP prefix is sent. A BGP configuration can be equivalently expressed as the attributes of the BGP routes of the local AS. During each time interval, the algorithm must find a non-dominated front of solutions (with respect to the objectives defined in section 6.0.1) that will be used to choose the BGP configuration to be applied for the next short-term time interval. There are three main issues to be tackled during the design of the algorithm: 1) tracking the traffic distribution for the next time interval, 2) computing the Pareto front and 3) choosing the next BGP configuration among the solutions from the non-dominated front.

Let us deal with the third issue right now. As will be shown later on, the choice of the solution to use during the next time interval among the possible solutions on the non-dominated front is an issue only whenever large trade-offs exist between the number of BGP route changes and the value of the traffic objective. It will be shown in the next sections that even if these two objectives are conflicting, allowing additional BGP route changes provides a decreasing marginal gain in the traffic objective. Henceforth, a very limited number of BGP route changes are required in practice and network operators will only have to choose the upper bound on the number of BGP advertisements allowed.

### 6.1.1 Tracking the traffic

One of the issues to be tackled by the on-line traffic engineering scheme is the tracking of the amount of traffic that will be sent through each provider during the next time interval. Network traffic is known to be bursty (see [129] and references therein), so that correctly predicting the amount that will be sent through any provider will be critical to be able to find a good BGP configuration. Note that traffic provisioning schemes like [63, 101] are intended to limit the resource over-provisioning, they are not aimed at tracking the traffic dynamics. For the algorithm to work in most practical situations, not having to rely on a particular model for the data is desirable, since different stubs may have very different user profiles and thus different traffic types.

Several authors have studied the predictability of network traffic [145, 134]. [145] studies the multi-step ahead predictability of network traffic relying on the ARMA and MMPP models, to evaluate the possibility of multi-step ahead prediction for traffic control. The main result of [145] is to show that smoothing and traffic aggregation both help the prediction. [134] studies the multiscale predictability of network traffic, for one step ahead prediction. [134] confirms the results of [145] concerning the benefits of smoothing the time series. However, [134] also finds by studying many different traffic traces that the predictability of network traffic highly depends on the considered trace. [134, 145] both evaluate the performance of predictability based on the ratio of mean squared error to variance. The results of [134] indicate that the performance of different predictors does not improve much with their increasing complexity. Relying on the last value of the time series or a simple moving average is as efficient as AR, ARIMA, and ARFIMA models [31]. Since smoothing might provide some benefits while complex predictors do not, we compare in this chapter the performance of adaptive exponential smoothing with a simple predictor: the previous value.

[36] compared several versions of adaptive exponential moving averages for bandwidth prediction. The authors of [36] showed that Internet traffic could be predicted with an adaptive EMA algorithm: the low pass EMA (LpEMA). The basic idea is to modify the classical EMA formula

$$e_i = (1-\alpha) \times e_{i-1} + \alpha \times tr_i \tag{6.1}$$

where $e_i$ is the estimate at time $i$, $\alpha$ the weight of the moving average and $tr_i$ the traffic at time $i$. Instead of a fixed weight $\alpha$, the weight is made adaptive and computed as follows:

$$\alpha_i = \alpha_{max} \times \frac{1}{1 + \frac{|m_i|}{m_{norm}}} \tag{6.2}$$

where $\alpha_{max}$ is the maximum weight, $m_i$ is the gradient of the traffic at time $i$ $(\frac{tr_i - tr_{i-1}}{t_i - t_{i-1}})$ and $m_{norm}$ a normalizing gradient. In this chapter, we computed $m_{norm}$ as the mean gradient over a time window of one hour (six 10 minutes intervals). We chose to rely on a time granularity of 10 minutes in this chapter.

Let us now have a closer look at the working of the LpEMA predictor. First, we compute $m_{norm}$ as the mean gradient (in absolute value) over these last 6 time intervals. The adaptive EMA weight, $\alpha_i$, is computed according to Formula 6.2. The name Low pass EMA is due to the behavior of $\alpha_i$ that reduces the impact of large changes in the traffic by smoothing out the time-series, having as reference the mean gradient as an indication of the mean variability of the time-series. If on the other hand relatively limited changes in the signal occur during the current time interval, then the value of $\alpha_i$ comes closer to $\alpha_{max}$ and the EMA better tracks the short-term changes in the time-series. The rationale behind this adaptability is that large changes in the traffic should not be followed unless they are longer-term trends. The greater adaptation of the EMA whenever the shifts in the traffic correspond more closely to the average behavior of the signal (mean gradient) are meant to follow these short-term variations that better match the dynamics of the traffic. The main reason for the smoothing behavior of the LpEMA concerns the fact that Internet traffic is bursty, hence tracking large shifts in the traffic is useless for prediction purposes. Unless these changes are longer-term trends, adapting the prediction based on traffic shifts will merely destabilize the prediction while not help in predicting such random traffic surges.

To assess the benefit of the LpEMA predictor to track the traffic of each provider, Figure 6.2 provides both the mean and the standard deviation of the prediction error for different values of $\alpha_{max}$ and each provider. On Figure 6.2, the prediction error has been computed for each time interval as the relative error between the predictor and the true value of the traffic. Each graph of Figure 6.2 plots the mean and standard deviation of this prediction error over the 6 days of the trace. Both the mean and the standard deviation of the error for the LpEMA predictor exhibits a first decreasing part but increase after some value of $\alpha_{max}$. The "last value" predictor has a lower mean error than the LpEMA one (no matter the value of $\alpha_{max}$) for the three providers. The standard deviation of the "last value" predictor is not always lower than the one of the LpEMA predictor. Only for provider 2 has the "last value" predictor a smaller mean and standard deviation than those of the LpEMA predictor for all values of $\alpha_{max}$. The behavior of the LpEMA predictor can be explained in the following way. Increasing the value of $\alpha_{max}$ first allows the predictor to adapt to the natural variability of the traffic. Then, as the value of $\alpha_{max}$ increases beyond some value, both the mean and the standard deviation of the error increase. This is due to the greater adaptability of the LpEMA prediction for increasing values of $\alpha_{max}$.

Figure 6.2 suggests two things. First there is no benefit to rely on the LpEMA predictor since its mean error is always larger than the one of the last value. Second, the ideal value of $\alpha_{max}$ seems to depends on the particular provider, hence it is also due to depend on the particular traffic trace. This section teaches that complex and adaptive techniques do not to provide any benefit in terms of the accuracy in the traffic prediction. The last value predictor already constitutes an accurate predictor, whose performance is comparable to an adaptive exponential moving average, both in its average and standard deviation. The simulations of section 6.2.2 will confirm that the last value predictor performs better than the LpEMA one when used with the traffic engineering scheme.

### 6.1.2 Searching the Pareto front

Once the method used to predict the traffic distribution during the next time interval has been chosen, we need to choose how to search for the changes in the BGP routes that will optimize the traffic objective while limiting these BGP route changes as much as possible. For each time interval, the algorithm must find the BGP configuration that is more likely to optimize the traffic objective, given that we assume the predictor to provide a reliable estimate of the traffic distribution. The objective to find, during each time interval, a non-dominated front representing the trade-off between the number of BGP route changes to be applied and the value of the traffic objective. The main issue with these two objectives is their conflicting nature: the more BGP route changes allowed, the better the expected value of the traffic objective. Changing BGP routes modifies the amount of traffic sent over each provider at each time interval. Increasing the number of BGP route changes will not always improve the traffic objective due to a changing traffic pattern. If some traffic from a previously congested provider has been moved to another (previously uncongested) provider, it could be necessary to undo the concerned traffic move to improve the traffic objective if congestion has now appeared on the previously uncongested provider.

We have several options concerning how the search can take place. We can start at each time interval with the last BGP configuration and search for the optimal one and sample the Pareto-optimal front for the next time interval. Another option is to start with the default BGP configuration (no BGP route change) and find the

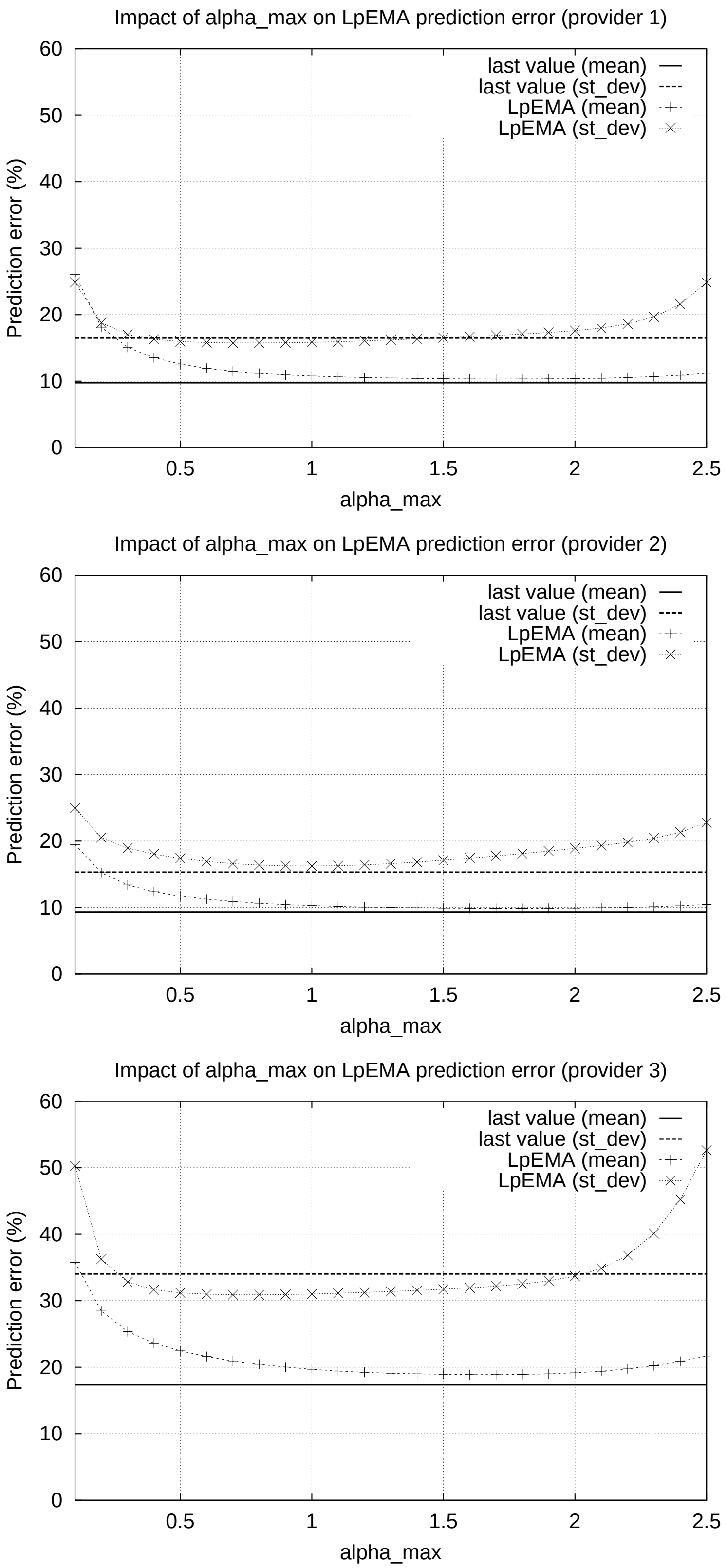



Figure 6.2: Effect of $\alpha_{max}$ value on prediction error: provider 1 (top), provider 2 (middle) and provider 3 (bottom).

Pareto-optimal front from scratch for each time interval. Finally, we can also start with the last Pareto-optimal front in search for the new Pareto-optimal front.

The first option has the drawback of requiring to potentially undo BGP route changes that are not optimal anymore due to changes in the traffic patterns. By having to start with a BGP configuration that already contains BGP route changes, there is a risk of having to remove some BGP route changes that were good for the last interval but need to be removed for the current time interval. To illustrate this issue, Figure 6.3 shows the two Pareto-optimal fronts for two consecutive time intervals, $n-1$ and $n$. The x-axis of Figure 6.3 represents the number of BGP route changes and the y-axis gives the value of the traffic objective of the best possible solution having a given number of BGP route changes. Figure 6.3 shows the two Pareto-optimal fronts for the two time intervals, containing both the default BGP solution (0 BGP route change) and the BGP configuration chosen at each time interval. Suppose that we have to start the search at the BGP configuration at time $n-1$, containing a few BGP route changes. To join the Pareto-optimal front for time interval $n$, we need to remove some BGP route changes from the BGP configuration at time $n$. On Figure 6.3, we show the image at time $n$ of the BGP configuration at time $n-1$. Because the traffic pattern changes, a good solution during the last time interval might have a relatively large value of the traffic objective during the next time interval. This issue however is theoretical. We will show in the next sections that this first option is the one that is the most efficient in terms of the number of required BGP route changes.

Figure 6.3 for instance shows that the search path between the BGP configuration at time $n-1$ and the Pareto-optimal front at time $n$ requires to remove BGP route changes. The effect of removing these BGP route changes is to improve the traffic objective, but removing BGP route changes might worsen the traffic objective as well in practice. Finding the path between the last BGP configuration and the current Pareto-optimal front consumes computational time that will not be spent on searching for the Pareto-optimal front of time interval $n$. Additionally, the search would have to allow any intermediate solution to add or remove BGP route changes, implying a larger search space compared to a search that only adds BGP route changes.

In the case of the second option, we start from the default BGP solution without BGP route change, and build the Pareto-optimal front from scratch for each time interval. This solution has the advantage of limiting the size of the search space because the algorithm can iterate by only adding successive BGP route changes to find the Pareto-optimal front. The drawback of this method is that one does not leverage the knowledge of potentially good solutions found during previous time intervals.

The last option consists in building the next Pareto-optimal front by starting from the last one. This solution has the same potential drawback as the first option, in that the size of the search space is very large because the search must allow moves backward and forward with respect to the BGP route changes. The most annoying problem is to work with a non-dominated set of solutions during the search of the Pareto-optimal front. This requires that the search effort be distributed over the

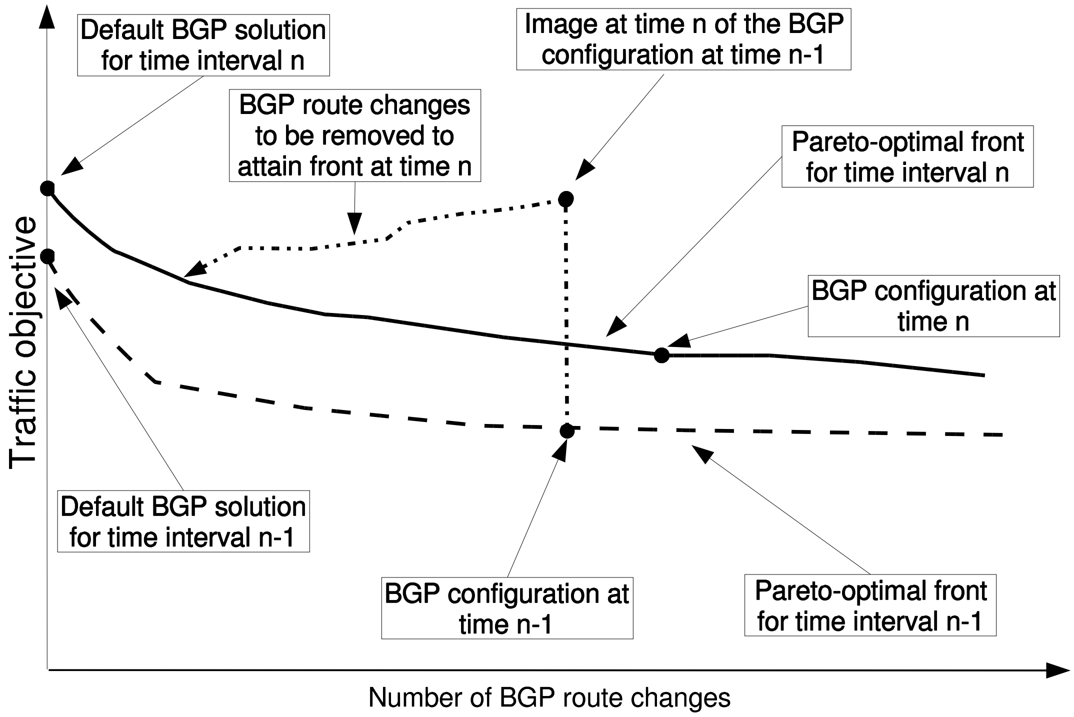



Figure 6.3: Search of the Pareto front from one time interval to another.

different points of the front. Some of the points of the old Pareto-optimal front however will not easily join the new Pareto-optimal front, so that valuable computational time may be wasted. pa We experimented the three options to determine which search heuristic algorithm to choose. Our simulations indicated that allowing to undo BGP route changes and starting with a Pareto-front when searching for the next front had problems to sample the Pareto front. In the remainder of the chapter, we do not allow to undo BGP route changes and always start with a single solution at each time interval in search of the Pareto front.

### 6.1.3 Search algorithm

The previous section discussed the choice of the search method. In this section, we describe in more details the working of the search algorithm. Figure 6.4 provides the pseudo-code describing the main operations during one particular time interval. In line 2 of Figure 6.4, we initialize the current population at the default BGP solution (0 BGP route change). Line 4 of Figure 6.4 contains the main loop that iterates over the population and adds BGP route changes one at a time. The loop on line 4 of Figure 6.4 iterates from the first BGP route change until *max_iter* BGP route changes, the maximum number of BGP route changes allowed. Within an iteration, the algorithm performs a local search by adding an additional BGP route change to the individuals of the population *pop* (line 6 of Figure 6.4). Individuals who were added a BGP route change that improve their traffic objective value are put in *improved_pop*. Then, this population of improved individuals is used to update the Pareto-optimal front of the current time interval (line 8 of Figure 6.4). At most one individual is added to the Pareto front at each iteration since each successive generation deals with a population that contains *iter* BGP route changes. Finally, selection is performed in line 10 of Figure 6.4 where individuals from *improved_pop* are chosen to make the population for the next iteration (line 12 of Figure 6.4).

```
1 // Population ← default BGP solution
2 init_pop(pop)
3 // Each iteration adds a BGP route change
4 foreach iter = 1 to max_iter {
5   // Trying to add a BGP route change
6   improved_pop = try_add_filter(pop)
7   // Updating current front
8   update_front(improved_pop)
9   // Selecting individuals
10  new_pop = select_population(improved_pop)
11  // Replacing old population by new one
12  pop = new_pop
13 }
```

Figure 6.4: Search algorithm for one time interval.

In the pseudo-code of Figure 6.4, there is no explicit mention to the minimization of the BGP route changes. Actually, the number of iterations *iter* is a higher bound on the number of BGP route changes since the algorithm might be unable to find an improved solution at some iteration. Making the objective in terms of the number of BGP route changes a dimension of the search provides an explicit control on the number of BGP route changes that are required by the solutions found by the algorithm.

The performance of the search algorithm is sufficient to be used as an on-line traffic engineering scheme. The current prototype Perl version of the algorithm takes about one second per iteration on a P4 2.4 GHz. A C version would probably be at least an order of magnitude faster.

## 6.2 Performance evaluation

This section provides results about the performance of the on-line interdomain traffic engineering scheme introduced in the previous sectionds. Section 6.2.1 first presents the typical context in which the scheme is to be used. Section 6.2.2 then discusses the impact of the traffic variability on the performance of the scheme. Section 6.2.3 proposes a modification of the basic on-line traffic engineering scheme to reduce the number of required iBGP advertisements. Section 6.2.4 then goes on to show that relying on the MED attribute of the BGP routes performs almost as well as relying on the `local-pref-on-line` attribute. Finally, section 6.2.5 evaluates the performance of the scheme for the second traffic objective introduced in section 6.0.1.

### 6.2.1 Scenario

The scenario of our simulations consists of a stub AS connected to three would-be providers, as illustrated by Figure 6.5. The local stub has one physical link with each provider over which a BGP session is established. In addition, the traffic engineering scheme centralizes the BGP routes announced by the BGP router of each provider and collects traffic statistics at the granularity of the BGP routes. The collection of traffic statistics could be performed in various ways (see [105] and references therein). The on-line traffic engineering algorithm runs on the machine labeled "optimization box". BGP routers `R01`, `R02` and `R03` each have a BGP session established with a different provider. The "optimization box" has eBGP multihop sessions established with the border BGP routers of each provider. It also receives the BGP routes from the providers and runs the algorithm to find how to tweak the BGP routes. Finally, the "optimization box" modifies the `local-pref` attribute of the routes in its BGP routing table and advertises these modified routes to the local routers `R01`, `R02` and `R03`. In practice, the "optimization box" would be a modified route reflector implementing the functions required for the on-line traffic engineering.

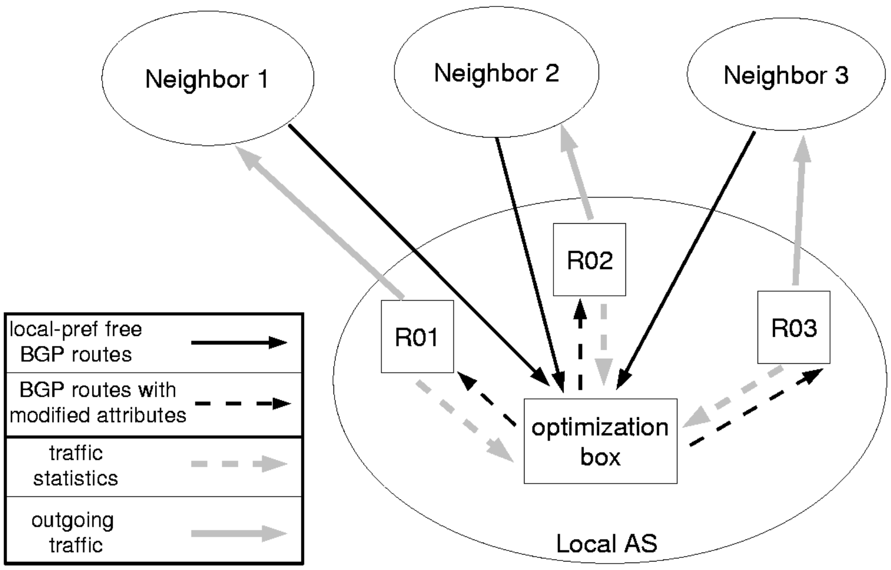



Figure 6.5: Simulation scenario.

The iBGP messages exchanged between the "optimization box" and each of the border BGP routers during each time interval are composed of two different kinds of updates. The first type of iBGP updates are sent by the "optimization box" to each of the border BGP routers to undo the route tweaking performed during the previous time interval. The second kind of iBGP updates are those concerning the BGP routes that are tweaked (preferred) for the next time interval. In the remainder of this chapter, we always speak in terms of the total number of iBGP messages exchanged between the "optimization box" and each border BGP router. This means that all figures that show the value of some traffic objective as a function of the number of iBGP updates in this section contain an even number of iBGP updates since for each new BGP route to be tweaked by the algorithm, there must be another iBGP update to reset a BGP route tweaked during the previous time interval.

The first traffic trace used in our simulations is a six days trace of all the outbound traffic from UCL, which we call *stub1* in the remainder of this chapter, starting from March $19^{nd}$ 2003 00:00 CET. During this six days period, UCL sent 1.7 terabytes of traffic or an average of 27 Mbps. This six days long trace was cut into time intervals of ten minutes. For each ten minutes interval, we recorded the amount of bytes sent to each BGP prefix.

The second traffic trace is an 18 days long trace of all the outbound traffic from BELNET, which we call *stub2* in the remainder of this chapter, a gigapop stub with 5 Gbps of capacity with its providers, starting from October $30^{th}$ 2003 12:00 CET. The trace was collected using the NetFlow sampling available on the access routers of *stub2* with a 1/4000 packet sampling rate. Figure 6.6 shows the evolution of the normalized outbound traffic of *stub2* during the 18 days of the trace. One unit on the y-axis of Figure 6.6 corresponds to the average traffic over the whole duration of the trace. Since packet sampling was used, we multiplied the byte count per second of the sampled trace by 4000 to obtain the expected total traffic. The purpose of the trace from *stub2* is to evaluate the benefit of on-line traffic engineering for larger stub ASes. Several traffic disruptions appear on the graph of Figure 6.6, due to maintenance operations on the Netflow collector. Although no actual traffic disruption occurred on the routers of *stub2* during that time, these periods can be considered as shutdown periods for the on-line traffic engineering technique.

