



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**TOWARD UNDERSTANDING THE LONGITUDINAL
STABILITY OF AN IP GEOLOCATION DATABASE**

by

Jonathan A. Culbert

March 2020

Thesis Advisor:
Second Reader:

Robert Beverly
Vinnie Monaco

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE March 2020	3. REPORT TYPE AND DATES COVERED Master's thesis	
4. TITLE AND SUBTITLE TOWARD UNDERSTANDING THE LONGITUDINAL STABILITY OF AN IP GEOLOCATION DATABASE			5. FUNDING NUMBERS	
6. AUTHOR(S) Jonathan A. Culbert				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) While many IP Geolocation Database (IPGD) studies exist gauging "accuracy" by comparing to ground truth (and other IPGDs), there is a dearth of studies looking at the stability of IPGD locations in longitudinal context. Here, we build a framework for such a study and apply it to MaxMind's GeoLiteCity (GLC) IPGD. We discover that there are indeed dynamic and anomalous behaviors in IPGD location histories that affect the stability of reported IP Geolocations (IPGs) over time. While only one IPGD is utilized in this study, we believe these findings will provide general insights into the construction and use of IPG data. The framework we developed quantifies dynamic behaviors (cycles, jumps, blinks, etc.) and redefines Paxson's "persistence" and "prevalence" metrics in the context of IPGDs. Applying our framework to the corpus found, for example, that of the 27,672,206 measurable network IP addresses in the corpus, 75% moved (jumped) on average nearly twice a year and 81% subsequently returned (cycled) back to where they started. Further, nearly 20% of the corpus exhibits 3 to 12 of these cycles over the 9.5 years of corpus history. We also demonstrate the utility of our framework to provide contextual insights in IPGDs by evaluating filtered subsets of the data (i.e., by Autonomous System type and by IPG continent). Lastly, we illustrate that dynamic IPGD location histories could have affected prior academic work and identify future work to expand our proposed framework.				
14. SUBJECT TERMS IP geolocation, MaxMind, longitudinal, geolocation, database			15. NUMBER OF PAGES 99	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**TOWARD UNDERSTANDING THE LONGITUDINAL STABILITY
OF AN IP GEOLOCATION DATABASE**

Jonathan A. Culbert
Lieutenant Commander, United States Navy
BS, The Citadel, 2009
BSAST, Thomas A Edison State College, 2009

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

**NAVAL POSTGRADUATE SCHOOL
March 2020**

Approved by: Robert Beverly
Advisor

Vinnie Monaco
Second Reader

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

While many IP Geolocation Database (IPGD) studies exist gauging “accuracy” by comparing to ground truth (and other IPGDs), there is a dearth of studies looking at the stability of IPGD locations in longitudinal context. Here, we build a framework for such a study and apply it to MaxMind’s GeoLiteCity (GLC) IPGD. We discover that there are indeed dynamic and anomalous behaviors in IPGD location histories that affect the stability of reported IP Geolocations (IPGs) over time. While only one IPGD is utilized in this study, we believe these findings will provide general insights into the construction and use of IPG data. The framework we developed quantifies dynamic behaviors (cycles, jumps, blinks, etc.) and redefines Paxson’s “persistence” and “prevalence” metrics in the context of IPGDs. Applying our framework to the corpus found, for example, that of the 27,672,206 measurable network IP addresses in the corpus, 75% moved (jumped) on average nearly twice a year and 81% subsequently returned (cycled) back to where they started. Further, nearly 20% of the corpus exhibits 3 to 12 of these cycles over the 9.5 years of corpus history. We also demonstrate the utility of our framework to provide contextual insights in IPGDs by evaluating filtered subsets of the data (i.e., by Autonomous System type and by IPG continent). Lastly, we illustrate that dynamic IPGD location histories could have affected prior academic work and identify future work to expand our proposed framework.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1	Introduction	1
1.1	Scope	2
1.2	Summary of Contributions	2
1.3	Thesis Organization	3
2	Background and Related Work	5
2.1	IP Geolocation Applications	5
2.2	IP Geolocation Methods	6
2.3	Related Work	8
2.4	MaxMind Databases	11
3	Methodology	15
3.1	Data Collection	15
3.2	Data Preparation	15
3.3	Data Measurements	20
4	Results	27
4.1	Overall Features of the GLC Corpus and Data	27
4.2	Measurements of the Corpus and CIDRs	32
4.3	Applied Measurement Results	44
5	Conclusions and Future Work	51
	Appendix: File Dates and Python Code	55
A.1	Corpus Filedates	55
A.2	Master File Construction Code	58
A.3	Stats Tabulation Code	64
A.4	IPv4 Coverage Tabulation Code	73

List of References	77
Initial Distribution List	81

List of Figures

Figure 2.1	GLC file dates contained in the corpus.	12
Figure 3.1	Line graphs illustrating typical corpus CIDR behaviors.	21
Figure 4.1	The count of <i>maximally sized</i> CIDRs and GLC Blocks over time.	27
Figure 4.2	GLC Location file's locId count to Blocks file's reference count. .	29
Figure 4.3	Counts of file dates and CIDRs per unique LLP.	30
Figure 4.4	IPv4 address space coverage per GLC Blocks file.	31
Figure 4.5	GLC address apace coverage in terms of CIDR prefixes.	32
Figure 4.6	The number of LLPs per CIDR.	34
Figure 4.7	The number of blinks per CIDR.	35
Figure 4.8	The number of unique LLPs per CIDR.	36
Figure 4.9	The number of repeats and dominant LLP repeat value.	37
Figure 4.10	The number of jumps per CIDR.	38
Figure 4.11	The number of cycles per CIDR.	39
Figure 4.12	The wander distance per CIDR.	40
Figure 4.13	The wander distance per LLP (average wander distance).	40
Figure 4.14	The sum total number of persistent days per CIDR.	42
Figure 4.15	CIDR persistence.	42
Figure 4.16	CIDR prevalence.	43
Figure 4.17	Corpus persistence/prevalence scatter plot.	44
Figure 4.18	Analysis by AS type.	46
Figure 4.19	Analysis by continent.	48

Figure 4.20 Eriksson study sample set 1. 49

Figure 4.21 Eriksson study sample set 2. 50

List of Tables

Table 2.1	Typical applications and descriptions of IPG.	5
Table 2.2	Pros and cons of IPG methods.	6
Table 3.1	Extracting CIDRs from IpNum values.	16
Table 3.2	The issue with simply consolidating raw GLC data.	17
Table 3.3	Master file contents based on given raw GLC data	18
Table 3.4	A second example of Master file contents.	18
Table 3.5	Database measurement tallies and metrics.	20
Table 3.6	Measurements scheme example for LLP locations A, B, C, and -. .	23
Table 4.1	Filtering the set of CIDRs in the corpus.	32
Table 4.2	Data filtering of PeeringDB entries for AS type analysis.	45

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AS	Autonomous System
BGP	Border Gateway Protocol
CAIDA	Center for Applied Internet Data Analysis
CDF	Cumulative Distribution Function
CIDR	Classless Inter-Domain Routing Address
DNS	Domain Name System
DoD	Department of Defense
GIP	GeoIPCity
GIP2	GeoIP2City
GLC	GeoLiteCity
GL2C	GeoLite2City
IPG	IP Geolocation
IPGD	IP Geolocation Database
ISP	Internet Service Provider
LLP	latitude/longitude pair
NPS	Naval Postgraduate School
NSP	Network Service Provider
RIB	Routing Information Base
RIR	Regional Internet Registry

RTT	round-trip time
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USG	United States government
USN	U.S. Navy

Acknowledgments

I would like to thank Dr. Robert Beverly and Dr. Vinnie Monaco for their time and guidance.

I would also like to thank Dr. Brian Eriksson for his willingness to share data from previous academic work, enabling an application demonstration of my framework.

Mostly, I wish to thank my wife, Shareen, and children for enduring the long hours and lonely evenings, the missed games, play, and vacations, and the seemingly endless isolation spent to complete this thesis. We're done.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:

Introduction

IP Geolocation (IPG) is the effort to determine the geographic location of an internet IP address. IPG has many uses as there are many organizations interested in determining where an IP address is located in order to do things like enforce laws, mine market data, and other reasons described in Section 2.1. With legal implications and uses, as well as market competition and digital content protection (and by extension revenue stream protection) at stake, reliably accurate IPG is a heavily demanded capability.

Examples of the impact of IPG include:

- “Operation Bayonet” – where Dutch investigators were able to capture data on over 10,000 IP addresses used by some 420,000 users in a European drug market [1].
- “The Silk Road” – where the US FBI was able to track the physical location of servers used by the silk road using Tor [1].
- “The LulzSec Takedown” – when the hacking group responsible for attacks on the CIA and Sony Pictures were collared. The leader of the group was compromised when he did not mask his IP address as he logged into a chatroom [1].

Conversely, there are cases of failed IPG as well, resulting in relentless harassment by authorities of innocent people who’s real estate property has been indicated as the location of an IP address [2]. Some cases include false accusations of crimes like kidnapping as well [2]. In one case, due to MaxMind selecting an arbitrary latitude and longitude for the “center” of the US and then providing this IPG for any query of their database where the only thing known about the IP address was that it was in the US *somewhere*, thousands of unwanted and misguided visitors showed up at a farmhouse in Kansas [3].

One of the primary means of obtaining IPG data is via subscription services to IP Geolocation Database (IPGD) providers that curate and supply IPG data. These services have increased the accuracy of their data bases over time by research and innovation in IPG methods. With respect to IPG academic research, there has been significant prior effort to analyze IPGDs in the context of location accuracy (e.g., [4], [5], [6], [7]). These efforts

include comparing database locations to selected ground truth, comparative analyses across different databases, and comparisons between various IPG methods. However, little prior work has investigated the *longitudinal* behavior of IPGDs, i.e., their evolution and dynamics over time. In this work we seek to develop a framework for evaluating the longitudinal stability of an IPGD and then apply that framework to an IPGD (MaxMind’s *GeoLiteCity* database) to gain insight into how the database location history behaves and how academic studies that utilize the data could be affected.