In addition, we gathered routing information bases from Oregon Route views [119] dating from April $2^{nd}$ 2003 (for the *stub1* trace) and October $30^{th}$ (for the *stub2* trace) to simulate the BGP routing tables of the would-be providers to which our stub would be connected. While it is known that BGP routes dynamics can be important for some prefixes, it has however been shown in [139] that BGP routes can be considered as stable over relatively long periods for traffic destinations, so we use one BGP routing table per provider for the whole duration of each trace. For *stub1*, the setting of the simulation was of 3 providers (AS1239, AS7018 and AS1668). For *stub2*, 3 providers have also been used (AS1239, AS3356[1] and AS1668). The impact of the choice of the particular BGP routing table used to simulate a would-be provider is limited, merely influencing the default traffic imbalance. [176] shows that for the optimization of a traffic objective, the number of available providers complexifies the search. Studying the impact of the choice of the particular provider on the default traffic distribution found by BGP is irrelevant in the context of this chapter since its purpose is to demonstrate the feasibility of on-line traffic engineering with BGP, not to quantify the expected gain of on-line traffic engineering.

[1]We could not rely on AS7018 for these time periods because it was not present in Route views RIB at that time.

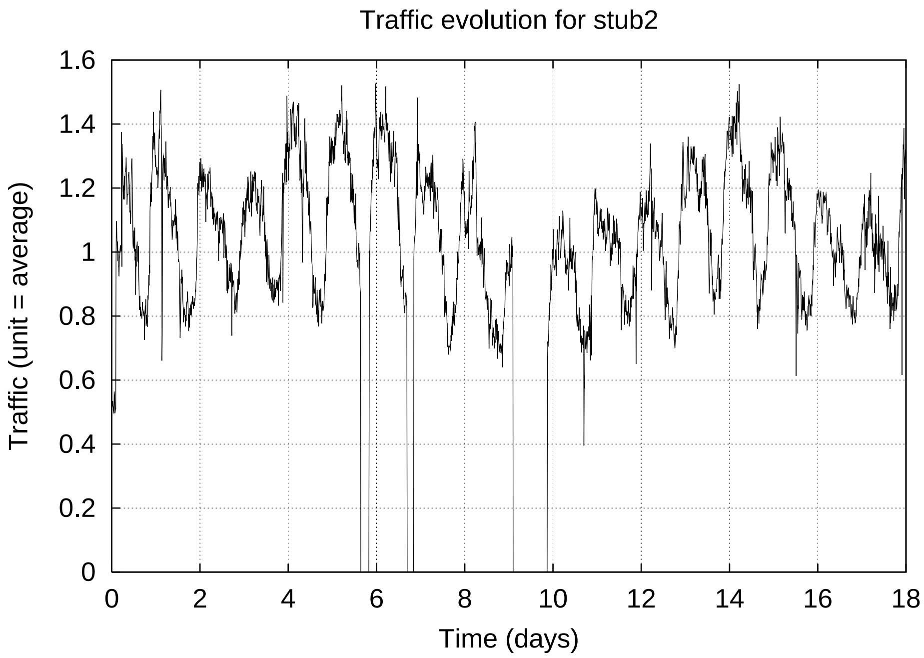



Figure 6.6: Evolution of *stub2* outbound traffic.

The performance evaluation has been carried out for both stub ASes but we present each result for one of them only.

### 6.2.2 Impact of traffic uncertainty

In this section, we study the effect of the impact of the predictors aimed at tracking the traffic dynamics on the quality of the achieved traffic balance. We also compare the results of the algorithm with and without the uncertainty of the traffic demands to understand to what extent the traffic dynamics prevents the algorithm to approach the optimal traffic balance.

Figure 6.7 plots the average traffic imbalance as a function of the number of iBGP updates allowed per 10 minutes interval. We call "imbalance ratio" the maximum amount of traffic sent through any of the three providers divided by the ideal traffic balance (total traffic divided by the number of providers). To compute the average, we took the whole non-dominated front for each time interval, and averaged the imbalance ratio for the six days of the simulations. Because all values of the number of iBGP updates are not present in the non-dominated fronts for each time interval, whenever some value for a given number of the iBGP updates was not present we used the value of the imbalance ratio for the solution having the larger number of iBGP updates smaller than the current number of iBGP updates. For example, if some solution for 60 iBGP updates did not exist, then we used the solution with the larger number of iBGP updates smaller than 60.

The four curves of Figure 6.7 start with the same value of the average traffic imbalance of about 1.9. The default route choice by the BGP decision process chose to

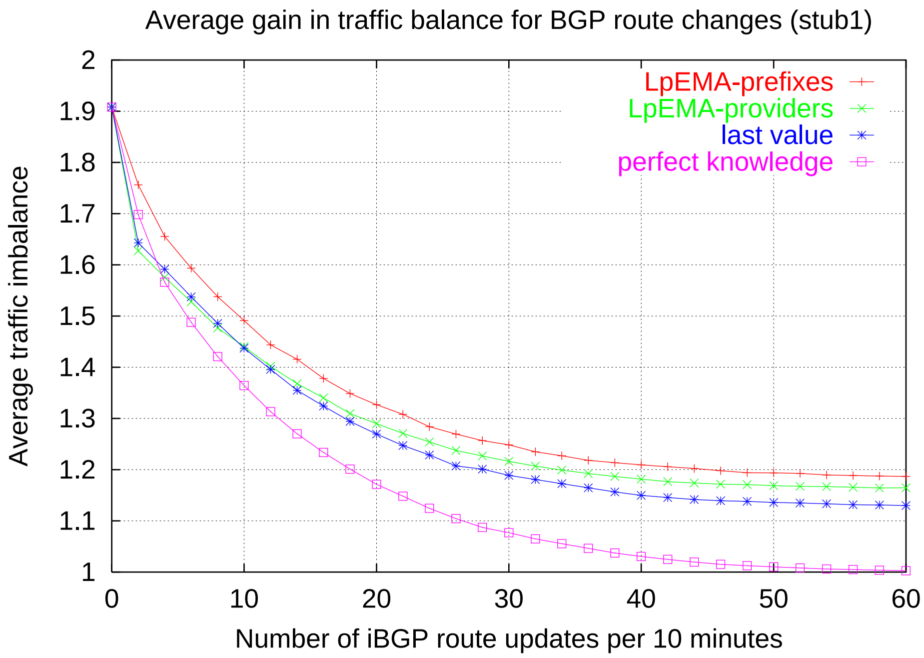



Figure 6.7: Average gain in traffic balance as a function of the number of iBGP updates.

send on average 1.9 times more traffic on one of the three providers than the average traffic over the three providers. With 10 iBGP updates, the average traffic imbalance is below 1.5 for all traffic tracking schemes. With 20, 30, 40, 50 and 60 iBGP updates, the average traffic imbalance is respectively below 1.33, 1.25, 1.21, 1.20 and 1.19. So for all practical purposes, relying on more than 40 iBGP updates provides a very limited improvement for the traffic balancing objective. The difference in the average traffic imbalance between the "last value" predictor and the prefix-based LpEMA is of about 0.05. This section confirms the results of Section 6.1.1 showing that the LpEMA predictor was not better than the last value. Even under perfect knowledge of the future traffic demand, the improvement provided by more than 40 iBGP updates is of at most 0.03. Relying on many iBGP updates to try to improve the traffic objective will not provide a significant gain.

The number of iBGP updates required for the optimization is small compared to the level of the "BGP noise". By BGP noise, we mean the iBGP updates that are routinely exchanged between BGP routers. A recent study at a large ISP [8] reports a BGP noise of more than one hundred iBGP updates per minute.

### 6.2.3 Tabu prefixes

In the previous section, we saw that relying on complex predictors was not worth for practical purposes and that a limited number of iBGP updates are enough to approach a good traffic balance. An issue with the on-line traffic engineering algorithm is that some BGP route might be tweaked back and forth during consecutive time intervals. This means that traffic flows would have to switch from one provider to another. In addition, if the on-line optimization algorithm decides to change the provider used to carry the traffic towards some prefix, then it would be desirable to stick to this choice during a sufficient amount of time to limit the effect of the on-line optimization on the instability of the topological distribution of the traffic.

For that purpose, we added a "tabu list" that maintains the set of BGP prefixes for which the on-line traffic engineering modified the route attributes during the last $x$ time intervals. The traffic for the BGP prefixes present in the "tabu list" cannot be moved to another provider during $x$ time intervals, starting from the time interval during which this BGP route change was applied. The tabu list contains the BGP routes changed by the algorithm during the last $x$ time intervals, hence BGP prefixes and the associated preferred provider.

To evaluate the practical interest of the "tabu list" method, we need to address the following questions:

1. How many new BGP route changes should be accepted per time interval?
2. For how long should a BGP route change be present in the tabu list ?

Figure 6.8 plots the average traffic imbalance for several values of the number of the iBGP updates per time interval (10, 20, 30 and 40) and the lifetime of the tabu list entries (0, 1, 3, 6, 9, and 12 time intervals). On the x-axis of Figure 6.8, the total number of BGP routes that have been tweaked by the on-line traffic engineering scheme for any time interval are shown. This total number of BGP routes that have been tweaked is not the same as the number of iBGP updates because with the tabu list method, we have no iBGP messages for the entries of the tabu list that stay the same between two consecutive time intervals. With the tabu list method, there are iBGP updates for the BGP routes whose attributes are reset to the default value (entries of the tabu list that expire) as well as iBGP updates for the BGP routes that are tweaked by the on-line traffic engineering scheme (entries that enter the tabu list).

Let $m$ represent the number of time intervals that an entry of the tabu list remains in the tabu list and $n$ represent the number of new tabu list entries accepted per time interval. Each time interval, there are thus $2n$ iBGP updates, $n$ for the new tabu list entries (newly tweaked routes) and $n$ for the old tabu list entries that expire. An entry of the tabu list expires when it has been present in the tabu list for more than $m$ time intervals. During each time interval, there are $n \times m$ tabu list entries that have not yet expired, as well as $n$ new ones selected to enter the tabu list, a total of $n \times (m + 1)$ BGP routes that have been tweaked.

Figure 6.8 presents the effect of the parameters of the tabu list method on the performance of the scheme. The important variable that guides the value of the traffic objective on Figure 6.8 is the total number of BGP routes that have been tweaked by the traffic engineering scheme, not the number of iBGP updates per time interval. The number of BGP routes that have been changed by the traffic engineering scheme corresponds to the number of entries in the tabu list. So on the traffic trace of $stub1$, there is ample choice for the number of iBGP updates

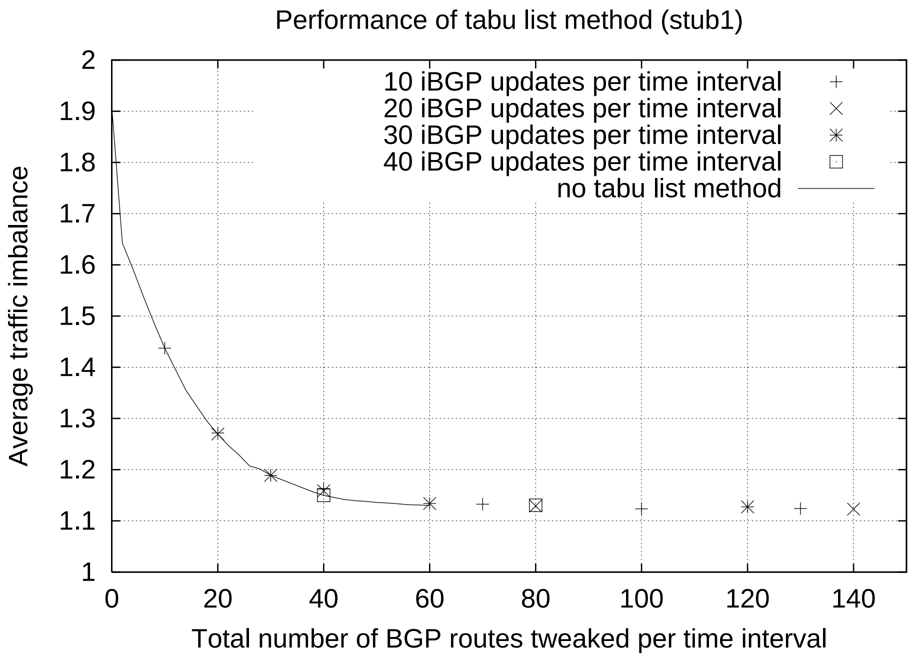



Figure 6.8: Performance of tabu list method with last value predictor.

allowed as well as the number of tabu list entries present during each time interval. The network manager has thus the choice to choose the acceptable burden on BGP as well as how close he wants to get from the optimal traffic balance achievable in practice by changing the size of the tabu list. The two questions above are thus linked and only the total number of modifications with respect to the default BGP routes seems to be relevant for what concerns the value of the traffic objective achieved. On Figure 6.8, we also reproduce the results obtained without relying on a tabu list ("last value" curve of Figure 6.7), for up to 60 BGP route changes. The tabu list results are seen to match the "no tabu list" ones, with the points for the tabu list results closely following the "no tabu list" curve. This confirms that the value of the traffic objective seems to depend largely on the number of tabu list entries present during each time interval.

The benefits of the tabu list method are obvious when comparisons are made with respect to the number of iBGP updates. Figure 6.9 gives for the *stub2* trace the imbalance ratio as a function of the number of iBGP updates per time interval with the tabu list method. The comparison made on Figure 6.9 is intentionally unfair towards the "no tabu list" method because the tabu list method limits the number of iBGP updates per time interval. The continuous curve on Figure 6.9 labeled "no tabu list" gives the improvement in traffic balance when the optimization method does not take into account the BGP routes that have been tweaked by the algorithm in the past. The continuous curve hence provides a higher bound on the number of iBGP updates per time interval to achieve a given traffic balance. The other three curves on Figure 6.9, labeled "tabu entry lifetime = x min", correspond to the result of the tabu list method with a tabu list entries lifetime of $x$ minutes ($x = 10$, 20, 30 and 60). The "no tabu list" method shows that *stub2* starts with a traffic imbalance

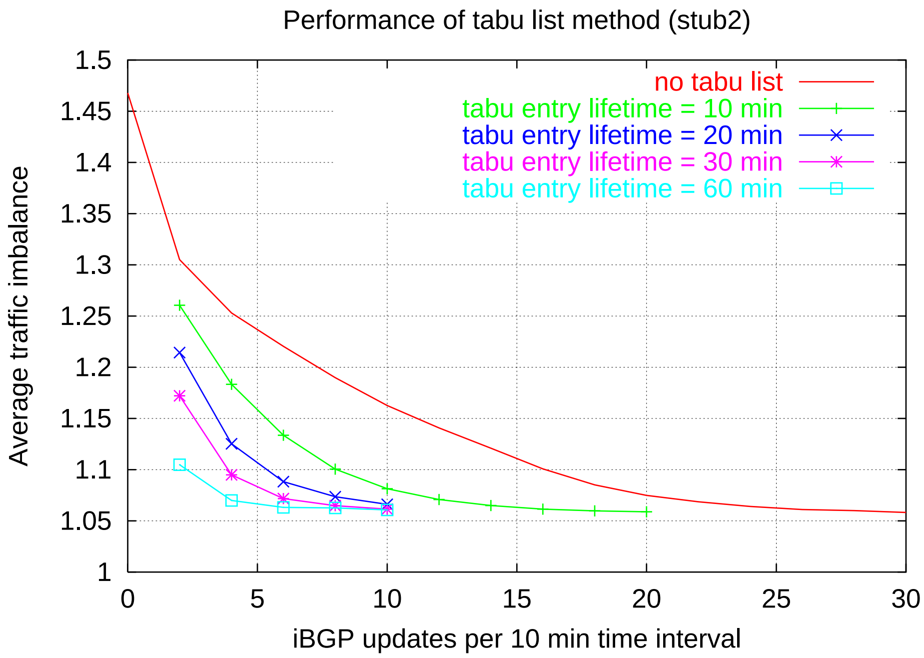



Figure 6.9: Benefit of tabu list method to limit the number of BGP advertisements.

(0 BGP advertisements) of less than 1.5 and the improvement in traffic imbalance decreases slowly, to stick to a traffic imbalance of about 1.06 for more than 30 iBGP updates per time interval. The curves corresponding to the “tabu list” method on the other hand start from a lower value of the traffic imbalance, thanks to their larger number of total BGP routes tweaked per time interval (entries of the tabu list). With only 10 iBGP updates per time interval, the “tabu list” method is able to attain a traffic imbalance a little above 1.06 for a tabu list entries lifetime of 30 and 60 minutes.

The tabu list method performs well compared to the method that allows all the BGP routes to change for each time interval. The tabu list method has the advantage of allowing to reduce the burden on the number of iBGP updates to be sent during each time interval, without impacting on the quality of the traffic balancing. Maintaining a list of tabu BGP routes is thus extremely interesting from a practical viewpoint.

### 6.2.4 MED tweaking

Up to now, the tweaking of the BGP routes was done by using the `local-pref` attribute, ensuring that the optimization will override all other BGP tweaking. This way of tweaking the BGP routes could make the BGP decision process choose a route with a longer AS path in order to optimize the traffic balance, potentially leading to a worse end-to-end quality of the routes. If the on-line optimization is required not to override the other types of traffic engineering (e.g. MED or IGP cost), then ideally one would prefer that the optimization tweaks a BGP attribute that is evaluated by the BGP decision process just before the last rule.

In the case of non-transit ASes, the optimization could rely on the MED attribute. According to the BGP Internet draft [138] the MED attribute cannot be compared for routes learned from different ASes, the route with the lowest MED being best. Allowing to always compare the MED attribute among BGP routes is however common [87]. Although there is no reason a priori to compare the MED attribute of routes learned from different neighboring ASes, stub ASes could rely on the MED attribute or another attribute having a meaning local to the AS to perform traffic engineering. Only limited modifications to BGP would be needed.

In this section, we allow the BGP instance running on the "optimization box" of the local AS to set the MED attribute of the routes received from the different neighboring ASes to prefer one route over another. The working of the optimization technique is similar to the one presented in section 6.2.3, except that the preferred route for the optimization algorithm gets a lower MED compared to the other routes towards a given prefix instead of having a higher value of the `local-pref` attribute. The main difference with the algorithm of section 6.2.3 is that only routes having the same `local-pref` value and AS path length will have their MED value compared. The optimization technique has thus a more limited choice in the prefixes that can be used to balance the traffic, compared to the `local-pref` way.

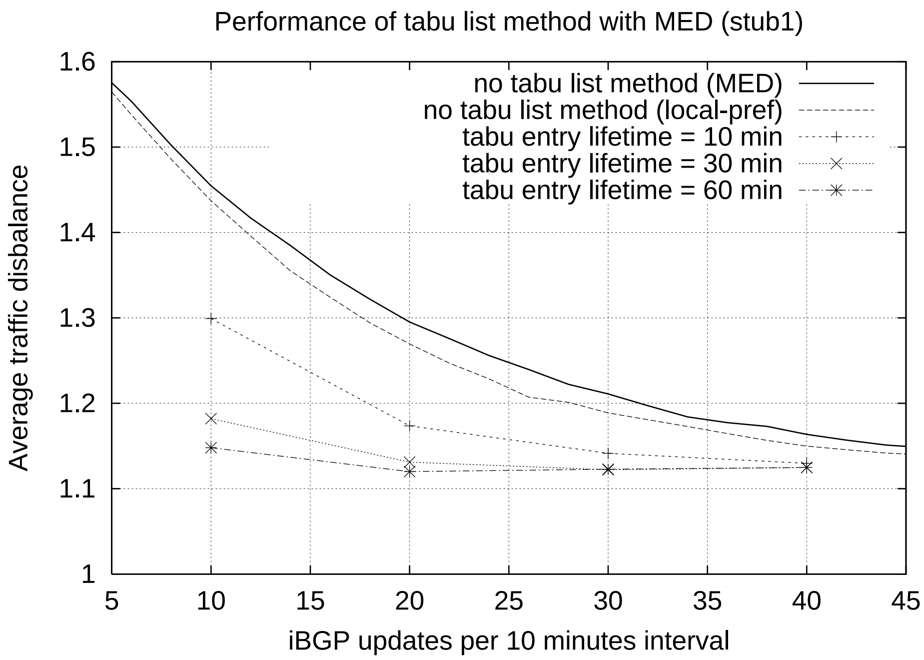



Figure 6.10: On-line optimization with MED.

Figure 6.10 presents the simulation results of the tabu list on-line optimization with the MED attribute for *stub*1. Figure 6.10 shows the average traffic imbalance as a function of the number of iBGP updates per 10 minutes interval. Figure 6.10 is the counterpart of Figure 6.9. Figure 6.10 shows that the performance of the tabu list on-line optimization with MED is quite comparable to the one with the `local-pref` attribute, only slightly worse. The two continuous curves on Figure 6.10 give the simulation results for the "no tabu list" method, with `local-pref` and MED. The dotted lines of Figure 6.10 on the other hand provide the simulation results for the tabu list method by tweaking MED. Increasing the size of the tabu list (by increasing the entries lifetime) provides a significant reduction of the number of iBGP updates to be announced.

### 6.2.5 Cost-weighted traffic balancing

The previous sections have provided simulations with a relatively simple traffic objective. In this section, we provide simulation results with the cost-weighted traffic balance objective introduced in section 6.0.1 for the traffic of *stub2*. This objective is related to the economical cost of the interdomain traffic, hence it is important in practice. The cost-weighted traffic balancing objective is defined as

$$Minimize \quad \sum_{i=1}^{n} c_i . \frac{tr_i}{\sum_{i=1}^{n} tr_i}$$

$$subject\ \ to \quad \sum_{i=1}^{n} \frac{tr_i}{\sum_{i=1}^{n} tr_i} = 1$$

where $c_i > 0 \ \ \forall i$, denotes the cost of one unit of traffic sent to provider $i$. We used the following costs: $c_1 = 1.5, c_2 = 1.25$ and $c_3 = 1$. These costs yield the following traffic distribution at the minimum: $tr_1 = \frac{10}{37} \times \sum_{i=1}^{n} tr_i$, $tr_2 = \frac{12}{37} \times \sum_{i=1}^{n} tr_i$, and $tr_3 = \frac{15}{37} \times \sum_{i=1}^{n} tr_i$.

We left the costs $c_i$ fixed in the following simulations but nothing in our scheme prevents the costs from varying with time or depending on other parameters. For instance, a larger cost can be attributed to the traffic sent to a provider during the busy hours or the cost could depend on the relative load of the access links. The sole constraint on the objective function is that the impact on the objective function of a change in the best BGP route for some destination prefix must be known for the search algorithm to be able to improve it. No constraint on the look of the objective function (convexity, piecewise linearity,...) is made.

Figure 6.11 presents the results of the on-line optimization using the `local-pref` attribute and for the traffic of *stub2*. The y-axis of Figure 6.11 represents the average normalized cost, i.e. the value of the traffic objective divided by the optimal cost. The x-axis represents the number of iBGP updates to be made per 10 minutes time interval, for up to 40 iBGP updates. The top curve on Figure 6.11 provides the result of the simulation without use of the tabu list method. The three bottom curves show the results using the tabu list method and with tabu list entries lifetimes of 30 minutes, 1 hour and 2 hours. As shown by the slower decrease rate of the top curve of Figure 6.11, the cost-weighted traffic balancing objective is more difficult. More iBGP updates are required to approach the optimal cost compared to the traffic balancing objective of the previous sections. The initial value of the cost-weighted traffic objective found by BGP is 1.33. Without the tabu list method, 20 iBGP updates per 10 minutes interval yield an average normalized cost of 1.2, 100 iBGP advertisements an average normalized cost of 1.1 (not shown on Figure 6.11), and 180 iBGP updates an average normalized cost of 1.06 (not shown on Figure 6.11).

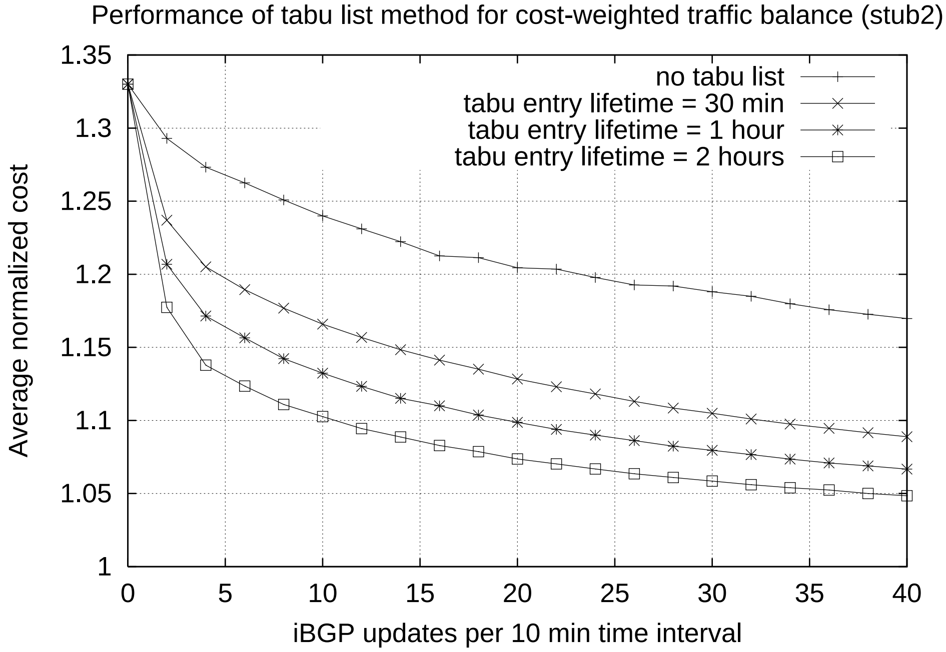



Figure 6.11: Optimization of cost-weighted traffic balance objective.

As expected, the tabu list method allows to drastically reduce the number of iBGP updates required to achieve a given gain in terms of the traffic objective. The larger the lifetime of the tabu list entries, the smaller the number of BGP advertisements required to reach a given average normalized cost.

## 6.3 Extensions of our approach to interdomain traffic engineering

In this chapter, we have proposed the utilization of an optimization box inside stub ASes to control the flow of their outgoing interdomain traffic. As presented in figure 6.5, this optimization box would collect the traffic statistics from the border routers and maintain multi-hop eBGP sessions with all the providers of the stub. In practice, the optimization box could be co-located with a Route Reflector [24] provided that it could receive all eBGP routes advertised by the providers of the stub. This could be achieved by using the BGP extension proposed in [181].

Besides controlling their outgoing traffic, stubs also need to control their incoming traffic. Several BGP tweaking techniques have been proposed to achieve this control [137, 68, 10] but they all suffer from the same drawback. To control its incoming traffic, a stub must influence the decision process of many remote ASes. Unfortunately, each remote AS, by controlling its outgoing traffic, may influence the flow of the incoming traffic in our stub AS. The approach that we propose cannot be used to control the flow of the incoming traffic. This control would require an explicit cooperation between distant ASes as proposed in [9].

Although the focus of this chapter is on stub ASes, transit ASes may also benefit from such on-line traffic engineering with BGP. According to [157], there were 2334 provider ASes in the Internet. Those providers have different sizes and different traffic engineering requirements. Most of those transit ASes are small national or regional ASes that are connected to larger providers. Those transit ASes will also need to optimize their provider links. Finally, the Tier-1 ISPs at the core of the Internet do not have any provider, but they often have multiple peering links with other Tier-1 ISPs that need to be engineered.

On-line traffic engineering might pose problems for transit ASes having a large number of downstream customer ASes since changing only a few BGP routes will require to announce the modified BGP routes to every downstream customer. If this number of customers is large, then even a few BGP advertisements every few minutes might constitute a burden. For transit ASes with a limited number of customers on the other hand, announcing a small number of BGP routes every few minutes will not hassle the BGP routers of the customers.

The main issue with transit ASes is that an iBGP change in the AS would cause a modification of the eBGP routes advertised by the transit AS to its BGP neighbors [68]. This would potentially increase the instability of BGP in the Internet. Furthermore, if the traffic engineering technique changes the best route towards some destination prefix, then downstream ASes with respect of the traffic direction could also change their best route to reach this particular destination prefix. Suppose that on Figure 6.12 transit AS 6 relies on a traffic engineering scheme to balance its outbound traffic. Assume also that the best BGP route chosen by transit AS 6 to reach destination prefix `X/Y` is the one through AS 2. Transit AS 6 thus advertises to AS 7 a BGP route to reach prefix `X/Y` with an AS path `6-2-1` of length 3. AS 7 receives another BGP route to reach prefix `X/Y` through transit provider 5, with an AS path `5-2-1` of length 3. If the traffic engineering scheme that transit AS 6 implements chooses to use the BGP route through AS 4 and AS 3 to reach prefix `X/Y`, then transit AS 6 will advertise to AS 7 a BGP route for prefix `X/Y` with an AS path `6-4-3-1` of length 4. If AS 7 does not tweak its BGP routes with the `local-pref` attribute, the BGP decision process of AS 7 will choose as its best BGP route to reach prefix `X/Y` the route learned from transit AS 5 that has a shorter AS path. Hence the effect of the tweaking of the BGP routes by transit AS 6 to balance its outgoing traffic will be to deflect part of the traffic it carried having as destination prefix `X/Y`.

A possible solution to avoid this problem would be to ensure that an iBGP change inside the AS does not cause an eBGP change. This could be achieved by using AS Sets [138] and new the BGP aggregation rules [138]. In BGP, aggregation is used to aggregate several prefixes together to reduce the size of the BGP routing tables. For example, assume that in figure 6.12 AS 2 advertises 10.0.0.0/8 to AS 6 and AS 4 advertises 11.0.0.0/8. In this case, AS6 could advertise prefix 10.0.0.0/7 to AS 7 with an AS path of 6-{2,4}. If AS6 engineers its outgoing traffic between its providers AS 2 and AS 4, then it could advertise prefix X/Y with an AS path of 6-{2,4,3}-1 to indicate that it can use different routes to reach X/Y. Since this advertisement covers both available paths, an iBGP change inside AS 6 would not

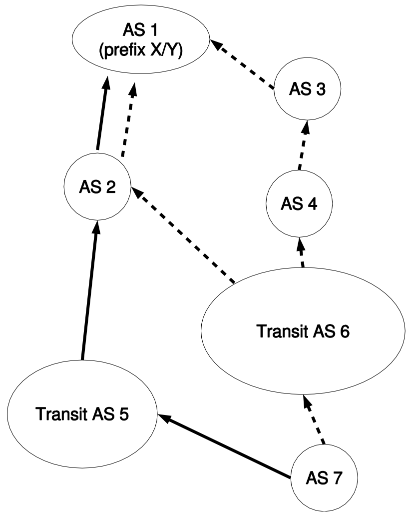



Figure 6.12: Interdomain traffic engineering with BGP for transit ASes

need to be propagated to AS 7. By using this aggregation, AS 7 receives a less precise AS Path than without the aggregation. This is a small price to pay compared to the benefit of reducing the number of eBGP messages. Other techniques can be used to obtain the exact AS-path to reach a given destination [114]. In practice, a small to medium transit AS would probably only use an on-line traffic engineering technique to control the traffic sent via its providers and not the traffic sent to its peers. The proposed aggregation should only be used for routes learned via providers. This can easily be achieved by marking those routes with a special BGP community value [146].

Another concern with traffic engineering techniques is that local objective functions might impact on other aspects of the end-to-end paths followed by IP packets like path length [148, 159, 153] or path performance [92]. For instance, an AS optimizing the traffic balance or some cost function over its inbound or outbound traffic without respect to AS path length might increase the length of the AS-level or IP-level path to reach some destinations. Hence care must be taken when optimizing particular objectives to prevent some traffic engineering function from impeding on other important aspects of the flow of the interdomain traffic.

## 6.4 Conclusion

In today's Internet, for both cost and performance reasons, Internet Service Providers often need to engineer their interdomain traffic. This interdomain traffic engineering is often done by manually tweaking the configurations of the BGP routers on an error-prone trial-and-error basis.

In this chapter, we have proposed a systematic approach to solve this important operational problem. Our approach allows the network operator to define objective functions on the interdomain traffic. Those objective functions are used by an optimization box placed inside the AS that controls the traffic. For this, it relies on the BGP routes and the traffic statistics received from the border routers of the AS. Based on the traffic statistics from the last period, the optimization box predicts the traffic for the next period. Then, it uses an efficient evolutionary algorithm to select the required iBGP changes to optimize the traffic during the next period. Those iBGP changes are then distributed inside the AS.

We have evaluated the performance of our evolutionary algorithm through simulations with real traffic traces from two stub ASes. Our simulations with two different objective functions show that the outgoing traffic can be efficiently engineered on a 10 minutes timescale by advertising not more than a few iBGP messages per minute. This is lower than the BGP noise in the Internet. Moreover, our prototype implementation written in Perl can be used in real-time since it requires only a few seconds of CPU on a P4 2.4 GHz to select the required iBGP changes for each 10 minutes period.

Finally, although our technique was designed for stub ASes, we have discussed how a different type of AS path aggregation could be used in small to medium transit ASes to allow them to engineer the traffic sent to their providers or peers without sending eBGP advertisements.

# Chapter 7

# Multi-objectives interdomain traffic engineering

In this chapter, we extend the problem of interdomain traffic engineering with BGP for stub ASes with as few BGP filters as possible. We study the optimization of two simultaneous objective functions, one defined on the short-term traffic dynamics and another on the daily traffic demands. We propose a multi-objective evolutionary algorithm aimed at sampling the objective space and illustrate its working with realistic traffic billing procedures.

In section 7.1, we describe the problem we deal with in this chapter. Section 7.2 presents our multi-objective evolutionary algorithm on which we rely to study the problem of multiple-objectives interdomain traffic engineering by tweaking BGP. Section 7.3 then presents the scenario of the simulations used to evaluate the problem described in section 7.1. Finally, section 7.4 presents the results of our simulations.

## 7.1 Problem statement

The problem this chapter tries to answer is the one of studying the trade-offs between the following three objectives :

1. the number of BGP filters,
2. the cost or balance of the traffic over the available providers on long timescales,
3. optimize the traffic balance over the available providers on short timescales.

The first objective concerns BGP routing. Given that there are many remote ASes with which an AS exchanges traffic on timescales of hours to days (see Chapter 3.5), one cannot require that engineering the interdomain traffic needs hundreds of BGP filters or even more. An objective of an interdomain traffic engineering technique should be to minimize as much as possible the burden on BGP. Tweaking the value of local BGP attributes should have no impact on the BGP advertisements sent to BGP peers for stub ASes. This does not preclude however the possibility of misconfigurations [109].

The second objective is related to long-term interdomain traffic engineering. With the rapid changes in traffic patterns and providers economics, multi-homed ASes will become more sensible to their usage of their upstream providers in the future. For this, BGP could be used to achieve long-term traffic engineering.

Finally, the third objective is more or less related to QoS. As far as the description of "route optimization" techniques allow to understand their internal working, the objective of such techniques is to find the best interdomain path to reach a given destination, based on both active and passive measurements [12]. While we have assessed the feasibility of optimizing the daily traffic demand of stub ASes in Chapters 4 and 5, we do not know how short-term traffic engineering like the one performed by "route optimization" techniques would interact with longer-term interdomain traffic engineering as proposed in Chapters 4 and 5.

An important concern for interdomain traffic engineering is to know whether long-term and short-term objectives are conflicting, and how much BGP filters would be required to achieve a "good" traffic engineering, namely minimizing the traffic objectives while not putting too much a burden on BGP. We would thus like to find out whether the three objectives are conflicting, neutral or harmonious [133]. This knowledge would tell us what we can realistically expect from traffic engineering when dealing with several objectives of a different nature like the three presented in this section. Note that we shall recurrently use the term "traffic objectives" for the second and third objectives related to the interdomain traffic.

## 7.2 BGP tweaking

Given that we do not wish to prefer some objective compared the other two objectives, we would prefer a search technique that will not bias the search towards subset of the three-dimensional search space. We know however that the sole parameter over which we have control in the setting of our problem is the BGP routing. The only thing we can do with BGP is to manipulate the BGP routes[1]. Due to the way the BGP decision process chooses the best route towards a destination, we have ample choice concerning how to tweak the BGP routes. Nonetheless, we decided in this chapter to tweak the value of the `local-pref` attribute. For this, we rely on what we call "BGP filters". A BGP filter is a pair $<$*prefix,provider*$>$ indicating that the traffic having the prefix *prefix* as destination will be forwarded through provider *provider* among the providers which advertised a BGP route towards this prefix. What we call a filter hence concerns a single BGP prefix. The actual effect of a filter on the BGP routes is to force the value of the `local-pref` attribute of the BGP route towards prefix *prefix* which was learned via provider *provider* to be set with a value of 110, hence a higher preference. All other routes for prefix *prefix*

[1]This is actually not true, we could also change the way the BGP decision process works, but this is outside the scope of this thesis.

(learned from other providers) have their `local-pref` attribute (re)set to the same value which will be strictly smaller than 110.

Our previous work in trying to find good BGP filters ([169, 176] and Chapter 5 showed us that the tie-breaking rules of the BGP decision process were the crucial part of tweaking BGP for interdomain traffic engineering. Most routes received by a non-transit AS from its upstream providers have a similar AS-path length. The rules of the BGP decision process (see section 0.4) that are situated after rule 2 will decide about the best BGP route. If a non-transit AS does not propagate the IGP metric into the BGP routes, the seventh rule of the decision process that will choose the best BGP route, namely the smallest `router-id` of the `next-hop` BGP router used to attain the destination, or the lifetime of the particular route depending on the particular router vendor's BGP implementation. This means that the "tie-breaking" rules of the BGP decision process will often bias the decision concerning the provider used to attain a destination towards the same provider, the one having the smallest `router-id` or whose route is the oldest.

The previous discussion concerning the BGP decision process is not just a networking technical detail. This aspect is of utmost importance because it implies that the complete search space permitted by BGP is vast since several rules of the BGP decision process could be modified by a BGP filter. However, our choice in this chapter is to try to understand the relationship between short-term and long-term interdomain traffic engineering and the BGP filters. Here our focus is on tweaking BGP in the most controllable way. In that respect, the `local-pref` attribute provides us with the certainty that setting a higher value of this attribute for the route of a given provider compared to its value for other provider's routes will force traffic towards this destination to get through this provider.

### 7.2.1 The search space

Having explained the main problem-related issues of our problem, this section is dedicated to the problem of finding the Pareto-optimal points for our three objectives. Given that the search is limited by the granularity of a BGP filter, we must now decide how we shall search the space of our three objectives. Several methods can be envisioned.

The first one consists in enumerating all possibilities. This solution is unmanageable for the simple reason that the size of the search space is too large. Assuming there are about one thousand ASes with whom the local AS exchanges a significant amount of traffic each day among the 16,000 ASes that compose the Internet in late 2003, and assuming that there are only two providers having each one route towards each destination of the one thousand networks, the number of possible BGP configurations for as few as ten BGP filters amounts to $\frac{1000!}{10!990!}$ possible BGP filters. This number has twenty-three decimal digits. We thus need to rely on another method to sample the Pareto-optimal front. Note that the number of possibilities is due to our restriction that several filters cannot be tried for a given prefix in the course of the search, and thus the order in which the BGP filters are applied is unimportant since they are independent of one another.

A second method would be to constrain the values of the two traffic objectives not to increase while applying the filters, for applying a filter should allow to improve in at least one of the two traffic objectives. This approach seems at first appropriate in that it will prevent the search to take place in regions whose BGP filters seem at first uninteresting. Unfortunately, we cannot guarantee that it will not be necessary to worsen the value of some objective through some BGP filter to further improve in the search and prevent the search to get stuck in a local optimum, in particular if the objectives are conflicting.

Thirdly, we could also work on the traffic objectives by trying local improvements of one of the two objectives and finding out how this relates to the BGP filters. The issue there concerns the discreteness of both traffic objectives. We cannot search for all solutions that are within some $\epsilon$ since this would require that we already know the effect of a BGP filter on the traffic objectives, which will not be true except for simplistic traffic objectives. One must however realize that although we can predict the effect of changing the value of the `local-pref` attribute of some route on the way traffic will be distributed over the different providers over the long-term, the extent of the change in the traffic objectives will depend on the particular objective functions. These objective functions not being forcibly linear and depending on the traffic dynamics on the short-term, basing the search on the traffic objectives could be even more difficult than the initial problem we aim to solve.

The previous discussion focused on the problem-specific constraints of our search. By now, it should be more or less apparent that the search of the Pareto-optimal front cannot be achieved through a blind search nor a local search in the traffic objectives. For such kind of Pareto-optimal front sampling, evolutionary algorithms are a well-known technique capable to find many points on the Pareto-optimal front in a single run [75, 49, 56]. Additionally, relying on the "evolutionary" paradigm allows to leverage the mechanisms of population-based search and selection among individuals. The advantages of evolutionary search can be measured by the number of recent studies that rely on such techniques in the literature. A list of references on evolutionary multi-objective optimization maintained by Carlos A. Coello Coello can be found at `http://www.lania.mx/~ccoello/EMOO/EMOObib.html`. The proceedings of the International conference on evolutionary multi-criterion optimization (EMO) also provides a large number of applications of evolutionary algorithm to real problems.

Now let us shortly discuss the reasons for our choice of the evolutionary paradigm to tackle our problem. The first reason is BGP. Our aim is to be as close as possible to the way BGP works in practice. Thus, we do not want to simplify the way the BGP decision process chooses the best route towards a particular destination because it is most critical for practical interdomain traffic engineering. Our algorithm takes as input real BGP tables and real traffic statistics. The second reason is that the traffic objectives need not be linear, convex, piecewise convex,... Interdomain traffic engineering objectives can be complex, non-linear, based on statistics,... Hence we consider that having to rely on strict assumptions concerning the traffic objectives would be too limiting for our study.

### 7.2.2 Data structure

The first issue one must tackle when developing an evolutionary algorithm is to think about the data structure used in the search. This aspect is important for two reasons. First, the data structure that represents an individual should allow easy manipulations during the search process. Second, the choice of the data structure must be concerned with the large amount of time an evolutionary algorithm often spends at evaluating individuals. So the structure must be close to the problem at hand, but also efficient for evaluating the fitness of the individual.

The aim of the individual's representation is to allow fast fitness evaluation, while at the same time containing information with respect to the three objective functions so that we do not need to recompute the individual's objective values each time we wish to manipulate it. The presence of the value of the two traffic objective functions of this individual seems valuable to efficiently compute new objective values for an additional BGP filters. The main question however concerns how to encode the BGP filters defined in section 7.2.

We can choose to directly encode which provider is preferred for each prefix. This would mean that the size of an individual would be in the order of the number of prefixes with whom our local domain exchanges traffic. We know however that there can be up to one thousand prefixes. A less costly choice in terms of the size of the encoding consists of putting in the individual only the prefixes for which the preference is changed compared to the default BGP solution, namely the BGP filters. The representation of an individual we use is shown on Figure 7.1.


Figure 7.1: Representation of an individual.

For what concerns the efficiency of the search, it is crucial that one does not have to recompute the complete traffic distribution among the providers when there are many prefixes with whom traffic is exchanged, especially since the number of BGP filters is due to be smaller than the number of such prefixes. This is only possible when the BGP filters have an effect on independent traffic aggregates, the prefixes in our case. If we had to try to further minimize the number of BGP filters, then we would have to work on traffic aggregates that leverage the way traffic is distributed over the AS-level topology. Such BGP filters however would not be independent due to several filters that could concern a common prefix in the topology. In that case the value of the `local-pref` attribute of the various routes for the concerned prefixes will not suffice to decide the best route but only the tie-breaking of the BGP decision process will. Putting complexity in the objectives' evaluation is not a good idea since this operation is performed frequently.

### 7.2.3 Search procedure

Depending on the relationships between the two traffic objectives which might be conflicting, harmonious or neutral, the search on the Pareto-front should have to be different [133]. Recall that we do not know beforehand the relationship between the traffic objectives, and our aim is to find based on our search of which type this relationship actually is. This means that our search method must be as lightly biased as possible towards any of the two traffic objectives to sample in the best possible manner the search space.

Our search works as follows. At the first generation, we start with a population of individuals initialized at the default solution found by BGP. Hence at generation zero all individuals have the same values of the two traffic objectives and contain zero BGP filters. At each generation, we use a random local search aimed at improving the current population by applying an additional BGP filter.

Each individual of the population is non-dominated with respect to the other members of the population for what concerns the traffic objectives. Recall that a solution is said non-dominated with respect to a set of solutions if no member of the set has better objective values for all objectives simultaneously. In addition, the current population is always made of individuals having the same number of BGP filters. At each generation, we parse the whole population and for each individual we try to apply an additional BGP filter. Whenever a BGP filter provides improvement with respect to at least one of the traffic objectives, we accept this improved individual and put it in the set of accepted individuals. We iterate this procedure until we find a target number of improved individuals or stop when we have performed a target number of tries (the variable `iter`). Figure 7.2 provides a pseudo-code description of the search procedure. Note that the pseudo-code given at Figure 7.2 concerns one generation, the purpose of variable `iter` is not to count the generations, but to ensure that the search will not loop indefinitely during the current generation.

### 7.2.4 Pareto-optimal front*

The previous section described the procedure to search for BGP filters that improve the individuals of the previous population with respect to any of the two traffic objectives. These improved individuals however are not non-dominated. Some of them can be dominated since we did not check for non-domination when accepting an improved individual. Improvement was sufficient to accept an improved individual. The next step is to check for non-domination [154] on this population of improved individuals to obtain a non-dominated front. For that purpose, we rely on the fast non-domination check procedure introduced in [57]. This procedure has time complexity $0(MN^2)$ where $M$ is the number of objectives and $N$ the size of the population. We do not describe this procedure in details but refer to [57] for the

```
accepted = 0
iter = 0
while ((accepted < MAXPOP) AND (iter == MAXITER)){
  foreach individual k {
      // trying a random BGP filter
      filter.prefix = rand_int_uniform(1,MAXPOP)
      filter.provider = rand_int_uniform(1,NUM_PROVIDERS)
      // if effect of filter is improvement accept it
      if (improved(k,filter)){
          accept(k,filter)
          // update counter for accepted improved individuals
          accepted++
      } // end if
  } // end foreach individual
  // update iteration counter
  iter++
} // end while
```

Figure 7.2: Pseudo-code of search procedure for a single generation.

original idea and to [56] for a detailed explanation. Let us only mention the main points here. Let $P$ denote the set of non-dominated individuals found so far at the current generation. $P$ is initialized with anyone of the individuals among the accepted ones. Then try to add individuals from the set of accepted ones one at a time in the following way:

- temporarily add individual $k$ to $P$
- compare $k$ with all other individuals $p$ of $P$:
  - if $k$ dominates any individual $p$, delete $p$ from $P$
  - else if $k$ is dominated by other members of $P$ remove $k$ from $P$

This procedure ensures that only non-dominated individuals are left in $P$. The number of domination checks is in the order of $0(N^2)$ while for each domination check $M$ comparisons are necessary (one for each objective). The maximum time complexity is thus $0(MN^2)$.

Having found the non-dominated front for a given number of BGP filters, we are left with determining the individuals of the next population. Actually, the number of non-dominated individuals from the set of improved ones is due to be smaller than the size of the population we use during the search process (MAXPOP). To constitute the population for the next generation, we have to decide how many individuals in the next population each non-dominated solution will produce. Because non-dominated individuals are not comparable between one another, we must choose a criterion that will produce MAXPOP individuals from the set of non-dominated ones. On the one hand, we would like to include at least every non-dominated individual in the population. On the other hand, depending on the way the accepted solutions are spread over the non-dominated front, we must sample differently different regions of the non-dominated front for a given number of BGP filters. This notion of sampling the non-dominated front is close to an idea of distance between neighboring individuals in the objective space. Maintaining diversity on the non-dominated front requires that individuals whose neighbors are farther apart be preferred over non-dominated individuals whose neighbors are close. The rationale behind this is that less crowded regions should require more individuals to be correctly explored than regions having more non-dominated individuals. The computation of the crowding distance for each individual is done according to [56] pp. 248. First the non-dominated individuals are sorted according to each objective. Then the individuals having the smallest and largest value for any objective are given a crowding distance $d^m$ of $\infty$ to ensure that they will be selected in the population. For each objective $m$, the crowding distance of any individual $i$, $1 \leq i \leq (|P| - 2)$, is given by

$$d_i^m = \left| \frac{f_{i+1}^m - f_{i-1}^m}{f_{max}^m - f_{min}^m} \right| \tag{7.1}$$

where $f_i^m$ denotes the value of individual $i$ for objective $m$, $f_{max}^m$ (respectively $f_{min}^m$) denotes the maximum (respectively minimum) of the objective value $m$ among individuals of the set $P$ of non-dominated individuals. The global crowding distance for all objectives is the sum of the crowding distance for all objectives. For our two objectives, this crowding distance represents half the perimeter of the box in which individual $i$ is enclosed by its direct neighbors in the objective space.