1.1 Scope

- We do not parse-out the different reasons an IPG contained in an IPGD could change, we only characterize the changes observed (“Movement Correlation” in Chapter 5 discusses this point further).
- Our study is limited to MaxMind’s *GeoLiteCity* (GLC) database.
- The framework is the intended major contribution of this work, secondary are illustrations of how it can be used to derive insights and how prior academic work could be affected by dynamic IPGD location histories.

1.2 Summary of Contributions

- We show that longitudinal location histories are varied and experience unexpected behaviors like sudden and large location shifts (1000’s of km correlating to 10’s to 100’s of degrees of latitude and longitude), repetitive location shifts (oscillating in an apparent pattern), temporary changes in latitude but not longitude (and vice versa), and other surprising behaviors. Specifically, we found that of the 27,672,206 measurable network IP addresses in the corpus, 75% moved (jumped) on average nearly twice a year and 81% subsequently returned (cycled) back to where they started. Further, nearly 20% of the corpus exhibits 3 to 12 of these cycles over the 9.5 years of corpus history.
- We categorize network addresses in the database to measure the behaviors of types of networks (by AS type and by continent) illustrating a capability of the measurement framework.
- We also give an example of a sample of data used in an academic study that exhibited these location shifts on short time scales both before and after data was collected—

illustrating how timing can impact a study's resulting measurements.

1.3 Thesis Organization

- Chapter 2 discuss IPG in detail and outlines some work that has been done using (specifically MaxMind's) IPG databases.
- Chapter 3 explains and establishes the methodology created to capture the longitudinal behavior of an IPGD.
- Chapter 4 provides the results of applying the developed methodology to the corpus of MaxMind's GeoLiteCity Databases compiled for this study.
- Chapter 5 discusses our conclusions from the study and outlines future work.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:

Background and Related Work

This chapter provides background on IPG uses, methods, and academic work.

2.1 IP Geolocation Applications

IPG is the effort to determine the geographic location of an IP address. The reasons one would be interested to do this activity vary from benign to exploitative in applications of every kind (commercial, government, military, academic, etc.). Table 2.1 summarizes IPG applications.

Application	Description
Digital Rights Management	Ensuring only paying subscribers in approved physical locations can view content
Export Compliance	Enacting or enforcing embargoes or export controls
Content Localization	Providing relevant content to local users (like news)
Price Optimization	Setting prices based on user location and adjusting for unique add-ons (i.e., taxes)
Geo Advertising	Pushing content to users aligned with the needs and interests of their geographic location
Traffic Management	Increasing internet access speeds by directing user traffic to most local servers
Traffic Analytics	Understanding the geographic origin of traffic to websites (trend insights)
Fraud Detection	Mismatches in customer addresses and traffic origin IP addresses indicating fraud
Security Protection	Blocking geographic regions from access (firewalls)
Law Enforcement	Evidence collection mechanism

Table 2.1. Typical applications and descriptions of IPG. Adapted from [8].

2.2 IP Geolocation Methods

Following is a brief overview of IPG methods of delay-based measurements and data-based records. Table 2.2 briefly summarizes and compares IPG methods.

Method Type	Pros	Cons
Delay-based	- empirical measurement - interaction with target	- adversarial manipulation [9] - accuracy difficult, slow
Data-based	- simple API query - based on reported data	- data inaccurate/incomplete - unknown historical stability

Table 2.2. Pros and cons of IPG methods.

2.2.1 Delay-based

Delay-based (aka measurement-based) methods actively take ping [10] measurements to the *target* from known locations (*landmarks*) to get round-trip times (RTTs). These RTTs are the total time for a packet to get to and return from the target. These RTT latency measurements are then fed into algorithms to locate the target based on the travel time. Constraint-Based Geolocation (CBG) (topology-aware delay-based IPG) is based on delay measurements but employs a triangulation-like technique from a set of known measurement landmarks. Each landmark is assigned a distance to each other landmark and acquires the delay time. This array of landmarks and distance/delay data points are used to locate the target by finding the centroid of the overlapping circles drawn from each landmark whose radius is the upper-bound of how far away the target could be [11]. The algorithms may refine geolocation beyond constraining the region via travel times and/or hop count, further honing in by including other information (like publicly available location information) or performing constrained optimization [11]. Machine learning techniques [12] have also been employed.

All delay-based techniques are made less accurate by fixed and variable sources of delays (latency) in networks. In general, fixed delays result from sources that have a consistent contribution to network delay. These include the time costs of propagation (time for signal

to travel through medium), processing (time for serialization, packetization, inspection, routing look-ups, etc.), and transmission (time to put actual voltage on the wire or create the effect in the medium to propagate). While these are “fixed” they are not controllable in a ping/traceroute because the path (and therefore the hardware encountered) is not controllable. Neither is the load of traffic on the network at the time of the measurement, which contributes to the variable source of delay—buffer queuing delays. In addition to error created by assumptions about the performance characteristics of the infrastructure and the propagation speed through the mediums encountered along the route, error is created due to the fact that the signal doesn’t travel in a straight line from source to destination. Additionally, routes to and from the target may not be the same (*symmetric routes*) and instead are *asymmetric*. Worse, the paths may contain large diversions (*circuitous routes*) that are much longer and less efficient than the presumed route. Overall, queuing delays are variable while other delay types are assumed fixed.

The bulk of the work in this area has focused on the mechanics of how the geolocations are calculated – scrutinizing the details in efforts to reduce the error bars on the measurements. This effort is necessary as at one point determination of a location within the physical borders of a country was an achievement due to the size of the inaccuracies and uncertainties in the measure. Though, work has reduced the uncertainty for locations inside ranges as small as 35 km [13] and work continues in this area. Wang et al. hybridized the constraint based approach and overlaid harvested publicly available location data from universities and the like and were able to reduce uncertainty in the best cases to a median of 690 meters [14]. Overall average accuracy across measurement-based methods is typically in the 10’s to 100’s of km [9].

2.2.2 Data-based

Instead of an active measurement, IPGDs are a means of obtaining the location of an IP address by referencing a database of previously acquired location information.

These IPGDs are both public and commercial/proprietary. Public sources include Regional Internet Registries (RIRs) (e.g., ARIN [15], APNIC [16]) that are the entities overall responsible for distributing internet addresses within their geographic regions. Their query-able records of distribution and registration comprise databases where IP addresses are con-

nected to location data (countries, cities, postal/metro codes, etc.). Commercial products from *MaxMind* [17], *IP2Location* [18], *W3C Geolocation* [19], etc. make available free and/or subscription data. Subscription versions generally use proprietary methods to increase the “accuracy” of their databases (Section 2.3 includes a discussion of accuracy in IPGDs). Though their subscription data is derived from proprietary means, they probably use *whois* [20] services, DNS LOC records, autonomous system numbers, and/or independently culled, purchased, or polled public information [9]. They most probably also obtain location information via agreements with Internet Service Providers (ISPs) who would have exceptionally accurate location information within their networks. Additionally, Domain Name System (DNS) and *reverse* DNS lookup tools can offer information (resolving an IP address to a domain-name that can then be investigated for location).

2.3 Related Work

As of the time of writing this study, we are unaware of any *longitudinal* studies of the characteristics and behavior of an IPGD. There are several works, however, that have looked at IPGDs, briefly summarized in this section.

2.3.1 Database Inaccuracy

Inherent Imprecision

The reality of inherent imprecision in block-based IPG was described in 2007 [21] concluding that, even in an ideal situation, 50% of the Classless Inter-Domain Routing Addresss (CIDRs) locations are off by 250 km from their individual IP locations. When “locating” a block of IP addresses a distributed group of locations will be returned correlating to subdivisions in the block. This situations forces that a best estimate be made to “locate” the block given the group of location data. Generally the centroid of the region created by the group of locations is used. In this way any attempt to actually measure a “block” location for database recording is inherently imprecise. Even in the case no active measurement is used to determine the IPG (eliminating the uncertainty contribution of the active measurement method), still presented is the problem of selecting one location to assign to a block of IP addresses with differing locations. It was shown in 2008 [22] that the geographic resolution of IPG databases were several times coarser than the confidence

region given by active measurements. In this work Siwpersad, Gueye, and Uhlig used MaxMind’s GeoIPCity (GIP) data and showed that in nearly all cases it is best to use an active measurement for IPG instead of a database for a single IP address. Yet, for blocks of IP addresses, IPG databases have a place and are “good enough” for some practical purposes even with their inherent imprecision problem.

Granularity

Poese et al. [23] questioned IPG database information content and concluded they are *generally* unreliable. Though they did a comparison of several databases, with MaxMind in particular they cited the *locId* (location identifier) to block ratio problem where ≈ 3.5 million blocks (CIDRs) are distributed across only ≈ 0.2 million unique locations (implying IPG databases are coarser grained than their content size would suggest). They show, in fact, this (ostensible) overly fine granularity suggested by their large number of unique blocks causes overall accuracy of the database to decrease as a large number of blocks are forced into a small number of locations, forcing a large number of blocks into locations distant from their ground truth location. This concept of the accuracy loss due to this operation is analogous to the of rounding or converting a float type to an integer type. When compared to Border Gateway Protocol (BGP) advertisements, MaxMind’s CIDRs differ by more than 50% with only approximately one-third of their entries matching. The other two-thirds of MaxMind’s entries having finer (larger prefixes) than BGP tables. They attribute this partially to the fact that database locations are not intrinsically linked to how blocks are used by their owners. Additionally, they noted that despite 244 countries and $\approx 175,000$ cities being represented in the database, the database is heavily biased to the United States of America (USA), which constitutes $\approx 45\%$ of the entries.