To achieve our goal of sampling the regions of the non-dominated front according to their crowding as well as generate a population of MAXPOP individuals from the non-dominated front, we perform a tournament selection based on the crowding distance of each individual. Figure 7.3 provides a pseudo-code describing the selection process. Until we have a population size of MAXPOP, we iterate the stochastic tour-

```
1 pop = 0
2 while (pop < MAXPOP){
3   // randomly choosing two non-dominated individuals in P
4   first = rand_int_uniform(1,|P|)
5   second = rand_int_uniform(1,|P|)
6   // largest crowding distance individual wins
7   if (crowd_dist(first) >= crowd_dist(second)){
8       population[pop] = first
9   }
10  else {
11      population[pop] = second
12  }
13  // incrementing population counter
14  pop++
15 } // end while
```

Figure 7.3: Pseudo-code for crowding distance based selection.

nament (line two of Figure 7.3). A tournament selects uniformly two individuals in the set $P$ (lines four and five of Figure 7.3) and the winner of the tournament is the individual with the largest crowding distance (lines seven to twelve of Figure 7.3).

### 7.2.5 Practical issues*

While the previous sections provided the main features of our search procedure, several practical issues need to be mentioned. First, in order to sample correctly the search space, a sufficiently large population must be used. By “sufficiently large”, we mean a few times larger than the number of possible BGP prefixes. The reason is that when the non-dominated front grows larger, the non-dominated individuals having the largest crowding distance must get the opportunity to have a large number of BGP filters tried during the next iteration. So the population size must scale up with the size of the non-dominated front in order to be able to progress in the search. On the other hand, the larger the population size, the more CPU time it takes at each generation. With a population limited to ten thousand individuals, it took up to one day on a P4 2.4 GHz with one gigabyte of RAM to perform one simulation (one hundred generations). Reducing the population size is dangerous since the sampling of the search space will suffer. The other issue concerns the number of non-dominated individuals found. The larger the non-dominated front, the more time it takes to sort its elements. Although the crowding distance procedure is quite fast $O(MN^2)$ with $M$ being the number of objectives and $N$ being the number of non-dominated individuals, with a non-dominated front made of less than one hundred individuals and two objectives, it is still scalable but when it goes higher than one thousand individuals the crowding procedure becomes a bottleneck. So both the size of the population and the non-dominated front need to be limited for practical purposes.

We used a population size of ten thousand and limited the non-dominated front size to one thousand individuals. In addition, we do not accept individuals as soon as they improve the individual from which they come, but we perform the non-domination check just after line nine of Figure 7.2 so that the number of accepted individuals does not grow too large. This prevents individuals that would be discarded during a later non-domination check phase to be accepted in the population of improved individuals. The issue with this way of accepting individuals is that we slow down the search in order to prevent the size of the population of improved individuals to grow too large. If one allows dominated individuals in the population of improved individuals, this will fasten the acceptance of improved individuals, hence a gain in CPU time. On the other hand, not doing this non-domination check at acceptance time permits uninteresting improved individuals to be accepted while they will be discarded later anyway. So in practice some tuning is required to decide if the domination check should be performed during the acceptance of improved individuals or later.

## 7.3 Simulation scenario

The scenario of our simulations is the same as the one of section 6.2.1. The traffic trace used in our simulations is a one day trace of all the outbound traffic from the Université catholique de Louvain starting from April 2 2003 17:20 CET. During this one day period, 215 GBytes of traffic were observed with an average bit rate of 19.9 Mbps. This one day long trace was cut into time intervals of ten minutes. For each ten minutes interval, we have one file giving the amount of bytes for every BGP prefix towards which the university sent some traffic.

In addition, we gathered a complete routing information base from Oregon Route views [119] dating from April 2 2003 to simulate the BGP routing tables of different providers to which our stub would be connected. While it is known that BGP routes dynamics can be important for some prefixes, it has however been shown in [139] and Chapter 3 that BGP routes can be considered as stable over relatively long periods, so we use one BGP routing table per provider for the whole day.

## 7.4 Simulations

Having provided the necessary background to understand the working of our search algorithm, we now proceed to analyze the behavior of the search for real datasets. Our objective is not to validate the algorithm or to prove that our search technique works. The aim of this section is to provide a case study concerning the search of "good" BGP filters to understand the relationships between the three considered objectives: the number of BGP filters, the long-term traffic objective and the short-term traffic objective. It should be seen only as an illustrative case study on our particular traffic demand and several traffic objectives. Choices of different traffic demands, objective functions and providers might lead to different results. No conclusion should be drawn based on these simulations concerning the quantitative improvements achieved in terms of any of the objectives.

### 7.4.1 The Broad Picture

Let us first understand how our algorithm performs when we simply sample the objective space without trying to optimize one particular objective, namely when running the basic algorithm described in section 7.2.3. In this section, we use a scenario where our stub AS is connected to four different large tier-1 providers. We relied on the BGP tables of AT&T (AS7018), UUNet (AS701), Verio (AS2914) and Sprint (AS1239). The long-term cost function corresponds to volume-based billing with costs per traffic unit of 0.5, 1, 1.5 and 2 for UUNet, AT&T, Verio and Sprint respectively. The amount of traffic that a "traffic unit" represents is not important, only the relative cost is. For the short-term objective, we computed for each ten minutes period the maximum amount of traffic sent through any of the four providers, and summed these maxima over the whole day (traffic balancing objective). While this scenario is probably not representative of a real stub AS since being connected to four different large tier-1's is unlikely in practice [157], it can be considered as a difficult setting for our algorithm due to similar AS path lenths for the BGP routes between tier-1 providers for many destinations. The cost functions used in these first simulations need not be the most realistic one can think of since our focus here is to understand the behavior of the search algorithm on a first scenario. More realistic simulations will be carried in the next sections. Note that throughout this chapter we do not consider a prefix if it has less than one megabyte of traffic during a ten minutes interval in order to limit the number of prefixes that have to be taken into account. This would be done in practice since removing such small traffic prefixes is marginal with respect to the total traffic while reduces the computational burden of the search. By doing this, we still consider about ninety percent of the total daily traffic while reduced the number of different prefixes seen over the day from more than eighty thousand to less than three thousand.

We plot on Figure 7.4 the non-dominated front found for one hundred generations of the algorithm with the previously described scenario. Recall that the successive generations correspond to successive values of the number of BGP filters. The point corresponding to the default BGP routing has zero BGP filters and a value of the two traffic objectives equal to one. We normalized the objective values of all points with respect to the value found by the default BGP routing (with zero BGP filter) in order for the graphs to be independent of the actual value of the objective functions that depend on the amount of traffic considered, the length of the time interval, and the particular costs defined for the objective functions. Because our focus here is on the look of the non-dominated front and not the actual numerical values of the objective functions, we decided not to show any actual value of the traffic objectives but only normalized ones.

Globally, we see two regions on Figure 7.4. The first region concerns point for the first few BGP filters. These points start at the top right of Figure 7.4 (default BGP solution) and converge to the non-dominated front which constitutes the second region of the graph. The second region of the non-dominated front indicates that the two traffic objectives are conflicting for a number of BGP filters larger than twenty. The conflicting nature of the objectives can be seen by a relatively linear (slightly convex) trade-off between the two traffic objectives, for a given number of BGP filters. Finding a solution providing a smaller cost on the long-term for a given number of BGP filters requires to worsen the short-term objective value. In the same way, finding a solution providing a smaller value of the short-term objective function for a given number of BGP filters requires that one worsens the value of the long-term objective function. But this is valid only for a sufficiently large number of BGP filters.

The 3 dimensional plot of Figure 7.4 does not show precisely the value of each objective for a given number of BGP filters, so Figure 7.5 provides the projection of the points of the non-dominated front on each traffic objective. On Figure 7.5, we can see that for the first twenty BGP filters, the non-dominated front for a given number of BGP filters is limited to one or two points. This means that all other solutions found for this number of BGP filters were dominated by the points shown on Figure 7.4. In addition, Figure 7.5 showing the evolution of the values of each

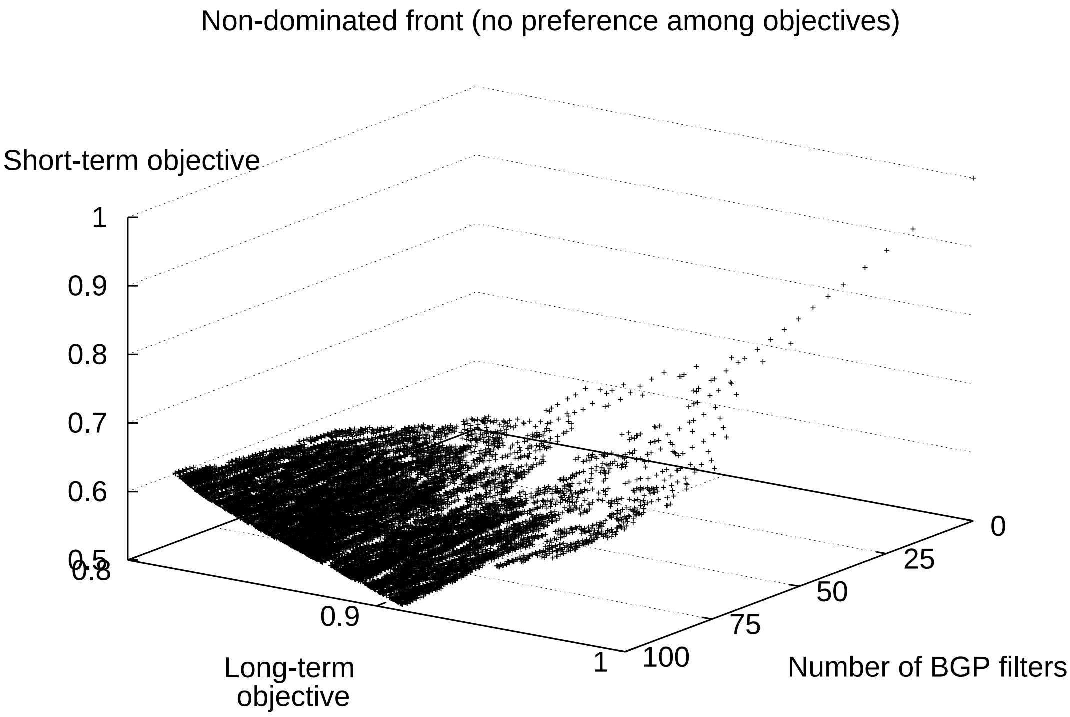



Figure 7.4: Solutions of the search without preference for any of the two traffic objectives.

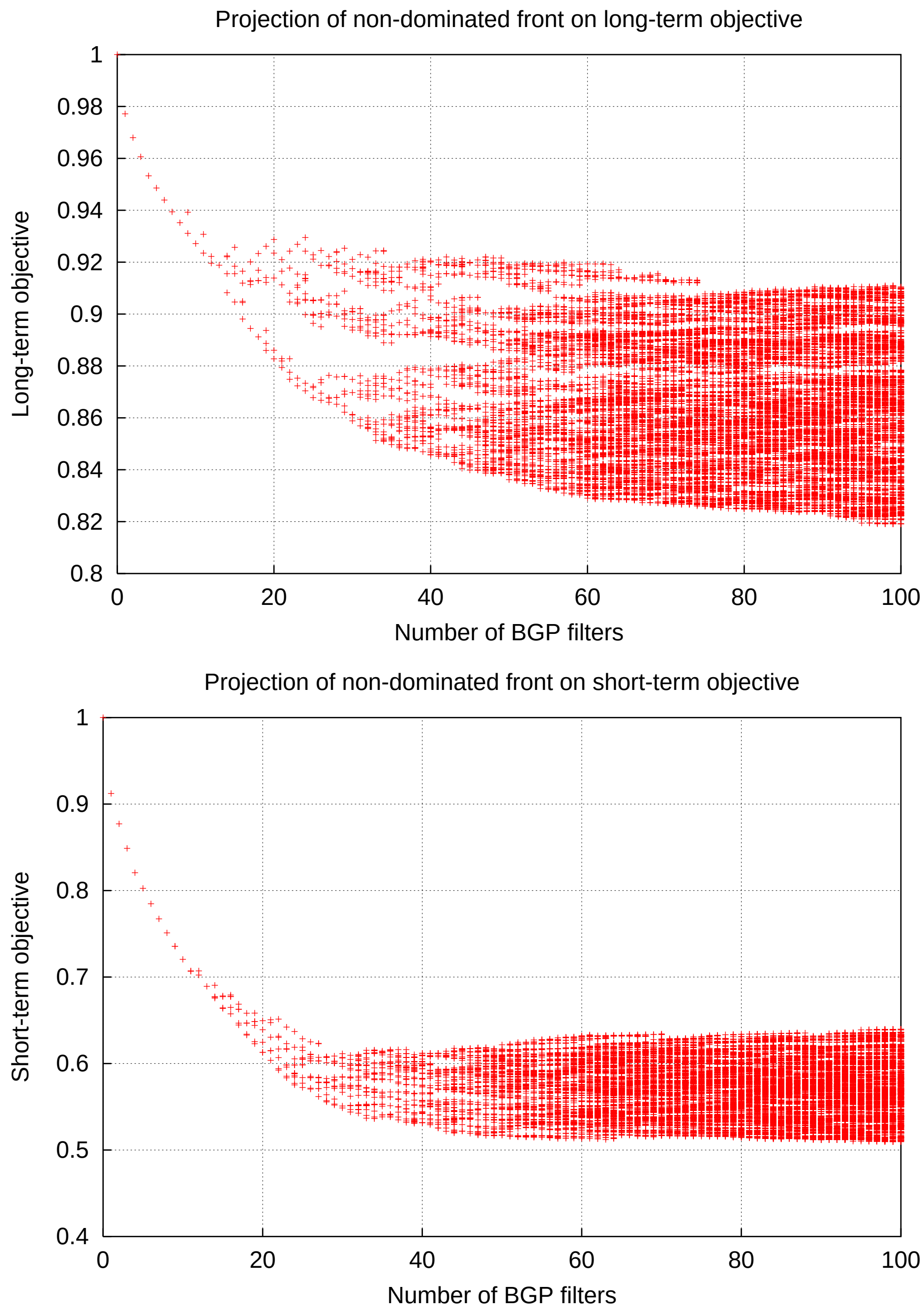



Figure 7.5: Projection of non-dominated front on each traffic objective (no preference among traffic objectives).

traffic objective as a function of the number of BGP filters for the non-dominated front tells us that the first few BGP filters allow a significant gain in terms of both traffic objectives at the same time. There are two explanations for this behavior.

First, BGP is a reachability protocol which was not designed to perform traffic engineering. The main feature of BGP is policy routing which is aimed at enforcing local policies in terms of routing. This is to be contrasted with our traffic engineering objectives, which do not care about routing policies but the distribution of the traffic on the available providers of the local AS. Being able to improve quite importantly the two traffic objectives means that this part of the search mainly counteracts the "inability" of the default BGP choice to optimize objective functions defined on the traffic. We insist that this "inability" of BGP to optimize traffic objectives is rather normal, in that BGP was not designed to do that. Relying on BGP filters to perform interdomain traffic engineering really is about "tweaking" BGP. Getting deeper into the working of the BGP decision process, we observed in Chapter 5 that the tie-breaking rules of the BGP decision process were extremely important to explain this inability of the BGP decision process to optimize a cost function defined on the traffic. The reason is that many prefixes have the same `local-pref` and AS path for the different providers of many stubs. [29] has compared the BGP routes of twenty tier-1 providers based on the route-views BGP tables [119], and found that about sixty percent of the routes had the same AS path length when comparing any two tier-1's. Thus it is likely that the tie-breaking rules of the BGP decision process are an important factor when trying to optimize a traffic objective by tweaking BGP, although it will depend on the different types of providers each stub is connected to.

Second, the size of the population used in our search also impacts the quality of the solutions found during the first few generations of the search. We used a population of ten thousand individuals for there were 2832 prefixes having more than one megabyte of traffic during at least one ten minutes time interval for the considered day. From this population of ten thousand individuals, we permitted that the number of solutions improving the individuals of the current population be arbitrary large. Each time a new individual improved the existing solution in the population from which it was improved, we checked for non-domination in the set of already improved and non-dominated individuals. If the newly found individual was non-dominated, we included it among the non-dominated front to prevent the same individual to be included more than once. Note that we cannot be certain that the non-dominated points we find for these small numbers of BGP filters are globally optimal with respect to the three objectives. The size of the search space, even for a few BGP filters, increases too fast for that. For instance, with four providers and 2832 prefixes that can be reached through any of the four providers, we have a worst case of $\frac{(2832\times 3)!}{10!((2832\times 3)-10)!}$ different possible solutions for ten BGP filters that could be applied, a number with thirty-two digits. Note that the number of prefixes is multiplied by three because a filter moves a prefix from the default provider found by BGP to one among the other three providers. Even if the actual number of BGP filters might not be large, the size of the search space grows quite fast. For the few first BGP filters, it is possible to enumerate all solutions, but this cannot be done for up to twenty BGP filters. So alone no practically feasible population size will ensure that we found the globally optimal set of BGP filters for up to twenty BGP filters.

### 7.4.2 Biasing the search towards one traffic objective

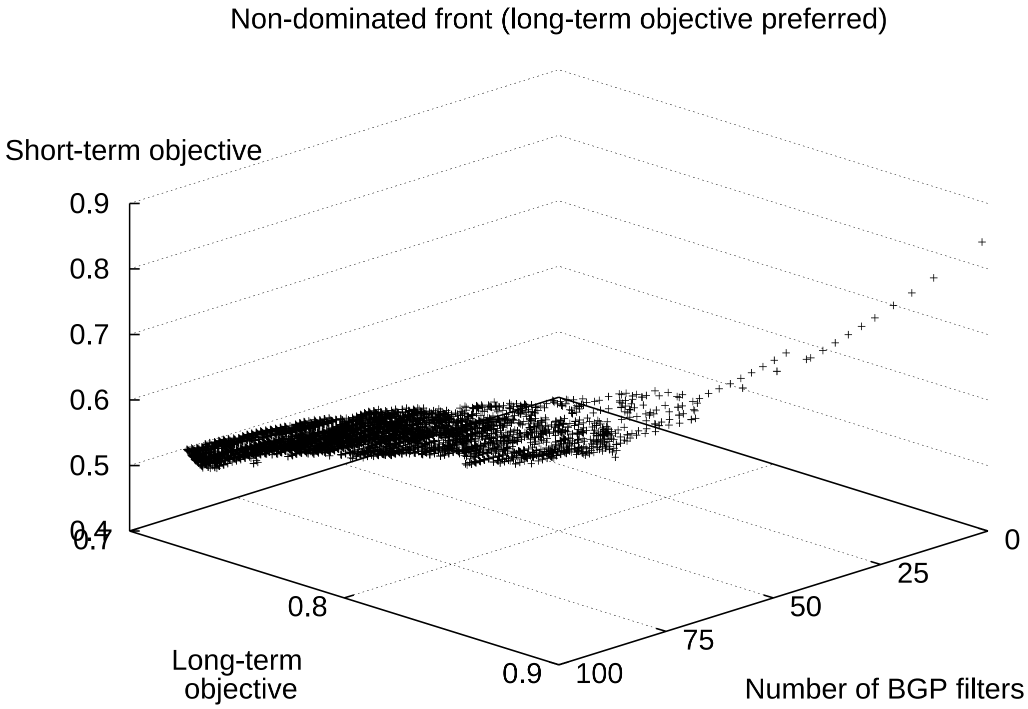
In the previous section, the algorithm was used to sample the non-dominated front for successive numbers of BGP filters. This provided a broad view of the relationship of the two traffic objectives we considered. If one is interested in obtaining better solutions in terms of one traffic objective, then a small change has to be made to the algorithm. When trying to sample the Pareto-optimal front, we aim to select solutions for the next generation that sample as broadly as possible the non-dominated front found at the current generation. When trying to find good solutions for a given traffic objective, we want to preferably select individuals based not on a crowding distance but on the value of the traffic objective we are interested in. So the only change to be made to the algorithm described in section 7.2.1 is within the selection procedure, where the individual that wins the stochastic tournament is the one having the best (depending on the type of objective) objective value for the traffic objective on which we focus.

Figure 7.6 provides the non-dominated front found with the selection process favoring the long-term traffic objective (top) and the short-term traffic objective (bottom). It is apparent on Figure 7.6 that when preferring any particular objective, the sampling of the non-dominated front is quite bad compared to the one found when a crowding distance is used (Figure 7.4). One would expect this behavior since selective pressure at each iteration favors the preferred objective, hence obtaining a representative non-dominated front for the two objectives will not happen whenever the two objectives are conflicting.

Figures 7.7 and 7.8 provide the corresponding projections of the non-dominated fronts over each traffic objective. The solutions found in terms of the particular traffic objective towards which the search was biased are not particularly better than those found with the crowding distance based selection (Figure 7.5). Only for large numbers of BGP filters do these biased search techniques find better solutions in terms of the preferred objective. Even when we prefer one particular traffic objective, the aim of the search is still to understand the relationship between the two objectives so our search does not guarantee that we find the best solutions in terms of the preferred objective. If one wants to obtain the best filters in terms of one particular objective, then a single-objective optimization technique should be used.

When one compares the best solution in terms of the each traffic objective found with and without bias, the biased search finds slightly better solutions, but at the price of driving the search towards BGP filters having a bad value of the other traffic objective (compare top of Figure 7.7 with top of Figure 7.5 and bottom part of Figure 7.8 with bottom of Figure 7.5).

The non-dominated front found when biasing the search towards one particular traffic objective indicates that the conflicting nature between the two traffic objectives



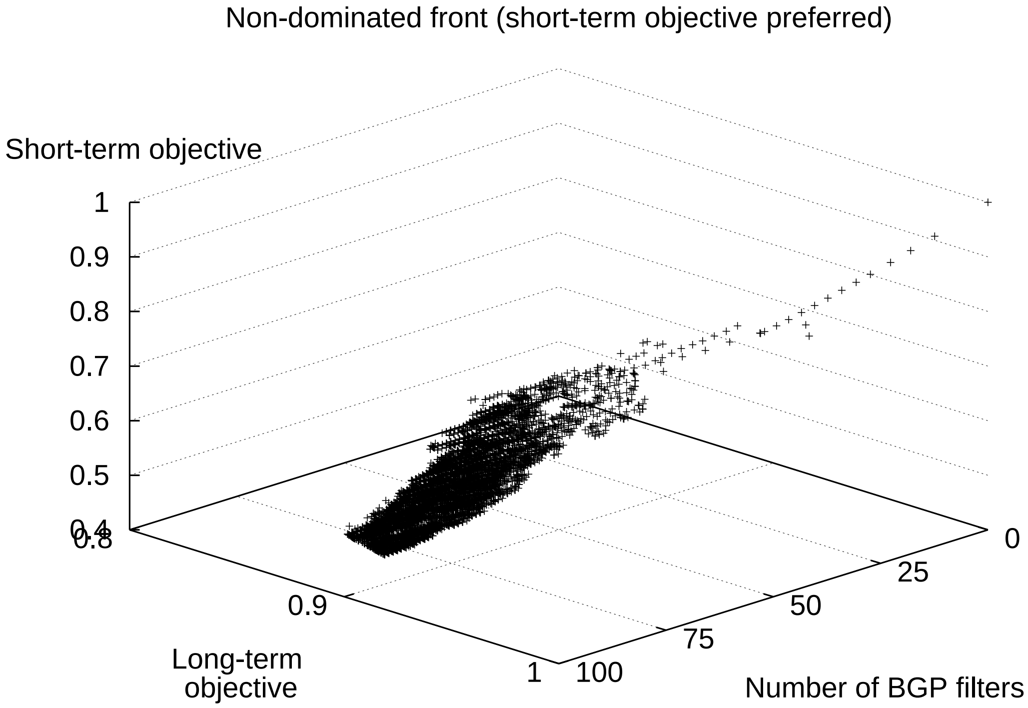




Figure 7.6: Solutions of the search for long-term preferred objective (top) short-term preferred objective (bottom).

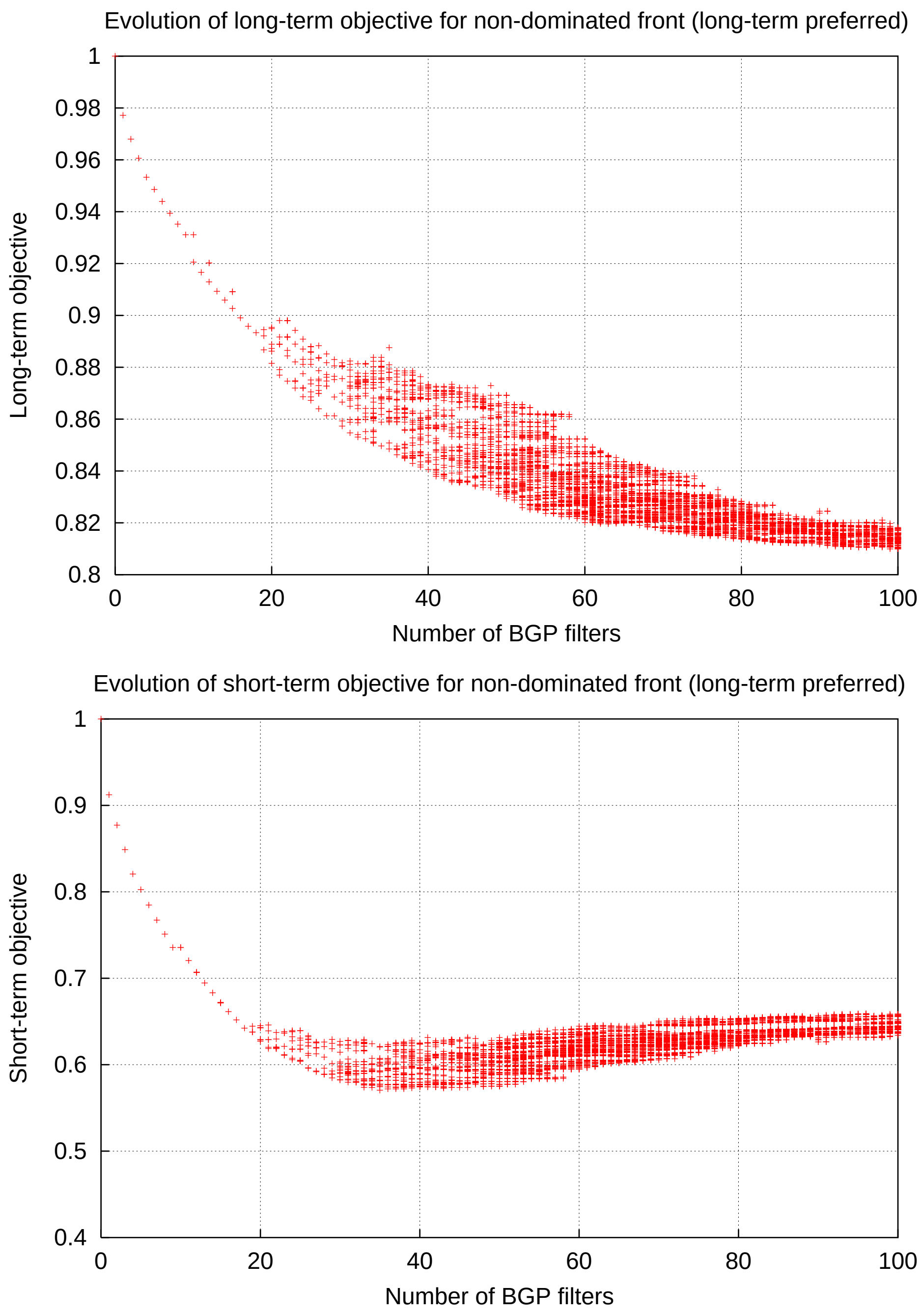



Figure 7.7: Projection of non-dominated front on each traffic objective for long-term preferred objective.

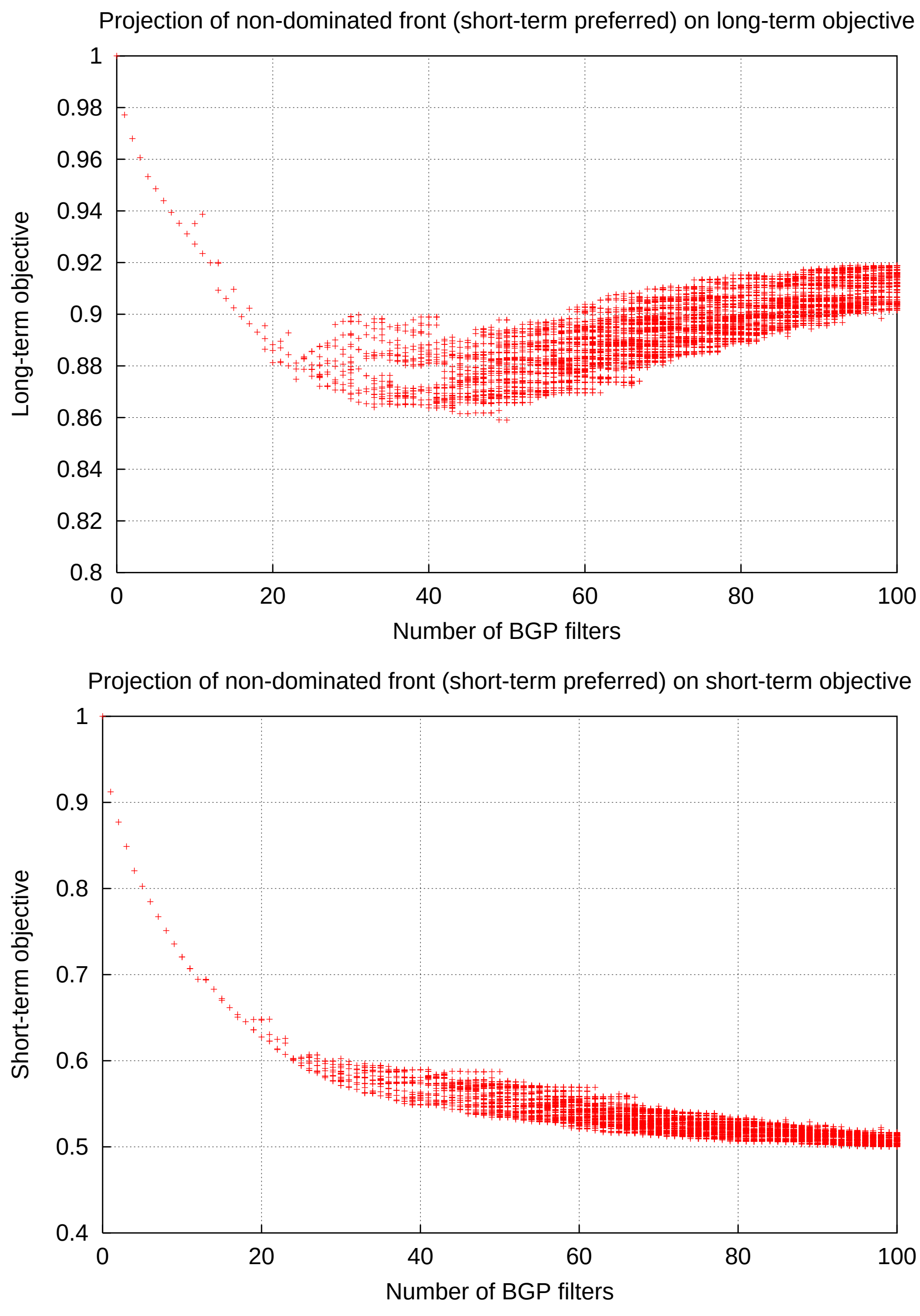



Figure 7.8: Projection of non-dominated front on each traffic objective for short-term preferred objective.

used is not an artifact of the sampling of the non-dominated front we performed through the crowding distance based selection of section 7.4.1. Hence the two objectives are really conflicting since when sampling the “best” solutions in terms of the short-term traffic objective, we find that for a given number of BGP filters, solutions having a better value of the short-term objective have a worse value of the long-term objective.

When we compare the points found by the algorithms with and without bias towards any traffic objective, we also find that the best fifteen BGP filters are found by the three methods. This constitutes further indication that the first non-dominated points found for the few first BGP filters are really local optima that would be found by any good local search method. The further BGP filters on the other hand concern such a large search space that it is likely that any search method would have a hard time finding the optimal solution, while the expected gain is quite small. As said previously, the more BGP filters to be applied the smaller the expected gain in terms of any traffic objective, only because of the distribution of the traffic among the prefixes. Only relying on some additional traffic aggregation by leveraging the knowledge of the AS-level topology might provide larger traffic shifts with a single BGP filter. Chapter 5 found that that the gain of leveraging the AS-level topology to reduce the number of BGP filters works for a limited number of BGP filters, and to approach the optimal value of one traffic objective will require a significant number of BGP filters, no matter the search technique. Drawbacks of taking into account the AS-level topology are the increased size of the search space and the overlap among the traffic influenced by different BGP filters.

Investigating the optimal value of a single long-term traffic objective while minimizing the number of BGP filters leveraging the AS-level topology is currently under way. It must be borne in mind however that even the optimal gain for a given number of BGP filters will not be able to compensate the distribution of the traffic for the interdomain destinations although the optimal solution could leverage more efficiently the AS-level topology than we did in Chapter 5.

Based on the simulations of this section, we feel confident that the crowding distance based method works well with conflicting traffic objectives since it finds good BGP filters and it seems to sample well the non-dominated front. This is not forcibly valid for other traffic objectives. We investigate this aspect in the next section with additional simulations.

### 7.4.3 Percentile-based billing

Up to now, we have used one short-term and one long-term objective function which are not those that are most likely to be used in practice. In this section, we compute a few non-dominated front for more realistic traffic objective functions to confirm the behavior of our algorithm on it. Note however that in practice the short- and long-term traffic objectives need not be conflicting, so the shape of the non-dominated fronts could be different than those found in the previous section.

Volume-based billing is most simple as a long-term cost function. In practice, billing is often done according to a $95^{th}$ percentile of the average bandwidth (in Mbps during each time interval) computed over equal time intervals of a few minutes. We use ten minutes time intervals in this section. We also use different bandwidth prices for the commitment and the traffic above the commitment, and different prices for the different providers. For the short-term traffic objective, we still use the load balancing objective that tries to minimize the sum over each short-term time interval of the maxima of the traffic sent through any provider. In addition, we change the different providers available by reducing their number to three and choosing three different types of providers : a tier-1 (AS1239), a tier-2 (AS1668) and a tier-3 (AS11608) provider. The reason for this choice is to limit the effect of the non-deterministic BGP routes due to relying on tier-1s only. Note that the classification as tier-1, tier-2 and tier-3 for these ASes comes from [157].

There was a total traffic for the considered day of about 215 gigabytes, a little less than 20 Mbps on average. We use a value of 7.5 Mbps for the commitment for each provider so that we can see during the search how many BGP filters are needed in order for the $95^{th}$ percentile to be under the commitment. If the search is to find BGP filters that allow the $95^{th}$ percentile to be under the commitment, then the long-term traffic objective cannot be improved, so only the short-term objective can improve from that point. Any traffic above 7.5 Mbps on average over the whole day gets billed fifty percent over the price below the commitment, for any provider. The traffic below the commitment gets billed using a constant cost, each provider bills the commitment whether or not the traffic is well above it or equal to it.

The non-dominated front for the long-term percentile-based traffic objective is provided on Figure 7.9. We see on Figure 7.9 that there is no smooth non-dominated front even for a large number of BGP filters, in contrast to the results of the long-term cost objective of section 7.4. The explanation for this phenomenon is the stochastic nature of the percentile-based objective which largely depends on the short-term dynamics of the traffic. Indeed, the value of the $95^{th}$ percentile depends on the distribution of the values of the traffic for each provider and each short-term time interval. Changing the provider used to carry the traffic for some prefix has a non-trivial effect on the value of the percentile. A cost function as used in section 7.4 is insensitive to the short-term traffic variability for some prefix, while the percentile is. A percentile-based cost function is hence an extremely difficult long-term traffic objective to optimize.

Figure 7.10 provides the projection of the non-dominated front for the percentile-based long-term traffic objective on the two traffic objectives. The projection on the long-term objective (top of Figure 7.10) illustrates quite well the difficulty of sampling evenly the non-dominated front for the percentile-based objective. The stochastic nature of the percentile-based objective prevents our search to be able to find improved individuals at successive generations. Correctly sampling the non-dominated front with such an objective is probably an interesting challenge. For what concerns the short-term load-balancing objective, the projection of the non-dominated front (bottom of Figure 7.10) has a look more similar to those of the previous simulations. Relying on several sub-populations each focusing on a subpart

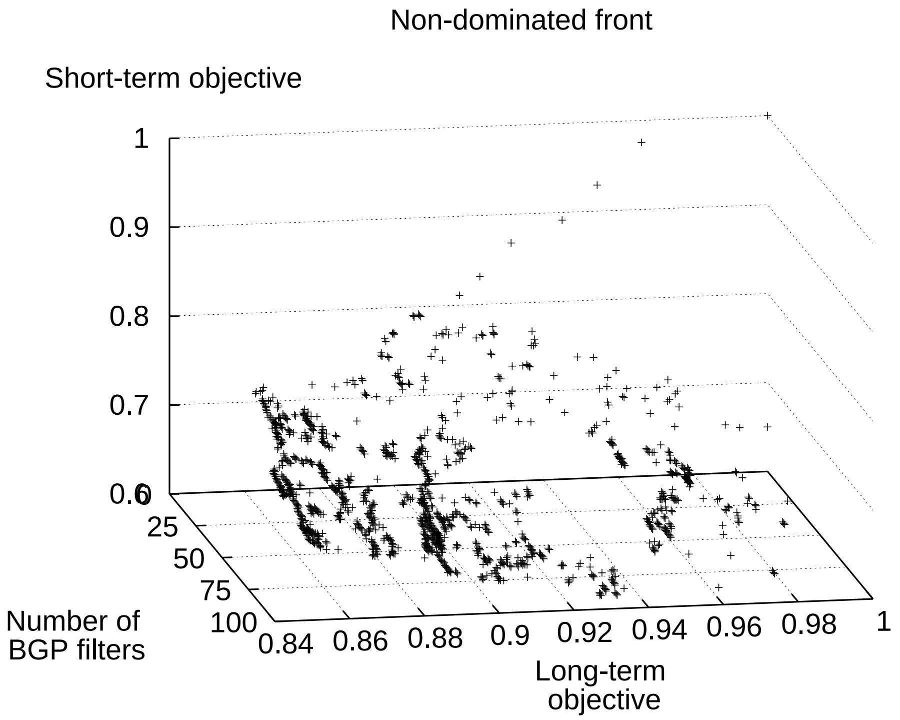


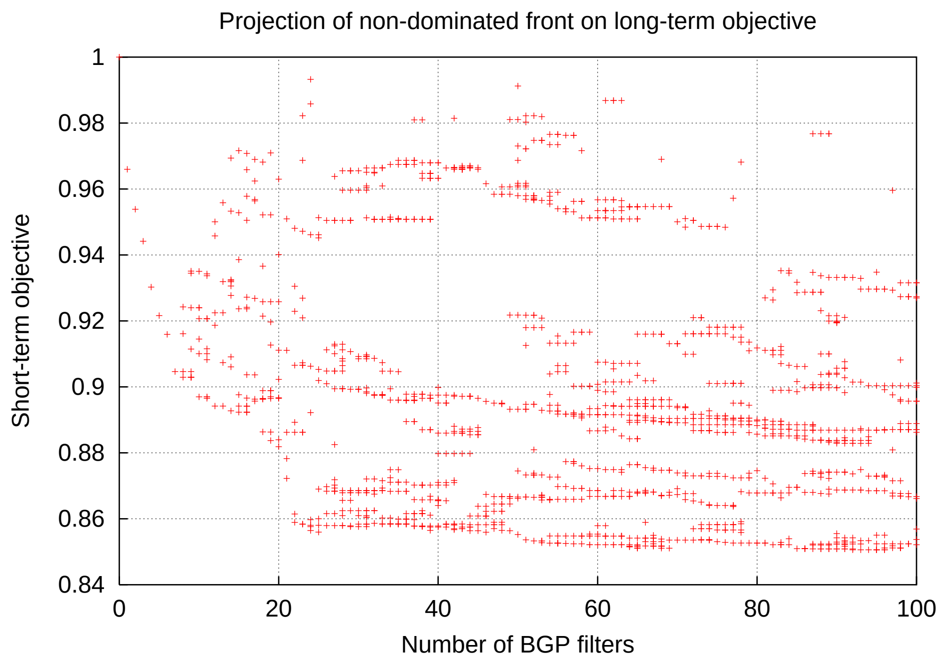

Figure 7.9: Solutions of the search with percentile-based long-term objective.

of the non-dominated front could solve the problem of finding the best values for a given objective, but it is difficult to foresee whether such a solution would actually provide a better sampling of the non-dominated front.

### 7.4.4 Load-balancing objectives

The last scenario we evaluate in this chapter is the same as the previous one except that the long-term objective is load balancing. This scenario should be the easiest one for what concerns the long-term traffic objective (see Chapter 5), but we have no a priori knowledge of its relationship with short-term load balancing due the short-term dynamics of the traffic for prefixes. A long-term cost function and a short-term load balancing objective might appear at first conflicting since minimizing cost tends to move as much traffic towards the lower cost providers, hence making the load among providers uneven. Load-balancing as the two traffic objectives on the other hand might appear harmonious, so at least improving the long-term balance should improve the short-term one on average. In the same way, one might think that improving the short-term balance is likely to improve the long-term balance. In practice, the validity of these intuitions depends to a large extent on the dynamics of the traffic over the short-term.

Figure 7.11 shows the non-dominated front for the two load-balancing objectives. We do not have a good looking front, but there are few non-dominated points for



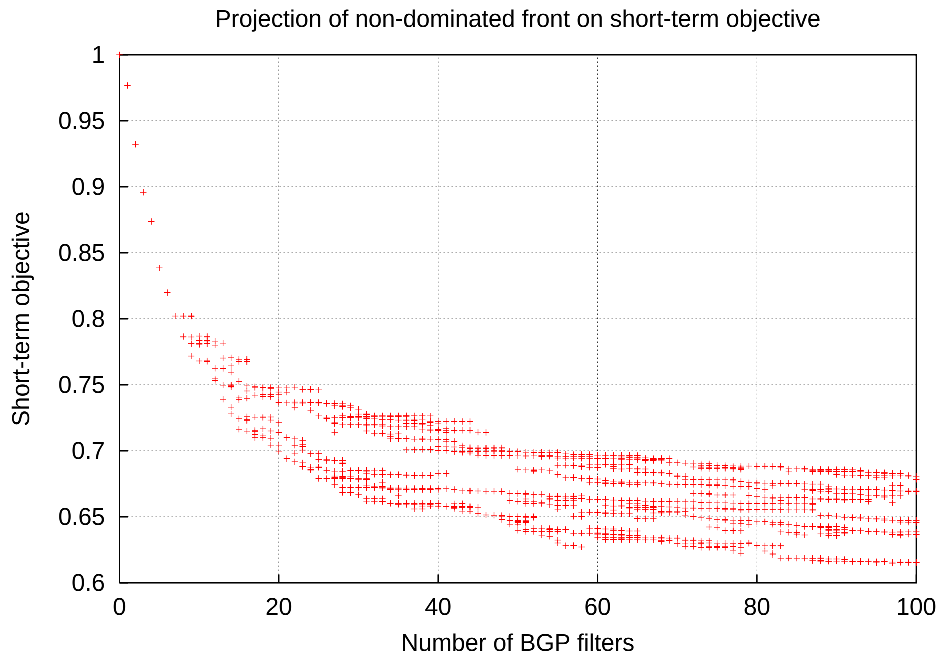




Figure 7.10: Projection of non-dominated front on each traffic objective for percentile-based long-term traffic objective.

each value of the number of BGP filters. Most of the gain is once more achieved through the first twenty BGP filters. The next twenty BGP filters still provide a non-negligible gain in terms of both traffic objectives but after forty BGP filters the search appears to be stuck. It could be due to an actual difficulty of improving the traffic objectives or to the size of the search space that prevents our algorithm to find improvements as well.

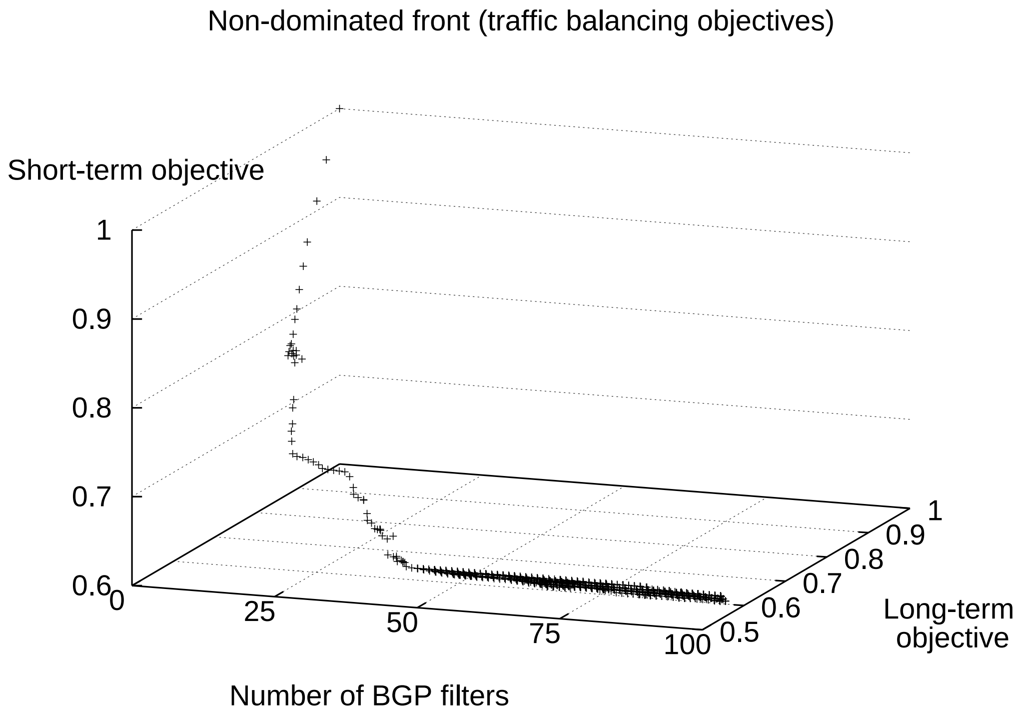



Figure 7.11: Solutions of the search for load balancing objectives.

Figure 7.12 provides the projections of the non-dominated front on the two traffic objectives. The absence of a well-spread non-dominated front indicates that the two load-balancing traffic objectives are indeed not conflicting. If they were conflicting we would be able to find a non-dominated front since we sample both objectives without biasing the search towards any of them. This simulation thus indicates that load balancing on both the short and longer timescales is possible even through a limited number of BGP filters.

## 7.5 Conclusion

In this chapter we proposed an evolutionary algorithm to evaluate multiple-objectives interdomain traffic engineering. The search consists in finding the BGP routing changes in order to optimize two objective functions defined on the interdomain traffic.

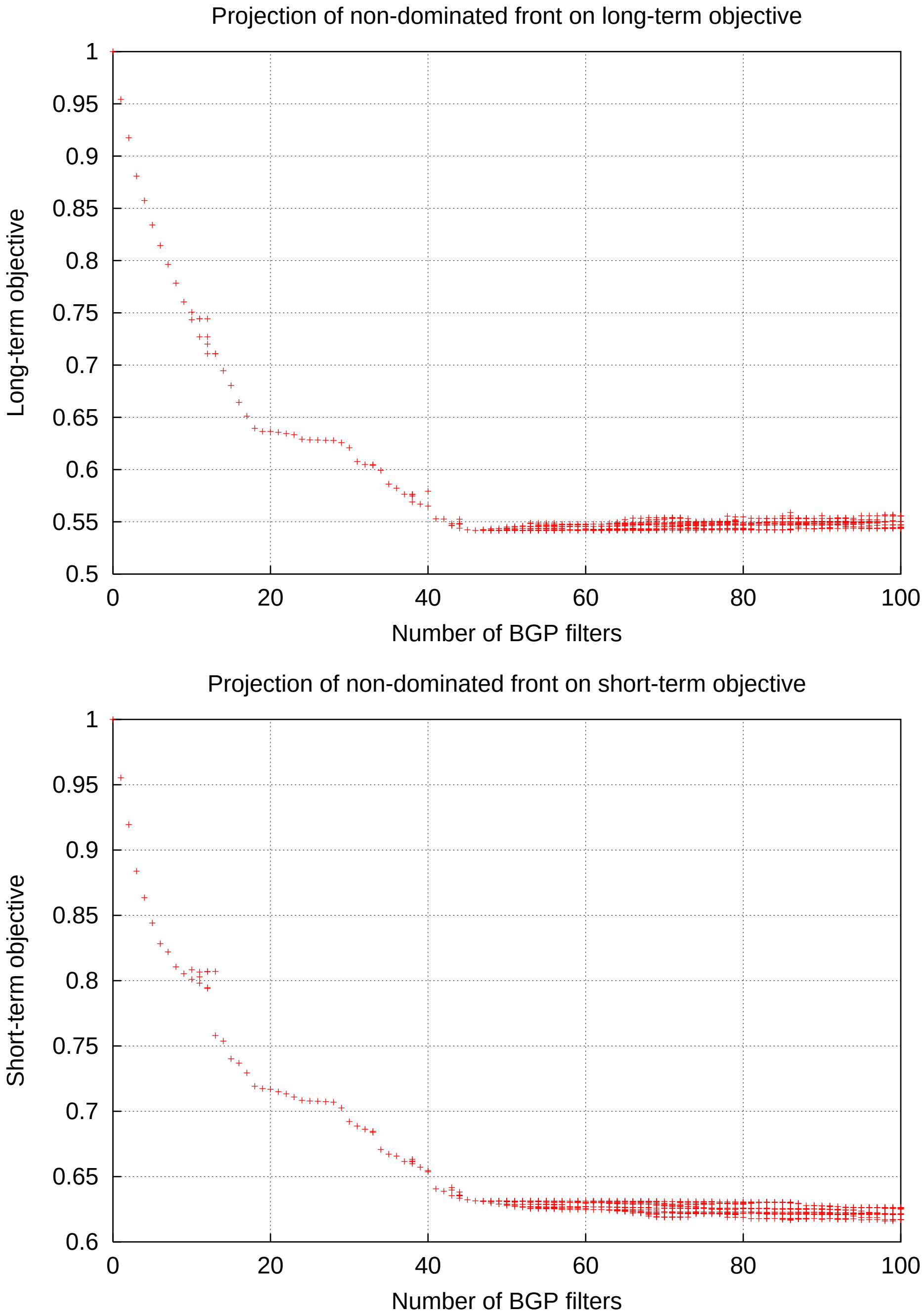



Figure 7.12: Projection of non-dominated front on each traffic objective for load balancing objectives.