2.3.2 Database Accuracy

“**Accuracy**” in IPGD is difficult to measure due to limited ground-truth data and the inherent difficulty in measuring the location of internet infrastructure [24]. Also, accuracy is different at different levels of scale (country, region, county, city, zip code). MaxMind self-advertised their GeoLiteCity (GLC) database as over 79% on the city level (for the US with a 25 mile radius) and their more accurate GIP database at 83% in 2009 [25]. Their country level accuracy was advertised at 99.5% as well. They qualified these claims based

on this methodology:

We arrive at the accuracy figure by checking known IP address and location pairs against our database. We obtain the known IP location pairs from sites that ask the web visitor to provide their geographic location. The geographic location along with the web visitor’s IP Address is forwarded to us. [25]

These claims persisted on their website until mid 2012 (exact date unknown) when it was deprecated for a generalized statement that the free version (*GeoLite*) of the database were simply less accurate than the paid (*GeoIP*) versions.

In 2012, at the country level, Zander used RIR allocation data as ground truth and compared to MaxMind’s *GeoLite* data showing 90% and 94.9% accuracy for allocated blocks/addresses, respectively [5]. In 2017, Gharaibeh et al. (building on the work of Huffaker et al. [26] that developed a method to geolocate routers using location hints that existed in their hostnames), using the Ark-topo-router dataset from CAIDA (2016) [27], showed that IPG databases suffer from an end-point bias [24] (that they have been shown to be far better at locating end-hosts vs routers/infrastructure) [24]. During this study they also were able to show a city-level accuracy in MaxMind of 68%, with at least 11.4% of the misses placed in cities over 40 km away. They further stated that they would not recommend MaxMind’s databases (free or paid) if city level accuracy was important, particularly in the ARIN region. They also state that MaxMind’s paid version (*GeoIP*) is in fact more accurate in all their testing than was the free (*GeoLite*) version. MaxMind fared well in an accuracy study of ground truth locations in central Europe [7] with just over 60% of the test set having a location error of less than 50 km.

Here are a few examples of the works related to the accuracy of MaxMind IPG databases (whether assessing accuracy or using MaxMind database data affected by how accurate the data is) that motivated this study.

MaxMind used to compare against for an accuracy result:

- *A Look at Router Geolocation in Public and Commercial Databases* [24],
- *A Learning-based Approach for IP Geolocation* [12].

MaxMind’s published accuracy evaluated:

- *Geocompare: A Comparison of Public and Commercial Geolocation Databases* [4],
- *On the Accuracy of IP Geolocation Based on IP Allocation Data* [5],
- *A Geolocation Databases Study* [6],
- *Location Accuracy of Commercial IP Address Geolocation Databases* [7].

In summary, when performing studies of IPGD accuracy, the ground truth is a major challenge to evaluating the database. Next is selecting a level of location that is comparable across databases. And last is minimizing the impact of geographic and end-point bias.

Our Work

In a different approach to study IPGDs, we will be interested to define and measure how IPGDs behave over time. How stable are their reported IPGs? How dynamic is the database? Our work will focus on *MaxMind’s GeoLiteCity* (GLC) databases (described in Section 2.4) because of their ubiquity of references and use in this area of internet studies, their consistent high ranking performance amongst their peer services, and—last but most important—it is the database we were able to acquire over the longest time frame facilitating a longitudinal study.

2.4 MaxMind Databases

These sections are monolithic and do contain some information repeated in other sections.

2.4.1 Database Description

MaxMind IPGDs are available in both a free *GeoLite* and paid subscription *GeoIP* types, the main difference being overall accuracy [17] (the reported latitude/longitude location vs the actual geographic location) [28]. The databases are also available in both IPv4 and v6 format and across different levels and entities (Country, City, Autonomous System Numbers (ASNs), Internet Service Providers (ISPs), etc.) **this study considers only IPv4 and furthermore only the *GeoLiteCity* databases.**

Aside a newer web-based service, users can download database files in comma-separated variable (CSV) format. Previously, .dat file formats similar to .mmdb (a newer binary

format) were available but have been deprecated. Also, free for download are coded application-program interfaces (APIs) for developers to incorporate into their websites and programs to perform queries of the database for purposes outlined in Table 2.1.

No matter the format or coding language of the API, the general function is to receive an IP address, perform a lookup, and then return the geolocation information contained in the database for that network. The API uses a “longest-prefix-match” scheme, just as a router would¹, to find the network IP address/CIDR in which the IP address resides in the database. This involves treating the 4 octets of an IP address as one 32-bit long sequence, and then traversing down a binary tree for each bit in succession, matching the bits left to right (most significant bit to least significant bit) until mismatch, then reporting the network information of the last match. In this way no IPv4 address can take longer than 32 operations to determine its network address (and retrieve its data content).

2.4.2 MaxMind’s GeoLiteCity Database

Figure 2.1 below provides a visual representation of the timeline of the data points collected for this study.

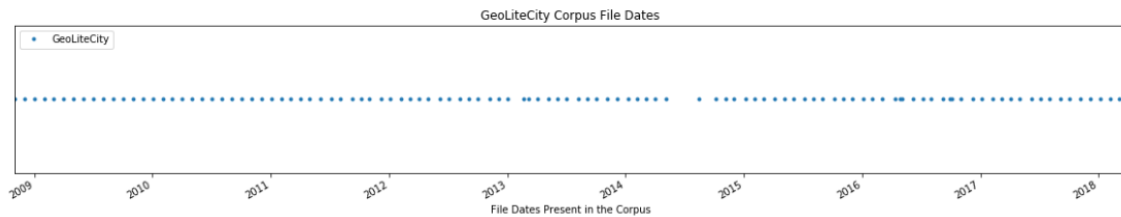


Figure 2.1. GLC file dates contained in the corpus.

We have acquired GLC database information on nearly a monthly basis from November 2008 to March 2018; see Appendix A.1 for specific dates. The GLC database in the CSV

¹Routers use a special kind of memory (ternary content-addressable memory (TCAM)) that allows all the bit comparisons to happen simultaneously, collapsing the matching event down to just one operation.

form comes in a pair of files, the “Blocks” file and the “Location” file. An example of the entries in the database follows (with just one matching pair of entries shown). The data is from the CSV database published on 20171206 (YYYYMMDD):

Example Blocks file content:

```
startIpNum,endIpNum,locId  
"16793600","16799487","872728"
```

Example Location file content:

```
locId,country,region,city,postalCode,latitude,longitude,metroCode,areaCode  
872728,"JP","31","Okayama","700-0827",34.6617,133.9350,,
```

In order to correlate the data, the “locId” (location identifier) from the blocks file is matched to the same locId in the Location file. The location file contains all the available location data while the blocks file only contains the block of IP addresses. The methodology of the data extraction is detailed in Section 3.2.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3: Methodology

This chapter discusses data grooming and the metrics developed to measure the IPG data.

3.1 Data Collection

Historical MaxMind data is not publicly available². Instead, we created our corpus using a combination of cached copies via the Wayback machine [30], general Internet searches, and from research collaborators who had amassed copies of the database over time. While MaxMind has published IPGD information in a number of formats since founding in 2002, the corpus of data collected for this study share the same fields, format, and is generally evenly spaced in time and contiguous. The corpus’ coverage is shown in Figure 2.1. Section 2.4 contains more details on file content.

A note concerning file dates, when files are downloaded they are contained in .zip directories whose .zip file name bare the data collection date (the date the data was collected, not the date of download). We manually transferred this collection date to the Blocks and Location CSV files. We were able to check these file collection dates by comparing them to the CSV file metadata (the “file creation/file modification date(s)”) to ensure they matched.. To additionally assure proper dating, some files were checked against downloads currently available from the Wayback Machine MaxMind website cache to ensure the file date, content, size, and metadata matched.

3.2 Data Preparation

3.2.1 CIDR Extraction

CIDR addresses look like typical IPv4 addresses but end with a slash and a number indicating the number of bits in the *IP network prefix* or simply “prefix”. The larger the prefix number (up to 32 indicating network of only one machine) the smaller the number of hosts in

²Indeed in private email correspondence with MaxMind, they stated that they do not retain historical databases in order to comply with GDPR laws [29].

the network and vice-versa: the smaller the prefix, the larger the number of possible host addresses in the network. Router tables contain CIDR addresses because a given CIDR can compactly represent many destination IP addresses sufficient for routing purposes, minimizing the size of the routing table that must be held in memory and the overhead communications to update it. As discussed in the database entry example in Section 2.4 the CSV version of the database contains "IpNum" (integer representations of an IP address) ranges, or "blocks," in the Blocks file. These must be converted to CIDRs for processing and to make the visualization more meaningful (being presented as typical IP Network addresses vs. large integers). If the difference between the starting and stopping IpNum is equal to a power of 2, then only one CIDR will result. If the range doesn't equal a power of two, then more than one CIDR will exist in a block. Table 3.1 illustrates finding the smallest set of the largest CIDRs (smallest prefix number) network addresses.

Block IpNums	IP Address	CIDRs
start: 16793600	1.0.64.0	1.0.64.0/20, 1.0.80.0/22, 1.0.84.0/23, 1.0.86.0/24
end: 16799487	1.0.86.255	

Table 3.1. Extracting CIDRs from IpNum values [17].

We used Python's *ipaddress* [31] package to convert start and stop IpNums to IP addresses. We then used the *netaddr* [32] package to convert those IP addresses to CIDRs. The following equations illustrate the general concept of conversions.

$$\text{IP Address} = w.x.y.z$$

$$\text{IpNum} = (w * 256^3) + (x * 256^2) + (y * 256^1) + (z)$$

$$w = \text{int}(\frac{\text{IpNum}}{256^3}) \% 256$$

$$x = \text{int}(\frac{\text{IpNum}}{256^2}) \% 256$$

$$y = \text{int}(\frac{\text{IpNum}}{256^1}) \% 256$$

$$z = \text{int}(\text{IpNum}) \% 256$$

3.2.2 Concept of Organization

The over-arching concept of data preparation was to create a body of data (a Master File) that contained the location data (specifically the latitude/longitude pair) one would have acquired if they had queried each published GLC database for all possible CIDRs contained in the data corpus. Ideally this file would contain the location history of every CIDR the product line had ever included as it would have been reported over time. This specificity is important because we are characterizing the behavior of the database as it would be *used* vs. simply consolidating the data contained in each file. Consolidating the data would have failed to capture the dynamic nature of the IPGD locations as networks/blocks were subdivided, consolidated, or moved over time.