To evaluate this problem, we relied on a real interdomain traffic trace and BGP routing tables from Oregon route-views. We studied two simultaneous traffic objectives, one short-term objective and one long-term objective, and evaluated different realistic instances of the traffic objectives.

The search showed that the problem is difficult. The different objectives of this problem can be conflicting and concern different timescales of Internet traffic. In addition, realistic objectives can be stochastic, depend on the dynamics of the Internet traffic and be interdependent. This chapter confirmed the results of the Chapter 5 showing that traffic balancing objectives are relatively easy optimized. We showed in section 7.4.4 that in theory it would be possible to balance the traffic both on a daily and a hourly timescale at the same time, relying on a limited number of BGP route changes (about 40 in our simulations). For daily traffic objectives like a volume-based billing cost or percentile-based billing cost on the other hand, these objectives might be conflicting with short-term traffic balancing. Furthermore, realistic traffic objectives like a percentile-based billing cost are difficult to optimize due to their dependence on the short-term dynamics of the traffic.

This chapter confirmed the problem that lies within the BGP decision process, namely the tie-breaking rules that make the balance of the traffic over several BGP neighbors particularly uneven. This aspect is really an issue for traffic engineering tools because it means that the optimization of some traffic objective might spend valuable time trying to counteract “bad” choices made by the tie-breaking rules of the BGP decision process.

# Part IV

# The future of interdomain traffic engineering

In Chapter 8, we evaluate the current state of interdomain traffic engineering and intent to foresee its future. We discuss the current limitations of the interdomain Internet that prevent interdomain traffic engineering to be practically feasible. We point to the further work on the Internet architecture that would ensure that the interdomain Internet be well designed for tomorrow's requirements. We also point for some further work in interdomain traffic optimization. Chapter 9 closes this thesis by summarizing its contributions.

# Chapter 8

# Evaluation and future work

Throughout this thesis, we have tried to look at interdomain traffic engineering from completely different perspectives. In the first part, we took the protocol-centered viewpoint of BGP. We saw in this first part the different aspects of the working of BGP that were important to perform interdomain traffic engineering. In the second part, we took the viewpoint of the interdomain traffic by analyzing its properties. Finally, in the third part we took the viewpoint of optimization by trying to evaluate the optimization of the interdomain traffic. The purpose of this chapter is to discuss the interrelationships between the different chapters of this thesis and their implications on the feasibility and future of interdomain traffic engineering.

## 8.1 BGP

Part I already discussed the complexity of tweaking BGP to perform interdomain traffic engineering. It illustrated the complexity of the BGP decision process (see Figure 5) and the interactions between its different rules made the tweaking of the BGP routes a non-trivial problem. Several issues concerning the BGP decision process must be discussed here. Chapter I discussed the difficulty of interdomain traffic engineering but only with respect to the BGP attributes and the BGP decision process. Here we extend the discussion to a broader perspective.

The first problem with the BGP decision process is related to policy routing. The first rule of the BGP decision process emphasizes the importance of local preferences on the choice of the best BGP route. Each AS decides which next hop AS it will use (among those possible by its available BGP routes) for IP packets entering the local domain and having as destination a prefix outside the local AS. This choice in BGP implies that the end-to-end AS path between a source and a destination depends on which ASes on the path modify the `local-pref` attribute of the BGP routes and its impact on the choice of the best BGP route by each intermediate AS on the end-to-end path. Routing policies used in the AS-level topology impact the way traffic enters multi-homed ASes [159, 80].

The second rule of the BGP decision process uses the shortest `AS path` length to choose the best route. While this might seem an appropriate distance metric for the routes, the number of AS hops have a limited meaning. First, some networks do not have an AS number. To obtain an AS number, one must be connected to the Internet through two different providers. Networks who are not multi-connected cannot obtain an AS number and the AS path to attain them from other ASes will henceforth be shorter than AS paths of networks that do have an AS number. Some large transit providers have several AS numbers for different reasons, ISPs can merge or split. Finally, the path across each AS depends on the diameter of the AS at the IP layer (in terms of router hops). Using the AS path length as a distance metric is thus not forcibly meaningful [92].

Next, the two rules concerning MED and IGP cost also constitute a local metric. As explained in Chapter I, these metrics represent an intradomain cost. For the IGP cost, it is the cost of the path up to the exit point of the local domain. For the MED attribute, it is often the IGP cost of the path that crosses the neighbor AS to reach the destination. These costs concern the local AS or its next AS hop, so these costs have only a local meaning.

The seventh and last rule of the BGP decision process (Figure 6) ensures that only one route will be selected for each destination prefix. In the context of large transit providers for which the previous rules are due to decide the best route, the last rule of the BGP decision process should not be heavily used. Actually, we do not really know since it depends on how much large transit providers actually tune their BGP routes to follow the best intradomain path according to the IGP routing [104]. For all other ASes on the other hand, this last rule as it is currently implemented systematically biases the choice of the best route towards the one whose `next-hop` router-id has the smallest IP address [138, 42, 46]. This rule implies that if the set of BGP egress points is relatively small and the set of different routes for the same destination prefix that are still non-differentiated by the BGP decision process is often the same, then this last rule will not distribute evenly the best routes among the various equivalent egress points. A better choice for this last rule of the BGP decision process would have been to rely on a hash function that would choose the best route in a more "random" way in order to prevent a systematic bias towards the same BGP `next-hop`. The choice of the actual means to improve the last rule of the BGP decision process however requires care to prevent unwanted transitions between external peers [42].

From the previous remarks, we have realized that the attributes of the BGP routes and the rules of the BGP decision process were questionable from an interdomain traffic engineering viewpoint. What is often considered as the really important features of BGP are policy routing and the route filtering mechanism. The BGP decision process aims at ensuring that only one route for each prefix will be chosen in every domain. The purpose of policy routing is to permit to prefer some routes over others for each prefix [155, 102, 138, 40]. This feature was explicitly considered as important for all interdomain routing protocols [155, 102, 138, 40]. This mechanism must be differentiated from route filtering and route redistribution [155, 102]. [40] explicitly discussed the question of restricting route information distribution but the reasons for limiting the redistributed routes were information hiding and resource consumption limitation. However, no study evaluated these aspects to the best of our knowledge. Route filtering allows the local AS not to accept some BGP routes. This possibility also means that the discarded routes cannot be redistributed to other peer ASes hence a potential information loss about the AS-level topology. In addition, the process of the redistribution of the BGP routes currently permits that only one route be announced to each BGP peer router for each prefix. This further restricts the spread of the topological information contained in the BGP routes as well as the variety of the different routes available to reach a particular destination. The problem solved by policy routing and route filtering are different. Policy routing is aimed at influencing the local choice of what the local AS considers to be the best route to reach a given destination. For this problem, nothing else than locally changing the attributes of the BGP routes is required. Filtering on the other hand solves the problem of topological information hiding and BGP attributes manipulation (AS path prepending, MED, communities). These two problems need not be coupled, at least not in the working of the interdomain routing protocol as it is the case currently with BGP.

## 8.2 Interdomain traffic dynamics

Chapter 1 aimed at capturing part of the timescale-dependent dynamics of the interdomain traffic. Self-similarity and scaling have been quite extensively studied in the context of networking [129, 62]. After more than a decade of research since the first paper on network traffic self-similarity [108], this topic is still (largely) misunderstood. There seems to be a gap between the understanding of the mathematical concepts and the physical emergence of scaling in network traffic [186]. [51] were the first to propose a physical explanation for self-similarity in Web traffic through the distributional properties of the flows. [163] then showed that a physical explanation could be found in the rise of self-similarity in network traffic through an aggregation process with independent ON/OFF sources and heavy-tailed ON/OFF distribution times. [163] thus formally proved the possibility for the presence of self-similarity in the traffic without dependence among the traffic sources. This however did not prove that self-similarity in the traffic **is** due to heavy-tails in the ON/OFF times distribution of the sources. However, [186] showed that the ON/OFF model is able to generate self-similar processes of different types like fractional Brownian motion and alpha-stable processes [144, 97, 160], that seem to match the behavior of network traffic [162, 121].

[173] showed that over timescales between minutes and hours, the sample path properties of the process of the number of IP sources, prefixes and ASes that are active at any given instant also constitute a self-similar process on a one week trace of all the incoming traffic of a stub AS. The implication of [173] is that independent of the assumptions on the dependence between the traffic sources and their ON/OFF times durations, the simple fact that the time evolution of the number of sources (at different aggregation levels) is a self-similar process is sufficient for self-similarity

to arise in the total traffic. [177] and Chapter 1 also showed that the process of the TCP flow arrivals had properties that could also explain the presence of self-similarity in the traffic. It is important to understand that [173] does not question the ON/OFF model. [187] confirmed that the ON/OFF model is likely to be right at the source level, i.e. for source-destination pairs at the IP level. What [173] showed is that independent of the validity of the ON/OFF model for source-destination pairs, the process of the number of network prefixes and autonomous systems that send traffic to the local network is sufficient to cause self-similarity in the traffic. This self-similarity could in turn be due to an ON/OFF model at the level of the network prefixes and autonomous systems. This is to be investigated in the near future, because it could imply that the Internet topology is partly involved in the emergence of self-similarity at the source level.

The question of what is the "true" cause of self-similarity in the traffic is probably without answer. This might seem a disturbing answer but searching for physical explanations can be wrong at times [35]. Some properties of complex systems can be "emerging", in the sense that they are properties of the system itself as a whole, not of some identifiable parameters of the system. As engineers, we tend to reason in terms of causes and effects, but this way of thinking is overly simplistic. Whenever some protocol or control variable impacts the system in some way, then one can study the relationship between this protocol or variable and the dynamics of the system. Causes and effects have a meaning in that case, since there can be a functional relationship between the system and its variables. In the case of a protocol, one can study the impact of the state machine defining the behavior of the protocol and the behavior of the system. This is because the state machine acts according to well-defined rules. In the Internet on the other hand, the traffic is generated by users (humans or machines) that do not follow precise rules or whose interactions are too complex to analyze. Therefore, the reason for the presence of self-similarity in the traffic might be the distributional properties of the flows as well as simply the behavior of the users, as we found in [173]. If such is the case, then it is possible that self-similarity in network traffic is an invariant that does not depend on the distributional properties of the flows but rather on the behavior of the users whose ensemble behavior generates a self-similar process. This discussion is not mere philosophical discussing about self-similarity, but has far-reaching engineering implications. If self-similarity in the traffic is actually an emerging property of the way users are active as a whole, it means that self-similarity is not an engineering problem because it is not rooted in the protocols or the properties of the various applications. This would mean that controlling the protocols to prevent self-similarity to arise will not work. Self-similarity would then have to be accepted as an inevitable fact for traffic engineering.

## 8.3 Topological distribution of interdomain traffic

Chapter 2 contributed to our understanding of how the traffic is distributed on the AS-level topology, as seen from two stub ASes. It showed that the ASes with which the stubs communicated were not far away in terms of the AS hop distance, between two and four hops away. We also saw that the traffic gets splitted in a tree-like manner, with limited traffic aggregation occurring in the AS-level topology, mainly at the peerings between the local stub and its provider, but not beyond. The results of the traffic analysis provided in Chapter 2 could be regarded as highly dependent on the particular traffic traces we used.

To assess the representativeness of the results of Chapter 2, we used the AS-level topology of [157] based on several BGP routing tables and computed the shortest AS path length between any two stub ASes of the topology. Figure 8.1 presents the distribution of the AS path length for the 11,976 stubs among the 14,695 ASes of the topology built with BGP tables dating from January 9, 2003. There were 30,815 distinct inter-AS links in this topology. Among the 11,976 stub ASes that the topology included, 4889 were single-homed and 7127 were multi-homed. The bars labeled "shortest path" on Figure 8.1 show the distribution of the AS path length of the shortest AS paths from every stub AS of the AS-level topology to every other stub AS. The "shortest path" in the topology does not take into account the routing policies applied by the intermediate ASes, so the shortest AS path from one stub to another is the shortest AS path that could be found given the available inter-AS edges available in the topology.

The topology provided by [157] corresponds to the AS paths seen in the BGP routing tables gathered from several vantage points mainly in the core of the AS-level topology, these are not necessarily the real AS paths that would be used by each stub. The AS paths given in the BGP tables are only valid for traffic sent by the ASes from which the BGP tables were gathered towards the destination prefixes of the routes of the BGP table. The reverse path however might not be valid. Because we cannot hope to obtain the BGP routing tables from all stubs of the Internet, we simulated the use of a common routing policy by all ASes [87]: a BGP route learned through a customer is preferred to a route learned through a peer, the latter being preferred to a route learned through a provider. [157] classified the peering relationships between the ASes in the AS-level topology as either "customer-provider" or "peer-to-peer". Based on this classification, we simulated the preference of customer routes to peer routes to provider routes. Each stub announced a single prefix to its neighbors. The advertisement of the BGP routes across the whole topology were simulated with the whole BGP decision process that chooses its best route according to the previously mentioned preferences. The bars labeled "customer preferred" on Figure 8.1 provide the distribution of the AS path length from each stub to each other stub of the topology from [157] when applying the "customer preferred" routing policy.

The top graph of Figure 8.1 shows the distribution of the AS path length for all stubs of the topology, while the middle graph shows it for the 4889 single-homed stubs and the bottom graph for the 7127 multi-homed stubs. For the middle and bottom graphs of Figure 8.1, we show the distribution of the AS path length for the AS paths used to attain every single-homed and multi-homed stub from every other stub AS (both single and multi-homed).

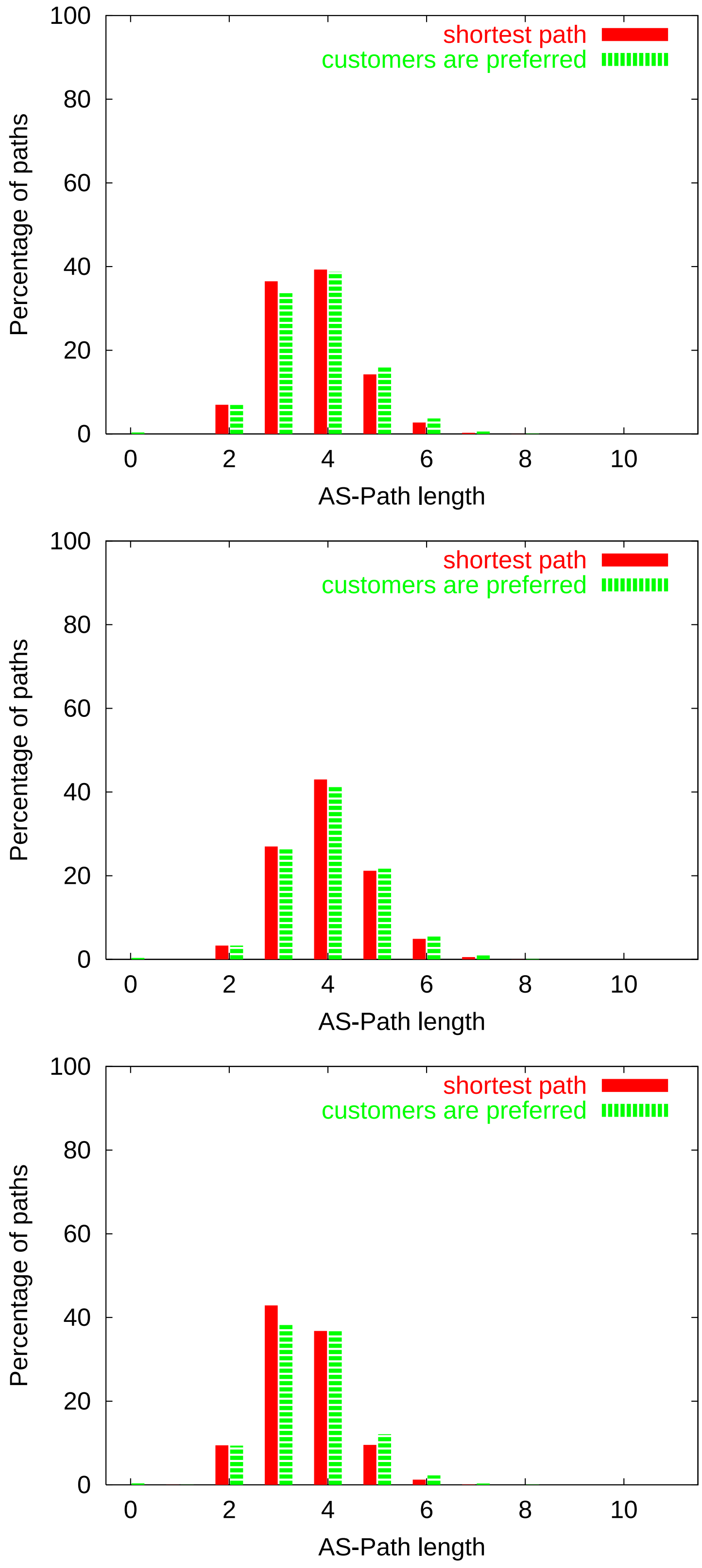



Figure 8.1: Distribution of AS path length for stub ASes in the Internet: all stubs (up), single-homed stubs (middle), multi-homed stubs (bottom).

The typical AS path length is between two and six, with most AS paths having length three or four. The top graph of Figure 8.1 shows the limited effect of the "customer-preferred" routing policies on the distribution of the AS path lengths. The distribution of the "shortest paths" shows that the typical AS path length is three or four. AS path lengths smaller than two or larger than six are rare. The effect of the routing policies is to slightly increase the AS path length compared to the shortest AS path, with an increase in the percentage of the AS paths of length five and six and decreases in the percentages of the AS paths of length three and four.

The middle and bottom graphs of Figure 8.1 show the difference between single-homed (middle) and multi-homed stubs (bottom). Compared to the distribution for all stubs (top), single-homed stubs have a smaller percentage of AS paths of length three while a larger percentage of AS paths of length five. The distribution for multi-homed stubs on the other hand is similar to the one of all stubs (top), with only slightly more AS paths of length three and less AS paths of length four and five.

The previous simulation shows that the results of Chapter 2 are not just a bias due to the choice of our stub ASes. The structure of the Internet is mainly responsible for this topological distribution of the traffic. What happens is that stub ASes get their Internet connectivity through tier-1, tier-2 or tier-3 providers, and it is mainly how stubs are connected to the Internet that explains the distribution of the AS path lengths seen by the interdomain traffic. Most AS paths contain two parts, the first part goes from the source AS to the core of the Internet (tier-1 providers), and then from the core to the destination AS. Since most ASes in the Internet are stubs, the interconnection of the stubs to the core prevails over the interconnection structure of the core.

We computed for these simulations the fraction of the AS paths due to the different levels of the hierarchy. We removed from the AS paths the first and last AS, and computed how many times an AS from each level of the hierarchy appeared in this "transit" part of the AS path, when the routing policies preferring customer routes over peer routes over provider routes. Since there were 14,695 ASes in the topology, we had 14,695 $\times$ 14,694 AS paths. We had on average 2.73 ASes in this transit part of the AS paths, among which about 42 % of tier-1's, 23 % of tier-2's, 24 % of tier-3's, and 10 % of tier-4's. These numbers show that BGP tables see almost half of the transit part of the AS paths that belong to tier-1's.

To illustrate on an example the impact of the hierarchical structure of the AS-level topology on the AS path length between ASes, Figure 8.2 presents a simplified view of the AS-level hierarchy. We show on Figure 8.2 four levels of the hierarchy. The core is made of tier-1's, to them are connected tier-2's, then the next level are tier-3's and finally stubs.

Consider the stub AS $S1$ as the local AS. $S1$ is directly connected to a tier-1, $P1$. $P1$ is connected to three providers: another tier-1 $P2$, two tier-2 providers $P3$ and $P4$. $P1$ has also a customer stub AS, $S2$. Stub AS $S3$ is connected to the Internet through tier-2 provider $P3$. $S4$ is a dual-homed stub AS connected to the Internet through a tier-1 ($P2$) and a tier-2 provider, $P4$. Finally, stub AS $S5$ is connected

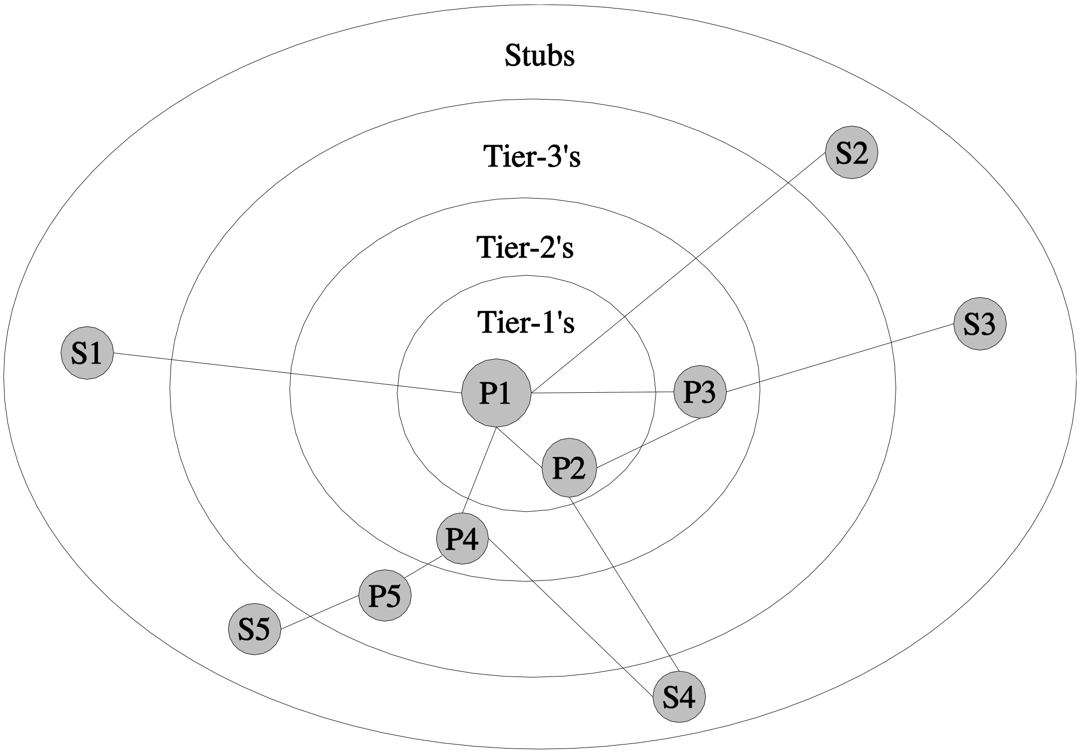



Figure 8.2: Impact of hierarchical structure of the AS-level topology on typical AS path length.

to the Internet through a small tier-3 provider, $P5$. If we compute the shortest AS path length between stub $S1$ and the other stub ASes, we have :

- $S1 - S2$ : AS path is S1-P1-S2, length = 2,
- $S1 - S3$ : AS path is S1-P1-P3-S3, length = 3,
- $S1 - S4$ : AS path is S1-P1-P2-S4, length = 3,
- $S1 - S5$ : AS path is S1-P1-P4-P5-S5, length = 4.

On the simple example of Figure 8.2, we can see why based on the AS-level topology of [157] most stub ASes are reachable within two to four AS hops depending on how they are themselves connected to the Internet.

The previous simulation indicates that if the structure of the Internet does not change fundamentally and the ASes with whom a stub communicates with samples randomly the other stub ASes of the topology, then on average the topological distribution of the traffic will have the same look as the distribution of the AS path length shown on Figure 8.1.