For example, given dates d1, d2, and d3 (consecutive months), locations A, B, and C (unique locations), and CIDRs 1, 2, and 2a (2a is a subnet of 2). Dashes “-” indicate a CIDR not present in the published Blocks file on that date.

	d1	d2	d3
1	A	A	A
2	-	B	B
2a	-	-	C

Table 3.2. The issue with simply consolidating raw GLC data.

- On d3, all three CIDRs are present with location data populated so on that date the Master File should contain locations as listed for the 3 CIDRs.
- On d1, if CIDRs 2 and 2a were queried both 2 and 2a should return nothing (a “None” from the API), so the Master File should likewise contain this same result
- On d2, though the raw data has no location for CIDR 2a, its location in the Master File should be recorded as **B**, having longest-prefix-matched to CIDR 2.

The Master File should then look like this:

	d1	d2	d3
1	A	A	A
2	-	B	B
2a	-	B	C

Table 3.3. Master file contents based on given raw GLC data

In the case where CIDR 2 was a subnet of 1 (and 2a of 2), the Master File should be:

	d1	d2	d3
1	A	A	A
2	A	B	B
2a	A	B	C

Table 3.4. A second example of Master file contents.

Table 3.4 now shows a more complete history of networks 2 and 2a as would have been reported from queries through the API for d1, d2, and d3. **Networks appearing to have moved location may or may not be true.** The ability to discriminate actual vs. apparent movement (physically or otherwise) of the hardware hosting a particular IP address or Network is beyond the scope of this study and is mentioned in the Future Work section. However, this situation of apparent movement is a major part of this study because clearly it affects the locations reported by the database queries that are subsequently used for purposes shown in Table 2.1 and academic work discussed in 2.3.

3.2.3 Master File Construction

Now with a method to extract CIDRs from IpNum blocks and with the concept of organization defined, the master file containing all corpus data can be constructed. In order to simplify the longest-prefix-matching operation required to properly construct the master file, binary radix trees (“tries”) are formed for each dated pair of Blocks and Location files. The node labels are the CIDRs generated from the blocks file and the leaves contain the latitude/longitude pair (LLP) present in the Location file for the matching locId.

The master file (a text file) is constructed of lines of data in this format:

```
cidr*[(lat_1,long_1), (lat_2,long_2),... (lat_113,long_113)]
```

each LLP in chronological order matching the 113 file dates in the corpus.

Each line is tabulated via the following steps:

1. The union set of all CIDRs is determined from all files in the corpus.
2. Each CIDR is looked up in each trie and the returned LLP is added to the line.³
3. Once completed, the line is written to the master text file.

In practice, the concept of the above steps were accomplished trie by trie (column file date) vs CIDR by CIDR (row) because loading all tries to memory simultaneously was impractical. Appendix A.2 contains the code used to perform this operation.

Author's Note:

It is well acknowledged by the author that a typical database like *MySQL* would have likely been a better format option than a text file for the nearly 72 GB master file. Early on in the analysis the data was conducive to the less sophisticated approach taken. Over time as the data grew in size and complexity, sufficient adaptations were made to support the austere system employed. By the time the ideas of switching to a typical database schema resurfaced, a working system had already been developed and used so converting it was more work than it was worth and was therefore eschewed.

3.2.4 Master File Support Files

To accommodate swiftly indexing to data in the large master file, a “lookup” file was made that was simply a JavaScript Object Notation (JSON) dump of a python dictionary whose keys were CIDRs and whose values were the byte offset into the master file at which the line of data for that CIDR occupied. In this way one could load the lookup JSON as a dictionary, perform a dictionary lookup the offset to any of the 27.6+ million unique CIDRs, and then index directly into the master file at that offset to retrieve the entire history of that CIDR.

³If the query returns a “None” (meaning there is no matching prefix in the trie) a dash ‘-’ is entered in place of a LLP to maintain the date/list index relationship.

This same master data file and lookup file scheme was used to form a statistics CSV containing data described below in the Data Measurements section.

3.3 Data Measurements

In 1997 Vern Paxson established two measures of stability in network routes using the terms *prevalence* and *persistence* [33]. We will use those same ideas and define them within IPGDs. Paxson’s work established route r as a unique route returned from `traceroute` and a "measurement" to be an instance of a `traceroute` execution that met data quality criteria. Our route r equivalent is a geographic location defined by a LLP and our “measurement” instance is simply the presence of a LLP for a (extracted) CIDR in the GLC database.

Our method is very straightforward, we will crawl over the chronologically ordered LLPs per CIDR and tally interesting events. Those tallies will be used to calculate persistence and prevalence (among other measures). Table 3.5 below summarizes the metrics recorded.

Tally	Description
num_data_points	the number of LLPs for a given CIDR
num_blink	the number of disappearances from the database in the history
num_unique	the number of unique LLPs found
num_repeat	the number of times a given LLP is repeated
num_jumps	the number of times an LLP does not equal the previous LLP
num_cycles	counted once per repeat when a jump is also a repeated LLP
wander_dist	total Haversine distance traversed (in Km) by LLPs
pers_days	cumulative total of days of LLPs that are part of a streak
max_repeat	max number of repeats for any LLP

Table 3.5. Database measurement tallies and metrics.

- streak - when an LLP exactly matches the previous LLP (has not moved). Each time this happens, the number of days between the LLP instances is added to `pers_days`
- A repeat is tallied when an LLP was seen *anytime* in the past, while a streak tallied only when an LLP exactly equals the most recently observed LLP
- `num_unique` and `num_repeat` are complementary
- jump instances and streak instances are complementary

Figure 3.1 contains a sample of CIDRs from our corpus.

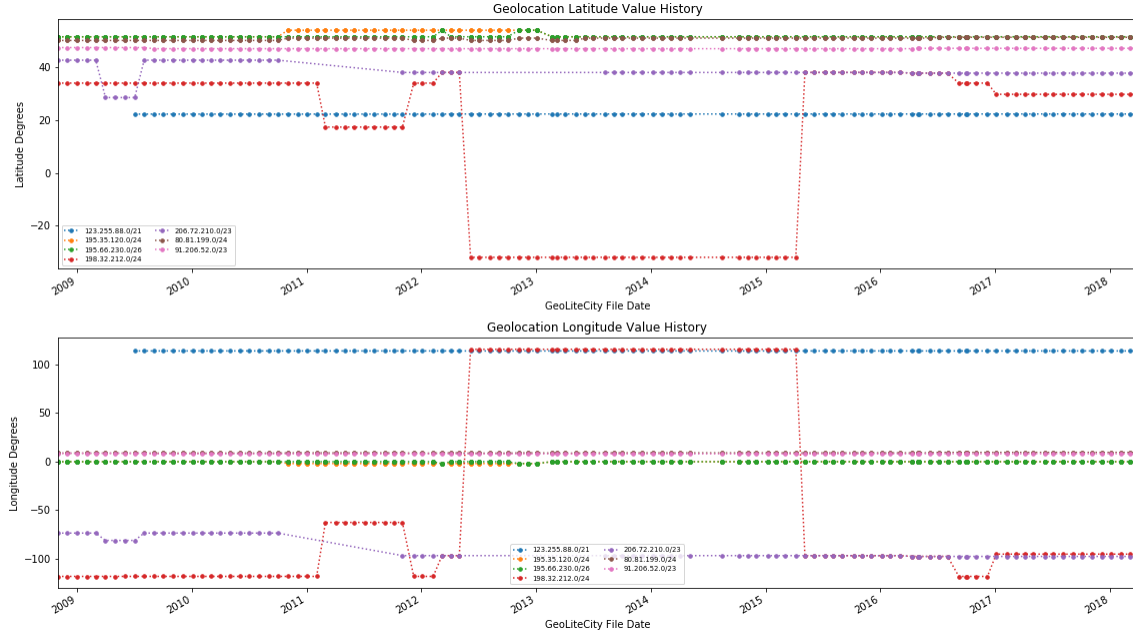


Figure 3.1. Line graphs illustrating typical corpus CIDR behaviors.

Note these features:

- red - jumps and cycle (early and late 2011), repeats, streaks
- purple - blink (mid 2012 to mid 2013)

The graphs above are visually striking and illustrate the dynamic history of these CIDRs. We are quantifying these longitudinal behaviors. There are several interesting features like wavelets, small lat/long changes, large lat/long changes, mirrored shifts across a CIDR's LLP, and more. These become more intriguing when it is realized that these CIDRs also just happen belong to Autonomous Systems (ASs) registered as type "Educational/Research".

3.3.1 Time in the Database

Figure 2.1 gives a visual representation of how the database file dates are distributed throughout time. With some exceptions over the 9.5 year period, file dates are generally monthly, defined by a particular day. The number of days between file dates is determined

and used in the *persistence* measurement described below. A single day is the level of granularity of time selected because of the ease in which dates can be corroborated in the records. Days are the lowest level of granularity that (for the most part) do not have to consider timezone issues or the delay that may occur between the timestamp related to the file itself and when it was available for download from the website (the earliest event possible where the GLC database could be considered “updated”).

3.3.2 Prevalence

Paxson defined *prevalence* as:

Prevalence: the unconditional probability of particular route [33].

Given that we observe route r at the present, how likely are we to observe r again in the future? This has implications for overall network predictability, and the ability to learn from past observations [33].

Paxson selected the “dominant route”; that is, the route that appeared most often, we will do the same. Mapped to our data set, we translate this to:

$$prevalence = \frac{max_repeat}{num_data_points - 1} \quad (3.1)$$

The -1 accounts for the very first data point location because the definition of a repeat is tallied only if it has been seen before. **This measure shows, for the given CIDR, what fraction of the set of all LLPs are the repeat count for the dominant LLP (how prevalent).**

3.3.3 Persistence

Paxson defined *Persistence* as:

Persistence: the likelihood that a route remains unchanged over time [33].

Given that we observe route r at time t , how long before the route is likely to have changed? This has implications for how to effectively manage router state, and for network studies based on repeated path measurements [33].

We measure persistence as follows:

$$persistence = \frac{pers_days}{total_corpus_days} \quad (3.2)$$

This measure shows, for a given CIDR, what fraction of the set of all days in the corpus are members of a *any* streak (have persisted). Here, pers_days takes all LLP streak instances seen for a given CIDR and sums the number of days connected to those streak instances. The LLP with the largest pers_days value is recorded as the most persistent.

3.3.4 Measurement Example

As an example of the overall scheme, consider the length 12 data point (LLP) sequence [-,A,A,B,A,C,A,B,B,-,-,B] where A,B, and C are unique LLPs and “-” means there is no matching network of any prefix length in the trie for that date (so the look-up returned a “None”). Also, for easy math, assume the number of days between each location instance are all uniformly 30 days.

Locations	-	A	A	B	A	C	A	B	B	B	-	-	B	Total
num_data_points		+	+	+	+	+	+	+	+	+			+	10
num_blink											+			1
num_unique		+		+		+								3
num_repeat			+		+		+	+	+	+			+	7
num_jumps				+	+	+	+	+						5
num_cycles					+		+	+						3
A_repeat			+		+		+							3
B_repeat								+	+	+			+	4
C_repeat														0
max_repeat														4
A_streak			+											1
B_streak									+	+			+	3
C_streak														0
sum_streak														4

Table 3.6. Measurements scheme example for LLP locations A, B, C, and -.

Applying Equation 3.1 for prevalence and Equation 3.2 for persistence we obtain:

$$prevalence = \frac{max_repeat(\mathbf{B})}{num_data_points-1} = \frac{4}{10-1} = \frac{4}{9} = 0.444,$$

$$persistence = \frac{(sum_streak)*(pers_days)}{total_corpus_days} = \frac{4*30}{12*30} = \frac{120}{360} = 0.333.$$

Location B is the most prevalent (repeated) accounting for $\frac{4}{9}$'s of the total LLPs beyond the first valid LLP. The location history for this CIDR is only somewhat persistent; throughout its history its location matched its last reported location $\frac{4}{9}$'s of the time (jumping $\frac{5}{9}$'s of the time). Location B is also the most persistent individual LLP. *In the case where the number of days between each measurement is not uniform*, it would be possible for the total pers_days at Location A (which had only 1 LLP that was part of a streak) to be greater than the sum of those of Location B, making Location A the most persistent.

3.3.5 Measurement Output

After processing as described above and as shown in python code in Appendix A.3, a CSV is generated with the following fields:

```
cidr,num_data_points,num_blink,num_unique,num_repeat,num_jumps,num_cycles,
wander_dist,pers_days,persistence,most_pers,pers_lat,pers_long,max_repeat,
prevalence,prevl_lat,prevl_long,loc_bool
```

where:

most_pers - is the persistence value of the LLP with the largest individual pers_days
pers_lat - is the latitude of the LLP with the largest pers_days value,
pers_long - is the longitude of the LLP with the largest pers_days value,
prevl_lat - is the latitude of the LLP with the largest repeat value,
prevl_long - is the longitude of the LLP with the largest repeat value,
loc_bool - True if most prevalent LLP == most persistent LLP.

3.3.6 GLC Coverage

Two other questions that will be answered are “How much of the address space does the GLC database cover?” and “Do any of the blocks listed in the blocks file overlap?”

To answer these, the following scheme was used (python code in Appendix A.4):

1. All Blocks in a single blocks file were converted into CIDRs and loaded into a radix trie with a data element of character 0.
2. All 2^{32} IPv4 IP addresses were looked up in the trie.
 - Successful lookup returns were checked for their data elements. 0 returns were overwritten by 1's, 1 returns tallied an overlap.
 - Unsuccessful look-ups ("None" returns) were tallied for space coverage.
3. Tallies reported IPv4 address space coverage (and any overlaps), and the next Blocks file was processed.

Note: While the entire 2^{32} IPv4 address space was tested, not all possible addresses necessarily need to be (or can be) located by IPG. This is because some IPv4 address are reserved or assigned for various purposes. As result, the total possible "assignable" or "unicast" IPv4 addresses is 3,720,249,092 of the 4,294,967,296 possible ($\approx 88.6\%$) [34].

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:

Results

4.1 Overall Features of the GLC Corpus and Data

Figure 4.1 shows how the entries in the blocks files are “packed.”

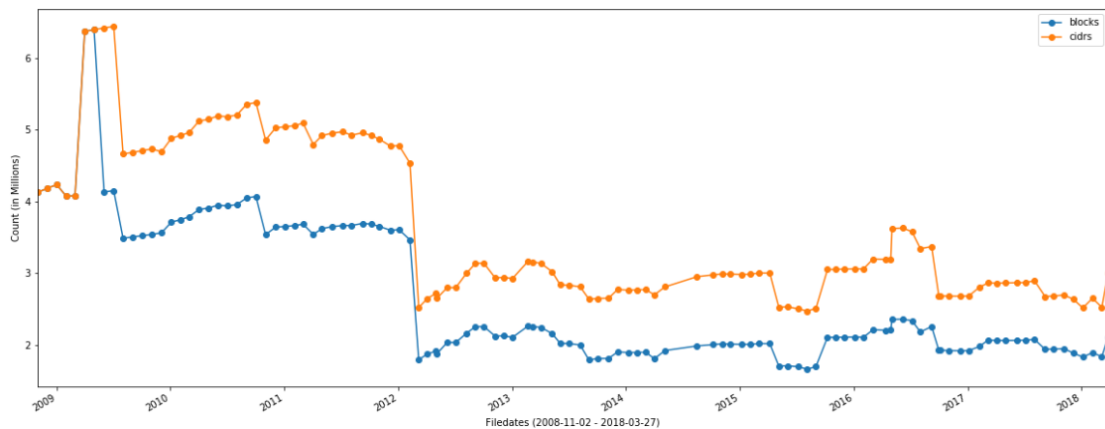


Figure 4.1. The count of *maximally sized* CIDRs and GLC Blocks over time.

That is, for the range of IP addresses covered by the `startIpNum` and `endIpNum` given in an entry in the blocks file (a “block”), how many *maximally sized* CIDRs can be derived? (This is not different than the process described in Section 3.2.1). Maximally sized means that the prefix value is minimized i.e., the CIDRs are as large as possible. If this was not done, then arbitrarily small CIDRs would be numerous (you could have as many /32 CIDRs as IpNums in the IpNum range). Taking the maximally sized CIDRs standardizes the process, minimizes the data, and is more likely to represent what would be found in routing tables. Early on, all the IpNum ranges were set exactly on CIDR (power of 2) boundaries, so the number of IpNum ranges equaled the number of maximally sized CIDRs. Moving

forward in time the one-to-one ratio changes (mid 2009). Continuing forward, despite significant changes (generally reductions) in the number of blocks in the blocks file, the number of maximally sized CIDRs per block is a relatively constant fraction greater than the total number of blocks. Interesting features include the one-to-one ratio early in the data set, the large abrupt changes in numbers of blocks, and (excluding the large jumps) the generally flat trend. See Appendix A.1 for a list of file dates. We contacted MaxMind for comment on the drop in 2012 (that correlated with a large LLP shift for most CIDRs), they declined to comment [35] citing the European Union’s General Data Protection Regulation (GDPR) [29].

Note: the concept of a “location”, LLP(Latitude/Longitude Pair), and “locId” appear to be used interchangeably in the text— but this is ostensible, there is an an important difference. Each locId entry in a single Location file has a unique LLP associated with it that defines a unique location, but that pairing of a locId and LLP is not constant over time. Ergo, the same locId doesn’t always associate with the same location specified by an LLP in a different Location file. locId’s are essentially just an index number assigned to each location that appears in a Location file, which vary in number and order. locIds are mentioned when the count of geographic locations in the Locations file is being considered.

Note: the mention of a “unique” locId (within the context of a single Location file) is equivalent with a unique geographic location and a unique LLP because there are no instances of a locId being issued for a an LLP that is equal to any other LLP in the Location file. While “unique” locIds only had context in a single Location file, the LLPs could be compared across Location files to compare geographic locations. The latitude/longitude float values in LLPs in the database were not converted, they were compared as strings to determine LLP uniqueness. Moreover, once the lat/long values of a LLP were determined, they were not changed. Once MaxMind assigned values to a LLP constituting a location that then was assigned a unique locId, it persisted in their database and showed up in later published Location files. While the locId for an LLP may change over time as it is just an index number, the location defined by the LLP float values (the raw string of characters) does not change. However, new LLPs with new latitude/longitude values are added to the Location files over time.

Figure 4.2 shows the content and utilization of the Location file’s locIds in the corpus.