The topological properties of the traffic are not simply facts that are interesting to know. For outgoing traffic engineering, the control of the routes is performed locally, so the topological properties are not overly important even if we shall see in the next section that it has implications on the tweaking of BGP routes. For incoming traffic engineering on the other hand, the properties of the topology are important. Influencing the incoming traffic requires that one be able to influence the source as well as the intermediate ASes on the path up to the local AS. For incoming traffic engineering be feasible, one must understand whence the traffic comes over the interdomain topology since policy routing and AS paths determine how the interdomain traffic crosses the AS-level topology. First, the limited AS hop distance between the local AS and interdomain traffic sources requires that the control information aimed at tweaking the path of the incoming traffic need to travel across few domains. The corresponding bad news however is that the difference in terms of AS path length between alternative AS paths for a given prefix will also be limited. BGP tweaking through AS path prepending henceforth is likely to be a coarse technique, as shown in [29]. In AS path prepending, a multi-homed AS will announce different routes to its different peers. AS path prepending consists in inflating the AS path announced to a particular peer with the objective of switching part of the incoming traffic from this link. Take the example of AS $S4$ on Figure 8.2. $S4$ has two providers, $P2$ and $P4$. Suppose that $S4$ performs AS path prepending by adding its AS number one more time in its route announced to $P2$ hoping to reduce the load of this link. The effect of this prepending is that $P2$ announced to $P1$ and $P3$ that it can reach $S4$ through a route with an AS path length of 2. The shortest AS paths for each stub is indicated on Table 8.1. Before prepending, $S1$

Table 8.1: Effect of AS path prepending on shortest AS path and provider used to attain S4.

| Source | Shortest AS path before prepending | Shortest AS path after prepending |
|---|---|---|
| S1 | S1-P1-{P2\|P4}-S4 | S1-P1-P4-S4 |
| S2 | S2-P1-{P2\|P4}-S4 | S2-P1-P4-S4 |
| S3 | S3-P3-P2-S4 | S3-P3-{P2-S4\|P1-P4}-S4 |
| S5 | S5-P5-P4-S4 | S5-P5-P4-S4 |

and $S2$ had similar AS path lengths through $P2$ and $P4$. The choice of the actual provider used to attain $S4$ hence depends on the choice of the next AS hop by $P1$. After prepending, $S1$, $S2$ and $S5$ send their traffic to $S4$ through $P4$ while for $S3$ it depends on the choice of the next AS hop by $P3$. So in this case, the effect of prepending is that at least three among the four sources attain $S4$ through $P4$.

If $P1$ prefers the routes learned from $P4$ to those by $P2$ because $P4$ is a customer while $P2$ is a peer, then $S4$ receives the traffic from $S1$, $S2$ and $S5$ from $P4$ while the traffic from $S3$ from $P2$. The net effect of prepending is then either null if $P3$ prefers $P2$ to $P1$ or to force all traffic to be received through $P4$ if $P3$ prefers $P1$ to $P2$.

If on the other hand $P1$ does not prefer $P4$ to $P2$ (if both are peers) but the BGP decision process decides that the routes announced by $P2$ are better, then before prepending the traffic from $S1$, $S2$ and $S3$ is received though $P2$ and the traffic from $S5$ through $P4$. After prepending, the traffic from $S1$, $S2$ and $S5$ is received through $P4$ and the traffic from $S3$ through $P2$.

The simplicity of the example contrasts with the complexity of the effect of AS path prepending due to the BGP decision process. Incoming traffic engineering depends on routing policies, on the particular values of the route attributes (for the tie-breaking rules) and the set of available alternative paths known by each AS along the path between the AS that performs prepending and the traffic sources. We hope that this simple example was convincing enough to show that the effect of AS path prepending is difficult to foresee in practice. The most important issue with BGP is the filtering mechanism that hides much of the information about the AS-level topology to BGP routers. This information is however required to predict the effect of BGP tweaking on the incoming traffic distribution of stubs.

The limited aggregation of the traffic on the AS-level topology also raises the question of the granularity of the route control. Relying on AS path prepending or not announcing routes to some peers have a binary effect: either it does not work at all or it works by switching almost all traffic from one upstream provider to another. If one requires a finer control of the incoming traffic, a better way to go could be through communities [28]. As explained in section 0.5.3, the community attribute is an optional attribute of the BGP routes that allows an AS to influence the filtering process of the BGP routes of the BGP neighbors. This mechanism however requires mutual agreement between the ASes. This mechanism could in theory provide a finer control of the incoming traffic, by allowing ASes to indicate to their upstream providers the routing policy to be applied or the tweaking of the BGP attributes to be performed upon redistribution of these routes to their BGP neighbors. Nevertheless, communities are not a panacea. Communities merely defer the route control problem one hop away. It allows the BGP peers of the local AS to perform the BGP route manipulations on behalf of the local AS. If the BGP peer performs the BGP routes manipulations defined by the values present in the communities of the routes, this will have no effect on the routing policies applied upstream in the AS-level topology. So even if communities provide a finer control of the inbound traffic, it is not certain that they constitute a solution to this problem.

## 8.4 Topological dynamics of interdomain traffic

Chapter 3 presented a study of the dynamics of the traffic on the AS-level topology, in the case of two stubs. Generalizing the results of this chapter is not possible, because it could depend on the context of the studied traces like the users profile, the applications, the upstream peers of the stub,... Studying two traffic traces cannot ensure any representativeness. However, gathering such traces is very difficult. The main contribution of this chapter are however

1. the confirmation of [139] concerning the stability of the BGP routes for the traffic,

2. the confirmation of Chapter 2 concerning the limited traffic aggregation on the AS-level topology,

3. the observation that a significant percentage of the total traffic has stable AS paths while another significant part has unstable AS paths.

Point 1 requires few comments. The reason for observing stable AS paths probably does not go deeper than the relative stability of the routing policies that do not to change very fast. ASes do not dynamically change their best route towards a destination unless some failure happened in the network. Dynamic traffic engineering is not common, especially through BGP. Point 2 has already been discussed quite extensively in the previous sections. Point 3 on the other hand requires some care to properly understand its implications on traffic engineering.

"Stability" as mentioned in point 3 actually refers to the concept of "stability in presence" defined in section 3.6.3. In that section we looked at how many one hour intervals any AS path was present among the largest ones accounting for ninety percent of the hourly traffic. Given that the AS-level topology is assumed to be stable for the traffic, it is important to know what fraction of the traffic one can rely on to perform traffic engineering without too much uncertainty on the amount of traffic one actually is able to influence. To perform traffic engineering, one needs to know what amount of traffic will cross which outgoing or incoming link. If one has a high confidence on the distribution of the traffic that will be received on the incoming links for, say eighty percent of the total traffic, but one has no idea of through which incoming links the remaining twenty percent of the traffic shall come, then knowing whence in the AS-level topology this traffic comes is important. Depending on the probability that some traffic might arrive through some AS path, one can compute the likelihood of the distribution of the remaining twenty percent of the traffic on the incoming links. This in order to take into account the uncertainty concerning whether some traffic will come from some AS path. If only long-term traffic engineering is performed, this might not be a problem. If on the other hand traffic engineering focuses on strict QoS guarantees or simply on optimizing a particular cost function that requires that a target traffic distribution among the ingress or egress links be ensured, the problem is different. In a best-effort Internet, these issues do not matter. Whenever tight traffic engineering objectives need to be met then a precise understanding of the behavior of the traffic will be required.

## 8.5 Optimizing traffic objectives

Throughout Part III, we tried to assess the feasibility of optimizing traffic objectives by tweaking BGP. It is certainly this part that uncovered the extent of the issues that lie within BGP if it is to be used to perform interdomain traffic engineering. Because the various chapters of Part III already discussed in details the implications on traffic engineering with BGP, we shall only add a few comments in this section. In this thesis, we only evaluated the optimization of objectives for outgoing traffic engineering for stub ASes. In the case of outgoing traffic, the local AS has full control over the manipulation of the BGP routes. For incoming traffic, the lack of control on the traffic precludes to even mention the term "control". If traffic engineering means tweaking the path followed by the traffic, then inbound traffic engineering is possible. If however one uses the term traffic engineering as "traffic optimization" as we tried to do in Part III, then one should realize that even for outgoing traffic this task is far from being easy. Chapter 6 however provided relatively favorable results concerning the feasibility of on-line traffic engineering the outbound traffic, in the case of a traffic balancing objective.

BGP was designed as a reachability protocol that provided various features among which policy routing is one. Most metrics present in the BGP routes are local administrative costs. `Local-pref` enforces strict preference among the routes. MED and IGP cost allow local intradomain metrics to be taken into account. The `AS path` length has a limited meaning [92]. Speaking crudely, we would say that BGP does not care about traffic engineering.

Throughout Part III we observed that optimizing a simple traffic objective while trying to keep the burden on BGP low was not simple. The tie-breaking rules of the BGP decision process were shown to have an important influence on the "inoptimality" of the default way BGP chooses its best routes and distributes the traffic among the available egress points. BGP is not to blame since the BGP decision process works on the set of routes towards a given prefix, not on the total traffic. Our feeling after having carried out the studies of Part III is that BGP should not be used to perform long-term traffic engineering. If BGP is to be used for traffic engineering, then either the purpose should be to compensate the uneven distribution of the traffic over multiple links due to the BGP decision process or to perform on-line traffic engineering. BGP solves well the problem of local policy routing and local traffic deflection. The intent behind BGP was not to provide the necessary information so that all ASes be able to perform traffic engineering.

It must be noted that the techniques used in part Part III are relatively weak from a pure optimization viewpoint, in that the optimality of the solutions found could, in theory, be relatively far from the true optima. In practice, we are confident that the solutions found by our algorithms are indeed good, because the main limitation in the optimization of traffic objectives comes from the distribution of the traffic over the interdomain sources or destinations, not the search technique. Our aim however was not at determining the optimal solutions but rather to assess how difficult it was to improve objective functions by tweaking BGP. We shall work on improving the optimization techniques in the near future, but we feel that the importance of optimality is marginal compared to the networking-related issues.

## 8.6 The future of interdomain traffic engineering

Before discussing the future of interdomain traffic engineering, let us first deal with its present. Except for a few commercial solutions whose internal working is opaque [12], interdomain traffic engineering is in its infancy. Contrary to intradomain traffic engineering that deals largely with technical objectives [76], interdomain traffic engineering deals mostly with economical objectives. Traffic billing and peering agreements are complex and difficult to evaluate. If in the future well-defined technical objectives need to be achieved at the interdomain level, then it is likely that interdomain traffic engineering will become a widely spread practice. If the future of interdomain traffic engineering need to rely on BGP, then the development of interdomain traffic engineering could suffer.

If interdomain traffic engineering is a requirement for the future Internet, then either BGP must be replaced with an interdomain routing protocol that provides the necessary information, or tools must be developed to ease the deployment of interdomain traffic engineering techniques. A relatively simple guideline that should be followed during the next years in order to prevent an inadequate interdomain routing architecture for the future Internet would be the following: *Build in the interdomain routing architecture having in mind the basic requirements for the future Internet.* Admittedly, designing a routing protocol is not a simple task [91], and particularly for a large system like the interdomain Internet [89]. Failing to take into account the problems that the protocol is aimed to solve must be prevented. Our viewpoint is thus that if interdomain traffic engineering is not considered as a requirement for the future interdomain Internet, then the interdomain routing architecture should not be bothered by this objective [61, 60].

Independent of the evolution of the interdomain routing architecture, how do we envision the future of interdomain traffic engineering? One might consider that economical objectives drive the evolution of the Internet. Henceforth, the viability of interdomain traffic engineering would depend on the demand by ASes for interdomain traffic engineering tools. This reasoning simplifies a little bit how things actually work in the Internet. Thinking in terms of independent offer and demand in the Internet is wrong. Users of the Internet architecture (mainly providers) have requirements for the tools that they need to run their network. Protocol designers and hardware and software vendors also care for the requirements of the users of the Internet architecture. Users run their network based on the existing infrastructure and the currently available tools. Designers try to foresee the requirements of the users and try to meet the technical aspects of these requirements. In practice, network managers do not really know what they want, it depends on what is available on the market. Designers too do not exactly know what network managers want (since the latter's do not know themselves) but guess what the important directions for improvement could be. This is a vicious circle.

Interdomain traffic engineering is a marginal requirement today because there are few tools available to solve the problem and providers do not really understand how BGP works[1]. Few tools exist today for interdomain traffic engineering because few providers ask for it. Is there a way out of this circle? Perhaps. The practice of multi-homing could be this way out. [9] noticed a trend towards more multi-homing during the last few years. Multi-homed ASes need to rely on interdomain traffic engineering. An AS multi-connected to the Internet need to manage the load on its multiple link with its BGP peers. To correctly balance the traffic over several

[1] Or rather if some know, they do not seem to share this information with others

BGP peerings requires a careful tweaking of the BGP routes. These ASes will ask for better interdomain traffic engineering tools since BGP on its own does not allow to automatically balance the traffic. Multi-homed stub ASes represent nowadays (late 2003) less than half of all stubs in the Internet according to BGP tables from large transit providers [157]. So our feelings concerning the future of interdomain traffic engineering is that it will depend on whether the proportion of multi-homed ASes grows large enough to put a sufficient pressure on the demand for interdomain traffic engineering tools. If both the designers of the Internet architecture and its users spend enough time to think about the requirements of the future interdomain Internet then interdomain traffic engineering might have a bright future.

## 8.7 Further work

As emphasized in the previous section, the two most important further works that emerge from this thesis are

1. the requirements of the future interdomain routing architecture,
2. developing automatic interdomain traffic engineering tools.

Working on the future interdomain routing architecture is crucial because it will be both the main constraint for interdomain traffic engineering tools as well as its main basis. What information will be available through the interdomain routing protocol will guide the choice of the techniques upon which the interdomain traffic engineering tools will rely. On the one hand, the requirements for the interdomain routing architecture must be carefully studied to know what are the goals and what are the non-goals. Currently, policy routing is a goal of the interdomain routing architecture while accurate dissemination of the interdomain topology is not. Should the interdomain routing architecture be designed having in mind the requirement of providing accurate topological information and relying on meaningful and global metrics for global interdomain traffic engineering be possible ? This question must be answered during the forthcoming years.

Working on routing alone is not enough. The four chapters of Part III were nothing more than the first step towards a more thorough evaluation of interdomain traffic engineering techniques. We relied on evolutionary algorithms due to their robustness and flexibility to easily be adapted to the problem at hand. These chapters on the other hand provide only weak optimality bounds on the gains that could be achieved through interdomain traffic engineering. The importance of evaluating the efficiency of various interdomain traffic engineering techniques is probably as important as working on the interdomain routing architecture, because it is by evaluating a system that one learns its strengths and weaknesses.

The amount of information and the ability to control the interdomain routing should ideally depend on the expected gains of more information and more control provided by interdomain routing. A trade-off between optimality, robustness and flexibility must be expected. Bringing a system to optimality requires a perfect information about its state and perfect control on its behavior. Optimality cannot be ensured under uncertainty, nor can one expect robustness to be possible if global and centralized control of the system is performed. The situation can be illustrated through comparison of the intradomain and the interdomain routing architectures in the Internet. At the intradomain level, the knowledge of the topology is good, the link state routing protocols distribute the whole intradomain topology to all routers. Globally optimizing the weights of the links is hence possible at the intradomain level [77, 76]. At the interdomain level on the other hand, a decentralized approach has been used, so that BGP routers only know the next hop AS to be used to reach a given destination and most metrics used for traffic control with BGP have a local meaning only. Globally optimizing the interdomain routing is thus not possible today, but the interdomain Internet is extremely robust. Important outages at the interdomain level have a limited impact on the reachability of most destinations. Today the interdomain routing architecture is designed to allow each AS to decide which routes it prefers to attain a destination. If advanced interdomain traffic engineering need to be performed, then ASes will at least need to allow other ASes to decide which route is best.

# Chapter 9

# Conclusion

Based on many real traffic traces, this thesis studied the implications of the characteristics of the interdomain traffic on traffic engineering. It evaluated the complexity and feasibility of interdomain traffic engineering with BGP, focusing on non-transit ASes.

Part I provided a detailed presentation of the working of the BGP protocol and the interactions between BGP and the control of the interdomain traffic. It showed the complexity of interdomain traffic engineering with BGP due to the complexity of the BGP decision process and the uncertainty for controlling the traffic, especially in the inbound direction, mainly due to the local policies applied by each AS.

Part II studied the characteristics of the interdomain traffic from a stub AS viewpoint. It showed that interdomain traffic exhibits wide fluctuations and that its variability reduces only on timescales larger than about one hour. The topological distribution of the interdomain traffic for two stub ASes showed that the traffic sent by a stub gets splitted on the AS-level topology within a few AS hops. Traffic aggregation on the AS-level graph occurs only with the direct peers of the stub. Finally, studying the dynamics of the topological distribution of the traffic showed that the AS-level graph seems stable for the traffic sent by a stub. On the one hand, the AS paths for an important fraction of the total traffic had a stable presence among the largest AS paths on a hourly basis. On the other hand, a significant fraction of the traffic had AS paths whose hourly presence among the largest AS paths is unstable in presence.

Part III evaluated the feasibility of interdomain traffic engineering by tweaking the BGP routing protocol. It relied on evolutionary algorithms to optimize objective functions defined on the interdomain traffic. Trying to optimize objective functions defined on the interdomain traffic for stub ASes was shown to be not trivial. We showed that the tie-breaking rules of the BGP decision process were largely responsible for the large distance between the default solution found by BGP and the optimum to be attained. Then, we showed that trying to minimize the number of BGP filters to optimize the cost or traffic balance of the daily traffic over the available providers was difficult. Trying to leverage the aggregation of the interdomain traffic on the AS-level topology to minimize the number of BGP filters was shown to provide a marginal gain mostly because of the limited aggregation of the traffic on the AS-level topology. Our evaluation of on-line interdomain traffic engineering provided the most positive results, indicating that BGP could be used to traffic balance the traffic while requiring a very limited number of BGP route changes every time interval. Finally, we studied the optimization of two traffic objectives, one defined on the short-term (ten minutes) and the other on the longer-term (one day). This study showed that traffic objectives working on these different timescales could be conflicting. Furthermore, we showed that percentile-based billing is a difficult objective to optimize due to its statistical nature and its dependence on the short-term traffic dynamics which is known to contain a lot of variability.

Part IV evaluated the current state of interdomain traffic engineering, discussed its issues and envisioned its future. It also provided a few guidelines for further work on the interdomain Internet. We discussed the limitations of BGP to perform interdomain traffic engineering. We then discussed the implications of the traffic characteristics on interdomain traffic engineering and showed that the topological distribution of the traffic was largely linked to the structure of the AS-level topology. Hence the topological distribution of the traffic is not due to change in the future unless the AS-level interconnection structure changes. Finally, we discussed two most important further work, namely the interdomain routing architecture and developing optimization methods to perform interdomain traffic engineering.

# References


[1] Abilene NetFlow Statistics. Available at `http://www.itec.oar.net/abilene-netflow/`.

[2] GNU Zebra routing software. `http://www.zebra.org`.

[3] Internet Traffic Archive. `http://www.acm.org/sigcomm/ITA/`.

[4] P. Abry, R. Baraniuk, P. Flandrin, R. Riedi, and D. Veitch. The Multiscale Nature of Network Traffic: Discovery, Analysis, and Modelling. *IEEE Signal Processing Magazine*, May 2002.

[5] P. Abry, P. Flandrin, M. Taqqu, and D. Veitch. Wavelets for the Analysis, Estimation, and Synthesis of Scaling Data. *In Park and Willinger (editors) "Self-Similar Network Traffic and Performance Evaluation", Wiley-Interscience*, 2000.

[6] P. Abry, P. Flandrin, M. Taqqu, and D. Veitch. Self-similarity and Long-Range Dependence through the Wavelet Lens. *In P. Doukhan and G. Oppenheim and M. Taqqu (editors), "Theory and Applications of Long-Range Dependence", Birkhäuser, Boston*, 2002.

[7] P. Abry, P. Gonçalvès, and P. Flandrin. Wavelets, spectrum estimation and $1/f$ processes. In *Wavelets and Statistics, Lecture Notes in Statistics*. Springer Verlag, 1995.

[8] S. Agarwal, C. Chuah, S. Bhattacharyya, and C. Diot. The Impact of BGP Dynamics on Intra-Domain Traffic. Sprint ATL Research Report Nr. RR03-ATL-080677, August 2003.

[9] S. Agarwal, C. Chuah, and R. Katz. OPCA: Robust Interdomain Policy Routing and Traffic Control. In *Proc. of IEEE OPENARCH*, April 2003.

[10] S. Agarwal and T. Griffin. BGP proxy community community. Internet draft, draft-agarwal-bgp-proxy-community-00.txt, work in progress, January 2004.

[11] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman. A measurement-based analysis of multihoming. In *Proceedings of ACM SIGCOMM 2003*, August 2003.

[12] D. Allen. NPN: Multihoming and route optimization: Finding the best way home. *Network Magazine*, February 2002. available from `http://www.networkmagazine.com/article/NMG20020206S0004`.

[13] C. Association for Internet Data Analysis (CAIDA). CoralReef Software Suite. `http://www.caida.org/tools/measurement/coralreef`.

[14] R. Atkinson and S. Floyd. IAB Concerns and Recommendations Regarding Internet Research and Evolution. Internet draft, draft-iab-research-funding-02.txt, work in progress, October 2003.

[15] P. Aukia, M. Kodialam, P. Koppol, T. Lakshman, H. Sarin, and B. Suter. RATES: A server for MPLS Traffic Engineering. *IEEE Network Magazine, pages 34–41*, March/April 2000.

[16] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and Principles of Internet Traffic Engineering. Internet Engineering Task Force, RFC3272, May 2002.

[17] D. Awduche, A. Elmalid, I. Widjaja, and X. Xiao. A framework for Internet traffic engineering. Internet draft, draft-ietf-tewg-framework-05.txt, work in progress, May 2001.

[18] D. Awduche, J. Malcom, B. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. Internet RFC 2702, September 1999.

[19] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms. Lawrence Erlbaum Associates (Hillsdale)*, 1987.

[20] J.-M. Bardet, G. Lang, G. Oppenheim, A. Philippe, and M. Taqqu. Generators of Long-Range Dependent Processes. *In P. Doukhan and G. Oppenheim and M. Taqqu (editors), "Theory and Applications of Long-Range Dependence", Birkhäuser, Boston*, 2002.

[21] S. Bartholomew. The art of peering. *BT Technology Journal*, 18(3), July 2000.

[22] J. Bartlett. Optimizing multi-homed connections. *Business Communications Review*, 32(1):22–27, January 2002.

[23] T. Bates, R. Chandra, and E. Chen. BGP Route Reflection - An Alternative to Full Mesh iBGP. Internet Engineering Task Force, RFC2976, April 2000.

[24] T. Bates, R. Chandra, and E. Chen. BGP Route Reflection - An Alternative to Full Mesh IBGP. Internet Engineering Task Force, RFC2796, April 2000.

[25] S. Bellovin, R. Bush, T. Griffin, and J. Rexford. Slowing routing table growth by filtering based on address allocation policies. preprint available from `http://www.research.att.com/~jrex`, June 2001.

[26] J. Bendat and A. Piersol. *Random data analysis and measurement procedures (third edition).* Wiley Series in Probability and Statistics, Wiley InterScience, 2000.

[27] J. Beran. *Statistics for Long-Memory Processes.* Monographs on Statistics and Applied Probability, Chapman & Hall, 1994.

[28] O. Bonaventure, S. D. Cnodder, J. Haas, B. Quoitin, and R. White. Controlling the redistribution of BGP routes. Internet draft, draft-ietf-ptomaine-bgp-redistribution-01.txt, work in progress, August 2002.

[29] O. Bonaventure, P. Trimintzios, G. Pavlou, B. Quoitin, A. Azcorra, M. Bagnulo, P. Flegkas, A. Garcia-Martinez, P. Georgatsos, L. Georgiadis, C. Jacquenet, L. Swinnen, S. Tandel, and S. Uhlig. Internet Traffic Engineering. Book chapter of Cost Action 263 final report, LNCS 2856, Springer-Verlag, pp. 118-179, September 2003.

[30] S. Borthick. Will Route Control Change the Internet ? *Business Communications Review*, September 2002.

[31] G. Box, G. Jenkins, and G. Reinsel. *Time Series Analysis: Forecasting and Control.* Prentice-Hall, 1994.

[32] J. Branke. *Evolutionary Optimization in Dynamic Environments.* Kluwer Academic Publishers, 2002.

[33] A. Broido, E. Nemeth, and K. Claffy. Internet expansion, refinement and churn. *European Transactions on Telecommunications*, January 2002.