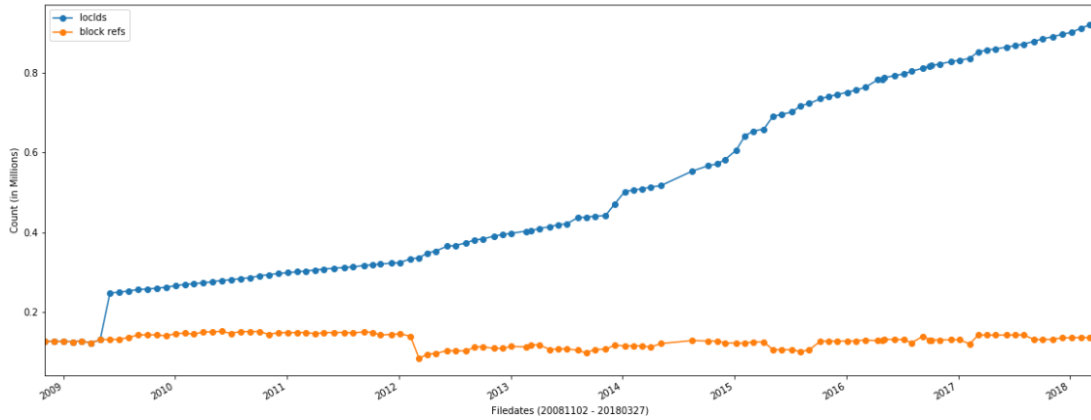


Figure 4.2. GLC Location file's locId count to Blocks file's reference count.

Shown are the count of entries (unique locId's present in the file versus how many of them are present (and therefore referenced) in the Blocks file. Again, early on there is parity, so there are no “extra” locations but that changes quickly (coinciding with major shifts in Figure 4.1). Going forward in time, the number of locations increases but the number of referenced locId's is relatively unchanging. Note that the count of the references includes repeats, so if a locId is referenced 12 times throughout a Blocks file (12 blocks map to the same locId and therefore have the same “location” or LLP), it is counted 12 times in the number of references.

Spot checking the number of locId's in *GeoIPCity* files for the same file dates indicates that the method to generate *GeoLiteCity* Blocks and Location file pairs was to use the *GeoIPCity* Location file (publishing it as is) but then down-sample the blocks file. This may have been accomplished by either by eliminating small blocks or consolidating smaller blocks into larger blocks, to reduce the location granularity offered in the *GeoLiteCity* product. Figure 4.4 will help to determine which case is more likely.

Figure 4.3's Cumulative Distribution Function (CDF) graphs provide insight on unique LLP behaviors.

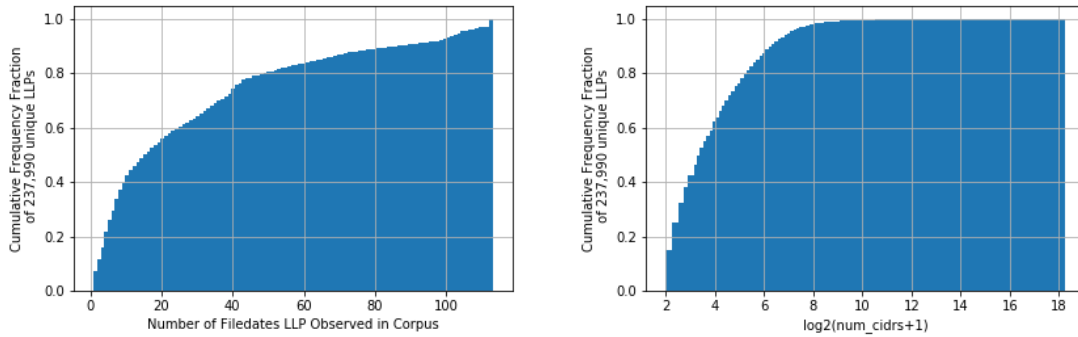


Figure 4.3. Counts of file dates and CIDRs per unique LLP.

Figure 4.3's left CDF gives an idea of how often members of the set of all LLPs are referenced. This graph excludes LLPs that are not referenced by the blocks files (which is the majority of the set, as shown in Figure 4.2 above). Here, 80% of the set of referenced LLPs appear in 48 of the 113 possible files/file dates, and roughly 50% of the LLPs appearing in 18 files/file dates. Only a few percent of the LLPs appear in all the files.

Figure 4.3's right CDF illustrates, in the set of all LLPs of the corpus, the count of CIDRs (*num_cidrs*) attached to each LLP. The $\log_2(\text{num_cidrs} + 1)$ is shown. The lowest *num_cidrs* seen in the corpus was three per LLP, explaining the start at two. This CDF is analogous to how many pigeons are in in each pigeonhole and give sense of how CIDRs are distributed among LLPs. 60% of the corpus LLPs are home to no more than 16 CIDRs while roughly 80% of the LLPs geolocate less than 2^5 (32) CIDRs, and about 40% have only about six CIDRs.

Over time, one would expect that granularity in the database would increase, meaning more *locId*'s and more (and smaller range) blocks. Figure 4.2 shows *locId*'s exhibiting this trend, but 4.1 shows that blocks do not. To gain insight into what is causing this, consider the fraction of the IPv4 address space covered by the GLC database in Figure 4.4. (See Section 3.3.6 for the algorithm description).

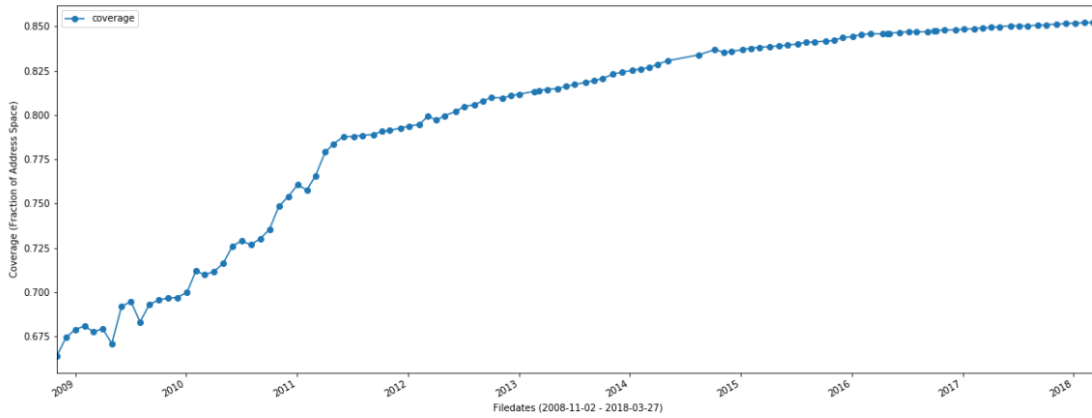


Figure 4.4. IPv4 address space coverage per GLC Blocks file.

IP address space coverage fraction (by file date) ranges from 0.664 to 0.852 with a generally increasing trend with the bulk of coverage space gain in the first third of the file date window, matching the intuition of a slowly improving database coverage. However, the increasing trend could certainly be influenced by RIR allocation of new address space over time. We do not know the precise cause of the trend. Coverage of the set of all CIDRs in the corpus (not shown) was ≈ 0.864 , very close to the ≈ 0.886 determined in Section 3.3.6.

With this information it seems likely that the GIP to GLC conversion involved consolidating vice eliminating blocks in order to reduce database granularity and reduce its reported accuracy. Of note, during this tabulation, there were no instances where any individual IpNum was recorded in more than one location for any file date (in other words there were no overlaps of IP address blocks in any single Blocks file).

Figure 4.5 illustrates prefixes and prefix space representation of CIDRS in the corpus.

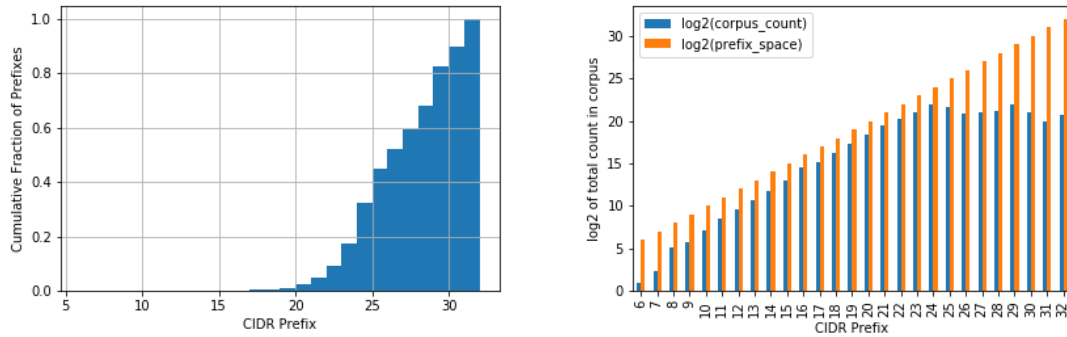


Figure 4.5. GLC address space coverage in terms of CIDR prefixes.

The left graph is a CDF of prefix distribution in the corpus. The right graph shows how many of the total possible number of prefixes are present in the corpus by the prefix value. Here, for example, of CIDRs having prefix values of 6 (/6 networks) and having the prefix space of 2^6 (64), only 2 (2^1) /6 networks (or /6 CIDRs) are present in the corpus. Space coverage increases as the gap between the two bars shrinks until a drop off after /24 (Class C) prefixes. While the poorer performance in networks smaller than /24 may be in part due simply to their greater number, that same line of reasoning would make coverage of the largest networks (with their small population) very good – and it is not. Explanation of this result is future work.

4.2 Measurements of the Corpus and CIDRs

This section describes the results of the measurements applied to the corpus.

4.2.1 Corpus Data Filtering

Table 4.1 enumerates the filtering of corpus CIDRs.

Description	Filtered	Remaining
Count of all unique CIDRs in corpus		27,679,413
Persistence = -1 (no valid LLPs)	5,984	27,673,465
Prevalence = -1 (only one valid LLP)	1,259	27,672,206

Table 4.1. Filtering the set of CIDRs in the corpus.

The “no valid LLPs” condition happened when the LLPs present in the history of a CIDR had invalid latitude and/or longitude values. All occurrences of this type had only one LLP. These were essentially malformed entries with improper latitude and/or longitude values (e.g., latitudes above 90.0). The “only one valid LLP” condition happened when a CIDR only had one LLP in its history and the latitude and longitude values were proper. These CIDs had to be filtered out as only one LLP does not allow for persistence or prevalence measurements. Both of these conditions were handled by setting persistence and prevalence to -1 (and then filtering out those entries). Python code for this adjudication is in Appendix A.3.

Note also that, as CIDs with one invalid or only one valid LLP were filtered out, the minimum number of data points (valid LLPs) for any CID is 2, which allows for a persistence and prevalence measurement. However, in the case where those two LLPs are not the same location, that means there are no repeat or streak instances, which results in a prevalence and persistence measured value of zero.

4.2.2 Corpus Measurements

This section presents the results of the measurements in tabular and graphic form along with discussion. For graphing purposes the $\log_2$ function was used necessitating that values have 1 added to them to ensure no attempts to take the log of zero occurred. The following graphs are CDFs (with some having overlaid histograms not to vertical scale) of measurement results. The scatter plots have pixel-sized data points.

Number of Data Points

`num_data_points` is the number of LLPs in the location history of a specific CID.

Measurement	units	mean	std	min	25%	50%	75%	max
<code>num_data_points</code>	count	107.81	13.44	2	113	113	113	113

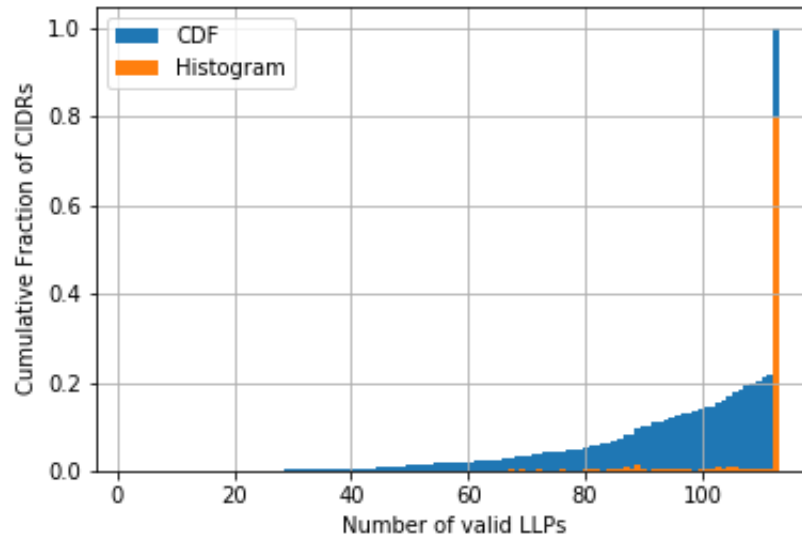


Figure 4.6. The number of LLPs per CIDR.

Data Observation: Most CIDs have the maximum number of valid LLPs.

Number of Blinks

`num_blink` is the number of occurrences where a CIDR present in the database fails to appear in a blocks file, and then does appear in a later blocks file. Essentially the CIDR disappears and subsequently reappears in the database.

Measurement	units	mean	std	min	25%	50%	75%	max
<code>num_blink</code>	count	0.08	0.32	0	0	0	0	13

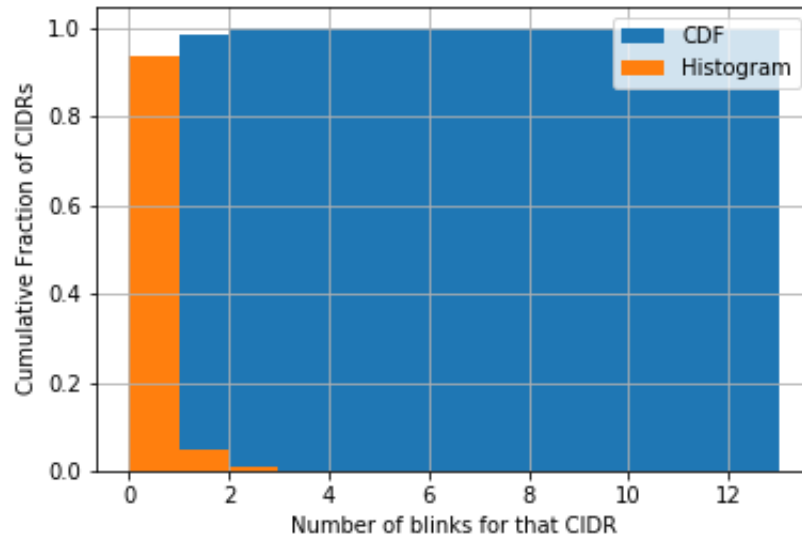


Figure 4.7. The number of blinks per CIDR.

Data Observation: Blinks are uncommon occurrences affecting less than five percent of the corpus CIDRs. Of those affected, approximately 80% only have one blink.

Number of Unique LLPs

`num_unique` is the number of unique locations (LLPs) that exist in a CIDR's location history. Alternatively, it is the size of the set of LLPs for a CIDR's location history.

Measurement	units	mean	std	min	25%	50%	75%	max
num_unique	count	10.04	5.25	1	6	9	14	37

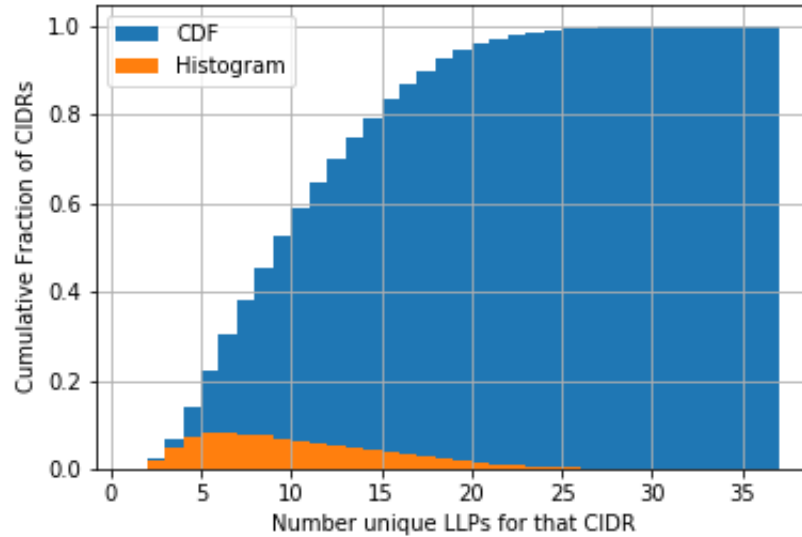


Figure 4.8. The number of unique LLPs per CDR.

Data Observation: The average CDR has about one new unique LLP for each year or location history in the database.

Number of Repeats and Maximum Repeat Value

`num_repeat` is the number of times any LLP is repeated in a CDR's location history after it is first seen (any time after it is first seen, not necessarily consecutively). This measure is related to `num_unique` because if a new LLP is not unique, it is a repeat. Of the set of repeated LLPs, the dominant LLP is the one repeated the most number of times (has the largest `max_repeat_val`).

Measurement	units	mean	std	min	25%	50%	75%	max
<code>num_repeat</code>	count	97.77	13.47	0	95	102	106	112
<code>max_repeat_val</code>	count	40.71	20.18	0	26	36	52	112

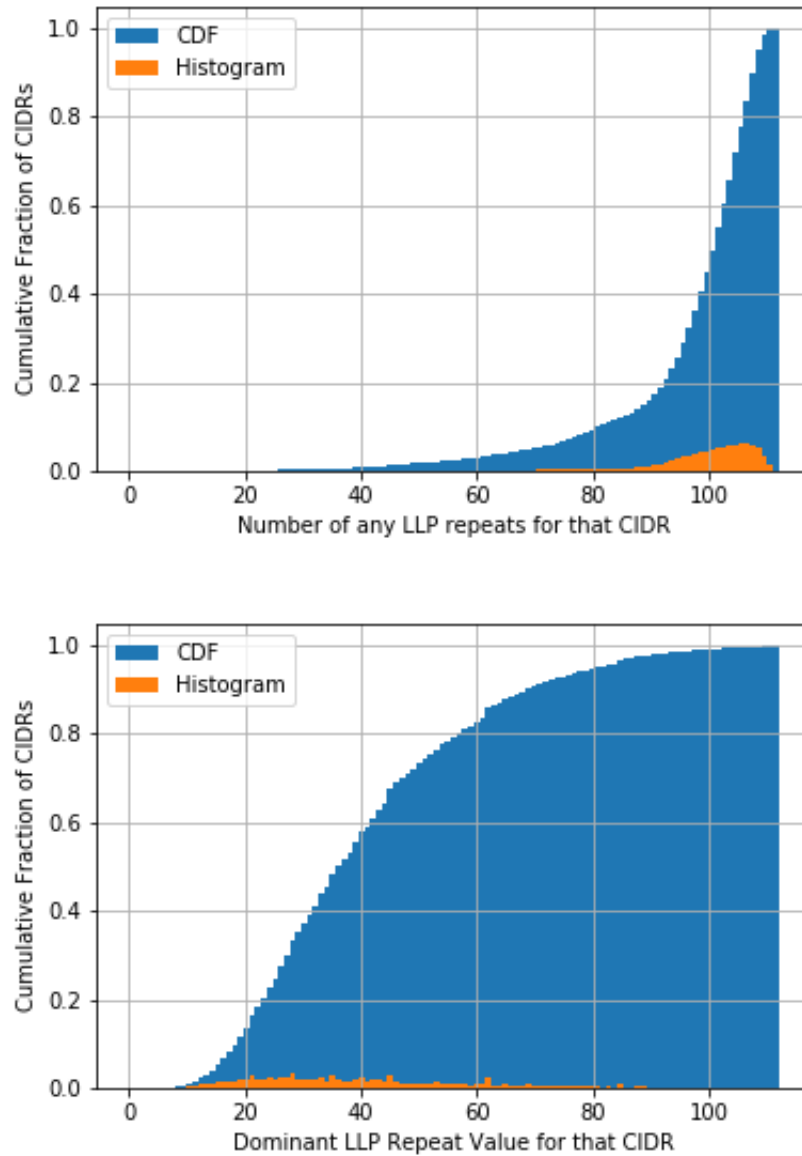


Figure 4.9. The number of repeats and dominant LLP repeat value.