[34] A. Bryzon. *Dynamic Optimization.* Prentice-Hall, 1999.

[35] M. Buchanan. *Ubiquity: the science of history...or why the world is simpler than we think.* Phoenix, London, 2001.

[36] L. Burgstahler and M. Neubauer. New Modifications of the Exponential Moving Average Algorithm for Bandwidth Estimation. In *Proc. of the 15$^{th}$ ITC Specialist Seminar*, July 2002. Wüerzburg, Germany.

[37] R. Caceres, N. Duffield, A. Feldmann, J. Friedmann, A. Greenberg, R. Greer, T. Johnson, C. Kalmanek, B. Krishnamurthy, D. Lavelle, P. Mishra, K. Ramakrishnan, J. Rexford, F. True, and J. van der Merwe. Measurement and analysis of IP network usage and behavior. *IEEE Communications Magazine*, May 2000.

[38] CAIDA. cflowd: Traffic Flow Analysis Tool. Available from http://www.caida.org/tools/measurement/cflowd/, 1998.

[39] A. Carlisle. Applying the Particle Swarm Optimizer to Non-stationary Environments. Ph.D. thesis, Auburn University, 2002.

[40] I. Castineyra, N. Chiappa, and M. Steenstrup. The Nimrod Routing Architecture. Internet Engineering Task Force, RFC1992, August 1996.

[41] C. Chatfield. *The Analysis of Time Series: An Introduction, Fifth Edition.* Chapman & Hall, 1996.

[42] E. Chen and S. Sangli. Avoid BGP Best Path Transition from One External to Another. Internet draft, draft-chen-bgp-avoid-transition-00.txt, work in progress, June 2003.

[43] Cisco. BGP Case Studies Section 1. Available at `http://www.cisco.com/warp/public/459/13.html`.

[44] Cisco. Sample Configurations for Load Sharing with BGP in Single and Multihomed Environments. Available at `http://www.cisco.com/warp/public/459/40.html`.

[45] Cisco. NetFlow services and applications. White paper, available from `http://www.cisco.com/warp/public/732/netflow`, 1999.

[46] Cisco. BGP Best Path Selection Algorithm. http://www.cisco.com/warp/public/459/25.shtml, 2002.

[47] K. Claffy, H. Braun, and G. Polyzos. Traffic characteristics of the T1 NSFNET backbone. In *INFOCOM93*, 1993.

[48] K. Claffy, G. Miller, and K. Thompson. The nature of the beast : recent traffic measurements from an Internet backbone. In *INET98*, 1998. available from http://www.caida.org/Papers.

[49] C. A. C. Coello. An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys*, 32(2):109–143, 2000.

[50] N. Z. Computer Science Dept., University of Waikato. The DAG project. Available from `http://dag.cs.waikato.ac.nz/`.

[51] M. Crovella and A. Bestavros. Self-similarity in world wide web traffic evidence and possible causes. In *Proc. of* ACM SIGMETRICS'96, pages 160–169, May 1996.

[52] T. Dang and S. Molnar. On the Effects of Non-stationarity in Long-Range Dependence Tests. *Periodica Polytechnica Ser. El.*, 43(4):227–250, 1999.

[53] M. Daniel and A. Willsky. Modelling and Estimation of Fractional Brownian Motion using Multiresolution Stochastic Processes. *In Lévy Véhel, Lutton and Tricot (editors) "Fractals in Engineering: From Theory to Applications"*, 1997.

[54] I. Daubechies. *Ten Lectures on Wavelets.* Number 61 in CMBS-NSF Series in Applied Mathematics, SIAM, Philadelphia, 1992.

[55] K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[56] K. Deb. *Multi-objective Optimization using Evolutionary Algorithms*. Wiley Interscience series in systems and optimization, 2001.

[57] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In *Proc. of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer-Verlag (LNCS 1917).

[58] S. V. den Bosch, F. Poppe, and G. Petit. Single-Path Traffic Engineering with Explicit Routes in a Differentiated-Services Network. In *IEEE International Conference on Communications*, Helsinki, Finland, June 2001.

[59] L. Deri. nProbe: an Open Source NetFlow probe for Gigabit Networks. In *Proc. of Terena TNC 2003*, May 2003.

[60] A. Doria and E. Davies. Analysis of IDR requirements and History Group B contribution. Internet draft, draft-irtf-routing-history-00.txt, work in progress, February 2002.

[61] A. Doria and E. Davies. Future Domain Routing Requirements Group B contribution. Internet draft, draft-irtf-routing-reqs-groupb-00.txt, work in progress, February 2002.

[62] P. Doukhan, G. Oppenheim, and M. T. (editors). *Theory and Applications of Long-Range Dependence*. Birkhäuser, Boston, 2002.

[63] N. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. Ramakrishnan, and J. van der Merwe. Resource management with hoses: point-to-cloud services for virtual private networks. *IEEE/ACM Transactions on Networking*, 10(5):679–692, 2002.

[64] E. Falkenauer. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2:5–30, 1996.

[65] E. Falkenauer. Applying genetic algorithms to real-world problems. In L. D. Davis, K. De Jong, M. D. Vose, and L. D. Whitley, editors, *Evolutionary Algorithms*, pages 65–88. Springer, New York, 1999.

[66] W. Fang and L. Peterson. Inter-AS traffic patterns and their implications. In *IEEE Global Internet Symposium*, December 1999.

[67] N. Feamster, J. Borkenhagen, and J. Rexford. Controlling the impact of BGP policy changes on IP traffic. Technical report, AT&T Technical memorandum, November 2001.

[68] N. Feamster, J. Borkenhagen, and J. Rexford. Guidelines for interdomain traffic engineering. *ACM SIGCOMM Comput. Commun. Rev.*, 33(5):19–30, 2003.

[69] N. Feamster and J. Rexford. Network-Wide BGP Route Prediction for Traffic Engineering. In *Proc. of ITCOM 2002, Boston, MA*, August 2002.

[70] A. Feldmann. Characteristics of TCP connection arrivals. *In Park and Willinger (editors) "Self-Similar Network Traffic and Performance Evaluation", Wiley-InterScience*, 2000.

[71] A. Feldmann, A. Gilbert, P. Huang, and W. Willinger. Dynamics of IP Traffic: a Study of the Role of Variability and the Impact of Control. In *ACM SIGCOMM'99*, pages 301–313, 1999.

[72] A. Feldmann, A. Gilbert, and W. Willinger. Data Networks as Cascades: Investigating the Multifractal Nature of Internet WAN Traffic. In *ACM SIGCOMM'98*, pages 42–55, 1998.

[73] A. Feldmann, A. Gilbert, W. Willinger, and T. Kurtz. The changing nature of network traffic: scaling phenomena. *ACM Computer Communication Review*, 28(2):5–29, 1998.

[74] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: methodology and experience. In *Proc. of ACM SIGCOMM2000*, September 2000.

[75] C. Fonseca and P. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[76] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional IP routing protocols. *IEEE Communications Magazine*, October 2002.

[77] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proc. of IEEE INFOCOM2000*, March 2000.

[78] C. Fraleigh, C. Diot, B. Lyles, S. Moon, P. Owerzarski, D. Papagiannaki, and F. Tobagi. Design and deployment of a passive monitoring infrastructure. In *Passive and active measurements workshop*, Amsterdam, April 2001.

[79] L. Gao. On inferring autonomous system relationships in the Internet. In *Proc. of IEEE Global Internet Symposium*, November 2000.

[80] L. Gao and F. Wang. The Extent of AS Path Inflation by Routing Policies. In *Proceedings of IEEE Global Internet Symposium*, 2002.

[81] M. Garey and D. Johnson. *Computers and Intractability - A Guide to the Theory of NP-completeness.* W.H. Freeman & Co., San Francisco, USA, 1979.

[82] A. Gilbert, W. Willinger, and A. Feldmann. Scaling Analysis of Conservative Cascades, with Applications to Network Traffic. *IEEE Trans. on Information Theory*, 45(3):971–992, 1999.

[83] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[84] J. Grefenstette. Genesis: A system for using genetic search procedures. In *Proceedings of the 1984 Conference on Intelligent Systems and Machines*, pages 161–165, 1984.

[85] S. D. Gribble. UC Berkeley Home IP HTTP Traces. Available at `http://www.acm.org/sigcomm/ITA/`, July 1997.

[86] T. Griffin and G. Wilfong. An analysis of BGP convergence properties. In *Proc. of ACM SIGCOMM'99*, September 1999.

[87] B. Halabi. *Internet Routing Architectures ($2^{nd}$ edition).* Cisco Press, 2000.

[88] S. Hergarten. *Self-Organized Criticality in Earth Systems.* Springer-Verlag, 2002.

[89] R. Hinden. Internet Routing Protocol Standardization Criteria. Internet Engineering Task Force, RFC1264, October 1991.

[90] J. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.

[91] G. Holzmann. *Design and validation of computer protocols.* Prentice-Hall, 1991.

[92] B. Huffaker, M. Fomenkov, D. Plummer, D. Moore, and K. Claffy. Distance Metrics in the Internet. In *Proc. of IEEE International Telecommunications Symposium (ITS)*, September 2002.

[93] G. Huston. Analyzing the Internet's BGP routing table. *Internet Protocol Journal*, 4(1), 2001.

[94] V. Jacobson, C. Leres, and S. McCanne. tcpdump. Available at ftp://ftp.ee.lbl.gov, 1989.

[95] S. Jaffard. Multifractal formalism for functions Part I : Results valid for all functions. *SIAM J. Math. Anal.*, 28(4):944–970, July 1997.

[96] Juniper. Junos software release 5.6 : New features list. `http://www.juniper.net/products/ip_infrastructure/junos/105012.html`.

[97] A. Karasaridis and D. Hatzinakos. Network Heavy Traffic Modeling using alpha-Stable Self-Similar Processes. *IEEE Transactions on Communications*, 49(7):1203–1214, July 2001.

[98] K. Keys, D. Moore, R. Roga, E. Lagache, M. Tesch, and K. Claffy. The architecture of Coralreef: an Internet traffic monitoring software suite. In *Proc. of* PAM2001, April 2001.

[99] L. Kleinrock and W. Naylor. On measured behavior of the ARPA network. In *AFIS Proceedings, 1974 National Computer Conference*, volume 43, pages 767–780. John Wiley & Sons, 1974.

[100] B. Krishnamurthy, C. Wells, and Y. Zhang. On the use and performance of content distribution networks. In *Proc. of the first ACM SIGCOMM Internet Measurement Workshop*, November 2001.

[101] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener. Algorithms for provisioning virtual private networks in the hose model. *IEEE/ACM Transactions on Networking*, 10(4):565–578, 2002.

[102] C. Kunzinger. Protocol for the Exchange of Inter-Domain Routing Information among Intermediate Systems to Support Forwarding of ISO 8473 PDUs. Internet Draft ISO/IEC 10747, draft-kunzinger-idrp-ISO10747-00.txt, April 1994.

[103] C. Labovitz, G. Malan, and F. Jahanian. Internet Routing Instability. In *Proc. of ACM SIGCOMM'97*, September 1997.

[104] A. Lange. Issues in Revising BGP-4. Internet draft, draft-ietf-idr-bgp-issues-01.txt, work in progress, June 2003.

[105] S. Leinen. Evaluation of candidate protocols for IP flow information export (IPFIX). Internet draft, draft-leinen-ipfix-eval-contrib-02, work in progress, January 2004.

[106] W. Leland, M. Taqqu, and W. Willinger. On the Self-Similar Nature of Ethernet Traffic (Extended Version). *IEEE/ACM transactions on networking*, 2(1):1–15, 1994.

[107] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the Self-similar Nature of Ethernet Traffic (Extended Version). IEEE/ACM *Transactions on Networking*, 1994.

[108] W. Leland and D. Wilson. High time-resolution measurement and analysis of LAN traffic: Implications for LAN interconnection. In *Proc. of* IEEE INFOCOM'91, 1991.

[109] R. Mahajan, D. Wetherall, and T. Anderson. Understanding BGP Misconfigurations. In *Proc. ACM SIGCOMM'02*, September 2002.

[110] B. B. Mandelbrot. Intermittent turbulence in self-similar cascades: divergence of high moments and dimension of the carrier. *Journal of Fluid Mechanics*, 62:331–358, 1974.

[111] B. B. Mandelbrot. *Multifractals and 1/f noise*. Springer-Verlag, 1997.

[112] B. B. Mandelbrot. *Fractals and Scaling in Finance*. Springer-Verlag, 1999.

[113] B. B. Mandelbrot. *Gaussian Self-Affinity and Fractals*. Springer-Verlag, 2001.

[114] Z. M. Mao, J. Rexford, J. Wang, and R. Katz. Towards an accurate AS-level traceroute tool. In *ACM SIGCOMM2003*, 2003.

[115] S. McCreary and K. Claffy. Trends in wide area IP traffic patterns : a view from Ames Internet Exchange. Available from `http://www.caida.org/outreach/papers/AIX0005/`, 2000.

[116] T. McGregor, H.-W. Braun, and J. Brown. The NLANR Network Analysis Infrastructure. *IEEE Communications Magazine*, May 2000.

[117] P. McManus. A passive system for server selection within mirrored resource environments using AS path length heuristics. Available from `http://www.gweep.net/~mcmanus/proximate.pdf`, April 1999.

[118] D. McPherson and K. Patel. Experience with the BGP-4 Protocol. Internet draft, draft-ietf-idr-bgp4-experience-protocol-01.txt, work in progress, August 2003.

[119] D. Meyer. University of Oregon Route Views Project. Available at `http://antc.uoregon.edu/route-views/`.

[120] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 1996.

[121] T. Mikosch, S. Resnick, H. Rootzén, and A. Stegeman. Is Network Traffic Approximated by Stable Lévy Motion or Fractional Brownian Motion? *The Annals of Applied Probability*, pages 23–68, 2002.

[122] T. Mikosch and C. Starica. Long-Range Dependence Effects and ARCH Modeling. *In P. Doukhan and G. Oppenheim and M. Taqqu (editors), "Theory and Applications of Long-Range Dependence", Birkhäuser, Boston*, 2002.

[123] U. of Waikato. The DAG project. `http://dag.cs.waikato.ac.nz/`.

[124] J. Org, M. Ian, and G. Brownlee. The auckland data set: an access link observed. In *Proceedings of the $14^{th}$ ITC Specialist Seminar*, April 2001.

[125] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: algorithms and complexity.* Prentice-Hall, 1982.

[126] V. Pareto. *Cours d'Economie Politique, volume I et II.* F. Rouge, Lausanne, 1896.

[127] K. Park, G. Kim, and M. Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In ICNP'96, 1996.

[128] K. Park, G. Kim, and M. Crovella. On the Effect and Control of Self-Similar Network Traffic. In *Proc. SPIE International Conference on Performance and Control of Network Systems*, pages 296–310, 1997.

[129] K. Park and W. Willinger. *Self-Similar Network Traffic and Performance Evaluation.* Wiley-Interscience, 2000.

[130] V. Paxson. Empirically-Derived Analytic Models of Wide-area TCP Connections. IEEE/ACM *Transactions on Networking*, 2(4):316–336, 1994.

[131] V. Paxson and S. Floyd. Wide-Area Traffic: The Failure of Poisson Modeling. IEEE/ACM *Transactions on Networking*, 3(3):226–244, 1995.

[132] M. Priestley. *Spectral Analysis and Time Series*. Academic Press, 1981.

[133] R. Purshouse and P. Fleming. Conflict, Harmony, and Independence: Relationships in Multi-criterion Optimisation. In *Proc. of the Second International Conference on Multi-Criterion Optimization (EMO2003), Portugal*, pages 16–30, April 2003.

[134] Y. Qiao, J. Skicewicz, and P. Dinda. Multiscale predictability of network traffic. Technical Report NWU-CS-02-13, Nothwestern University, October 2002.

[135] B. Quoitin, S. Tandel, S. Uhlig, and O. Bonaventure. Interdomain Traffic Engineering with redistribution communities. *Computer Communications Journal*, October 2003.

[136] B. Quoitin, S. Uhlig, and O. Bonaventure. Using redistribution communities for Interdomain Traffic Engineering. In *Proc. of the third COST 263 workshop on Quality of future Internet Services, Zurich, Switzerland*, pages 125–134. Springer Verlag, LNCS2511, October 2002.

[137] B. Quoitin, S. Uhlig, C. Pelsser, L. Swinnen, and O. Bonaventure. Interdomain traffic engineering with BGP. *IEEE Communications Magazine, Internet Technologies Series*, May 2003.

[138] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). Internet draft, draft-ietf-idr-bgp4-22.txt, work in progress, October 2003.

[139] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang. BGP Routing Stability of Popular Destinations. In *Proc. of the second ACM SIGCOMM Internet Measurement Workshop*, pages 197–202, November 2002.

[140] R. H. Riedi. An improved multifractal formalism and self-similar measures. *J. Math. Anal. Appl.*, 189:462–490, 1995.

[141] S. Romig, M. Fullmer, and R. Luman. The OSU Flow-tools package and Cisco NetFlow logs. In *Proc. of* USENIX LISA, December 1995.

[142] M. Roughan and D. Veitch. Measuring Long-Range Dependence under Changing Traffic Conditions. In *Proc. of INFOCOM'99*, 1999.

[143] S. Sahni and T. Gonzalez. P-complete approximation problems. *J. ACM*, 23:555–565, 1976.

[144] G. Samorodnitsky and M. Taqqu. *Stable Non-Gaussian Random Processes: Stochastic Models with Infinite Variance*. Chapman & Hall, 1994.

[145] A. Sang and S. Li. A Predictability Analysis of Network Traffic. In *Proc. of IEEE INFOCOM 2000*, 2000.

[146] S. Sangli, D. Tappan, and Y. Rekhter. BGP Extended Communities Attribute. Internet draft, draft-ietf-idr-bgp-ext-communities-06.txt, work in progress, August 2003.

[147] S. Saroiu, P. Gummadi, and S. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proc. of Multimedia Computing and Networking Conference*, January 2002.

[148] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of internet path selection. In *Proceedings of ACM SIGCOMM'99*, 1999.

[149] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. In *Proc. of the second ACM SIGCOMM Internet Measurement Workshop*, November 2002.

[150] G. Siganos and M. Faloutsos. BGP Routing: A Study at Large Time Scale. In *Proc. of IEEE Global Internet Symposium*, November 2002.

[151] P. Smith. Weekly routing table report. Weekly reports from APNIC's router in Japan sent to `bgp-stats@lists.apnic.net`.

[152] D. Sornette. *Critical Phenomena in Natural Sciences*. Springer-Verlag, 2000.

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

[154] N. Srinivas and K. Deb. Multi-Objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation*, 2:221–248, 1995.

[155] M. Steenstrup. Inter-Domain Policy Routing Protocol Specification: Version 1. Internet Engineering Task Force, RFC1479, July 1993.

[156] J. Stewart. *BGP4 : interdomain routing in the Internet*. Addison Wesley, 1999.

[157] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *INFOCOM 2002*, June 2002. Web site with latest data available at `http://www.cs.berkeley.edu/~sagarwal/research/BGP-hierarchy/`.

[158] P. D. Surry and N. J. Radcliffe. Formal algorithms + formal representations = search strategies. In *Parallel Problem Solving from Nature*, pages 366–375, 1996.

[159] H. Tangmunarunkit, R. Govindan, S. Shenker, and D. Estrin. The Impact of Routing Policy on Internet Paths. In *Proceedings of INFOCOM'01*, pages 736–742, 2001.

[160] M. Taqqu. The modeling of Ethernet data and of signals that are heavy-tailed with infinite variance. *Scandinavian Journal of Statistics*, 29(2):273–295, 2002.

[161] M. Taqqu and V. Teverovsky. On Estimating the Intensity of Long-Range Dependence in Finite and Infinite Variance Time Series. *"A Practical Guide to Heavy Tails: Statistical Techniques and Applications"*, 1998.

[162] M. Taqqu, V. Teverovsky, and W. Willinger. Is network traffic self-similar or multifractal? *Fractals*, 5:63–73, 1997.

[163] M. Taqqu, W. Willinger, and R. Sherman. Proof of a Fundamental Result in Self-Similar Traffic Modeling. *ACM Computer Communication Review*, 27, 1997.

[164] A. Tewfik and A. Kim. Correlation structure of the discrete wavelet coefficients of fractional brownian motion. IEEE *Trans. Inform. Theory*, 38:904–909, 1992.

[165] K. Thompson, G. Miller, and R. Wilder. Wide-Area Internet traffic patterns and characteristics. *IEEE Network magazine*, 11(6), November/December 1997.

[166] P. Traina, D. McPherson, and J. Scudder. Autonomous System Confederations for BGP. Internet Engineering Task Force, RFC3065, February 2001.

[167] S. Uhlig. 3D-LD : a Graphical Wavelet-based Method for Analyzing Scaling Processes. Infonet technical report Infonet-2002-04, available at `http://www.infonet.fundp.ac.be/doc/tr/`, April 2002.

[168] S. Uhlig. High-order Scaling and Non-stationarity in Flow Arrivals. *Submitted to ACM Computer Communications Review*, 2002.

[169] S. Uhlig. Interdomain Traffic Engineering: an Evolutionary Approach. Université catholique de Louvain, Research Report RR 2002-10 INFO, December 2002.

[170] S. Uhlig. Conservative cascades: an Invariant of Internet traffic. In *Proc. of the 2003 IEEE International Symposium on Signal Processing and Information Technology, Darmstadt, Germany*, December 2003.

[171] S. Uhlig. An Evolutionary-based Study of Multiple-objectives Interdomain Traffic Engineering. Submitted, 2003.

[172] S. Uhlig and O. Bonaventure. On the cost of using MPLS for interdomain traffic. In J. Crowcroft, J. Roberts, and M. Smirnov, editors, *Proc. of the first COST263 workshop on Quality of future Internet Services*, pages 141–152. Springer Verlag, LNCS1922, September 2000.

[173] S. Uhlig and O. Bonaventure. Understanding the Long-term Self-similarity of Interdomain Traffic. In M. Smirnov, J. Crowcroft, J. Roberts, and F. Boavida, editors, *Proc. of the second COST263 workshop on Quality of future Internet Services*, pages 286–298. Springer Verlag, LNCS2156, September 2001.

[174] S. Uhlig and O. Bonaventure. Implications of Interdomain Traffic Characteristics on Traffic Engineering. *European Transactions on Telecommunications*, January 2002.

[175] S. Uhlig and O. Bonaventure. Towards a more systematic approach for interdomain traffic engineering. Submitted, 2004.

[176] S. Uhlig, O. Bonaventure, and B. Quoitin. Interdomain Traffic Engineering with minimal BGP Configurations. In *Proc. of the* $18^{th}$ *International Teletraffic Congress, Berlin*, September 2003.

[177] S. Uhlig, O. Bonaventure, and C. Rapier. 3D-LD : a Graphical Wavelet-based Method for Analyzing Scaling Processes. In *Proc. of the 15$^{th}$ ITC Specialist Seminar, Würzburg, Germany*, July 2002.

[178] S. Uhlig, V. Magnin, O. Bonaventure, C. Rapier, and L. Deri. Implications of the Topological Properties of Internet Traffic on Traffic Engineering. In *Proc. of the 19$^{th}$ ACM Symposium on Applied Computing, Special Track on Computer Networks, Nicosia, Cyprus*, March 2004.

[179] D. Veitch, P. Abry, P. Flandrin, and P. Chainais. Infinitely divisible cascade analysis of network traffic data. In *Proc. of ICASSP*, 2000.

[180] M. Vetterli and J. Kovacevic. *Wavelets and subband coding.* Prentice-Hall, 1995.

[181] D. Walton, D. Cook, A. Retana, and J. Scudder. Advertisement of multiple paths in BGP. Internet draft, draft-walton-bgp-add-paths-01.txt, work in progress, November 2002.

[182] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. Wu, and L. Zhang. Protecting BGP Routes to Top Level DNS Servers. In *Proc. of 23rd International Conference on Distributed Computing Systems (ICDCS)*, May 2003.

[183] Y. Wang, Z. Wang, and L. Zhang. Internet traffic engineering without full mesh overlaying. In *INFOCOM2001*, April 2001.

[184] Z. Wang. Internet traffic engineering. *Special section of IEEE Network magazine*, March-April 2000.

[185] M. Wickerhauser. *Adapted wavelet analysis: from Theory to Software.* A K Peters, 1994.

[186] W. Willinger, V. Paxson, R. Riedi, and M. Taqqu. Long-Range Dependence and Data Network Traffic. *In P. Doukhan and G. Oppenheim and M. Taqqu (editors), "Theory and Applications of Long-Range Dependence", Birkhäuser, Boston*, 2002.

[187] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson. Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level. *IEEE/ACM Transactions on Networking*, 5(1):71–86, 1997.