Data Observation: Half of the CIDRs in the Corpus have an LLP that is repeated at least 36 times (representing almost $\frac{1}{3}$ (32.14%) of the opportunities to repeat in the corpus's time window).

Number of Jumps

`num_jumps` is the number of occurrences where an LLP is not equal to the previous LLP (i.e., the location of the CIDR has moved). If the two LLPs are equal, then a repeat has occurred as well as a count for a streak. The streak counts are needed to calculate persistence. High jump counts indicate dynamic histories.

Measurement	units	mean	std	min	25%	50%	75%	max
num_jumps	count	13.18	6.85	0	8	13	18	46

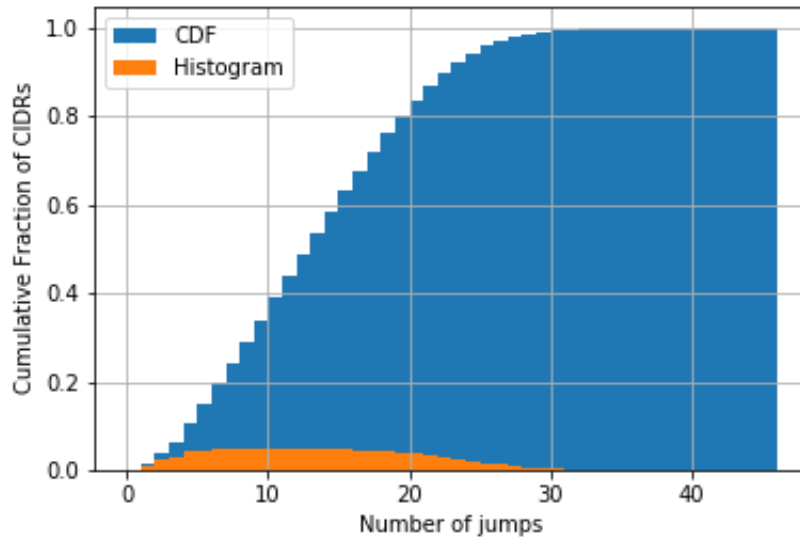


Figure 4.10. The number of jumps per CIDR.

Data Observation: Most CIDRs (75%) move about twice a year (18 jumps over 9.5 years).

Number of Cycles

`num_cycles` is the number of occurrences when a jump is followed by a repeat. To qualify, the repeat must be the same LLP that preceded the jump. The repeat can occur anytime in the future after the jump. This measure captures the interesting behavior of leaving a location only to return at a later date.

Measurement	units	mean	std	min	25%	50%	75%	max
num_cycles	count	2.12	1.60	0	1	2	3	12

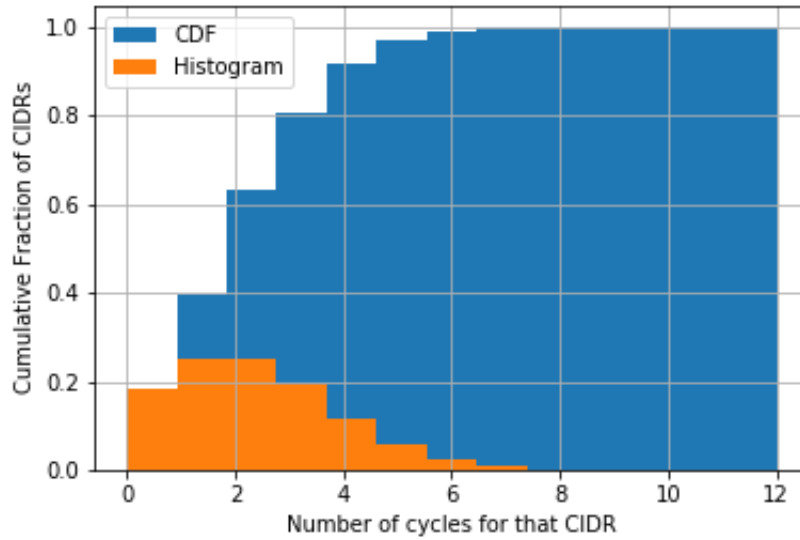


Figure 4.11. The number of cycles per CIDR.

Data Observation: Cycles are very prevalent, 81% of CIDRs have at least one cycle and nearly 1 in 5 CIDRs has 3 to 12 cycles.

Wander Distance

wander_dist is the sum total of the haversine distance (in Km) between successive LLPs. CIDRs with less LLPs will likely have smaller wander distances due to having less movement opportunities. A detailed investigation of wander distance was not in the scope of our study.

Measurement	units	mean	std	min	25%	50%	75%	max
wander_dist	km	4450.46	5921.85	0	977	2484	5863	275899

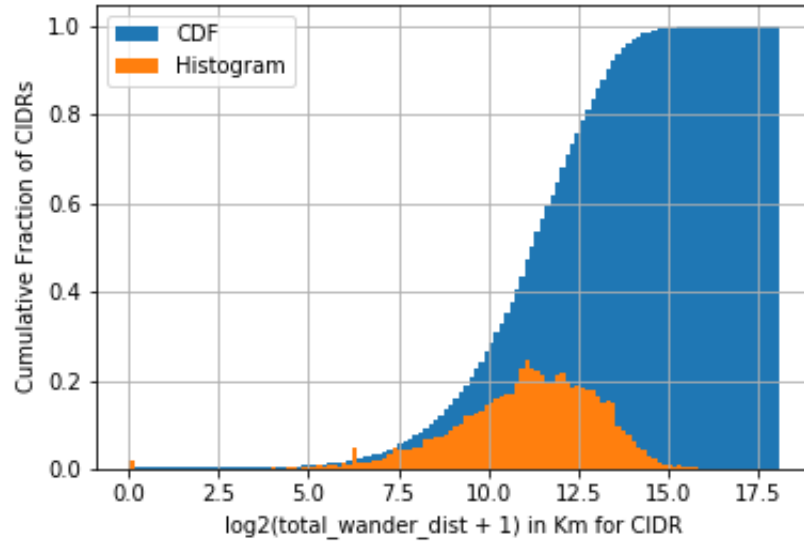


Figure 4.12. The wander distance per CIDR.

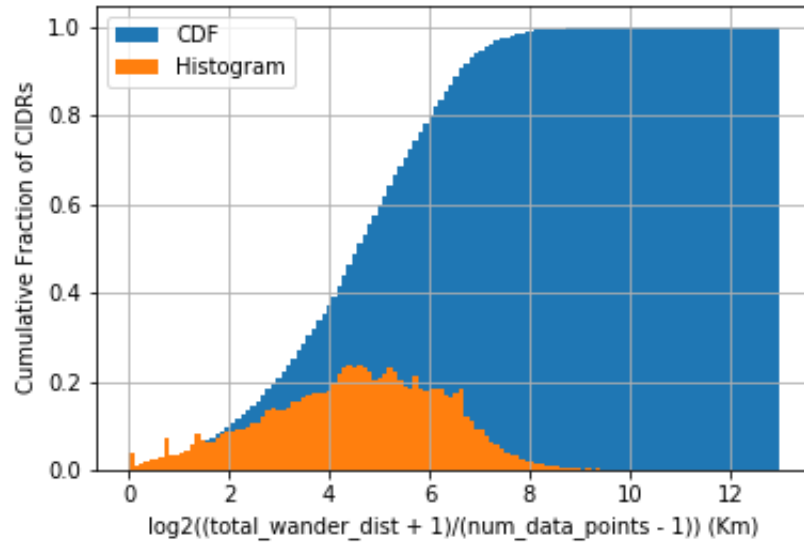


Figure 4.13. The wander distance per LLP (average wander distance).

For wander distance, we also recalculated the measure by excluding LLPs at the north and south poles and at LLP (0,0). Few data points were removed so the additional graphs were

not included. “Average” wander distance is wander distance per (num_data_points - 1) (i.e., wander distance per wander opportunity). Wander distance per CIDR is the total wander distance value of a CIDR, regardless the number of LLPs in its history. The average wander distance is a CIDR’s total wander distance divided by the number of LLPs it had in its history, giving a sense of on average how far it moved when it (if it moved) for whatever number of LLPs existed in its own unique location history.

Persistence

pers_days is the total number of days that a CIDR has consecutive LLPs that are equal (identical locations). This measure is used to calculate persistence.

persistence is the fraction of time (in days in the corpus) that a CIDR’s recorded LLP was part of a streak (i.e., stayed at the location where it was last reported in the new report). See Section 3.3.3.

persistence_most_pers is the persistence value of the individual LLP that has the most days of persistence (**pers_days**).

Measurement	units	mean	std	min	25%	50%	75%	max
pers_days	days	2871.72	417.04	0	2743	2950	3141	3433
persistence	none	0.84	0.12	0	0.80	0.86	0.91	1
persistence_most_pers	none	0.34	0.17	0	0.20	0.30	0.43	1

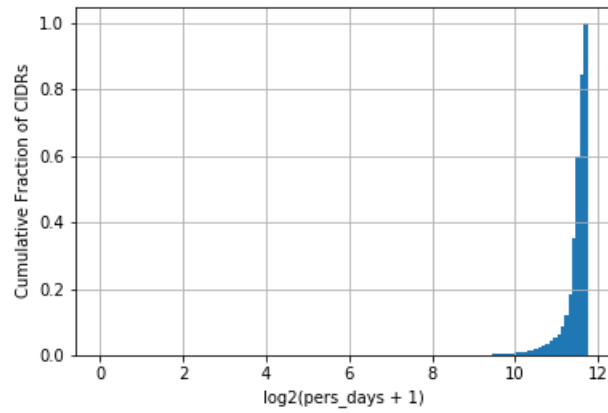


Figure 4.14. The sum total number of persistent days per CIDR.

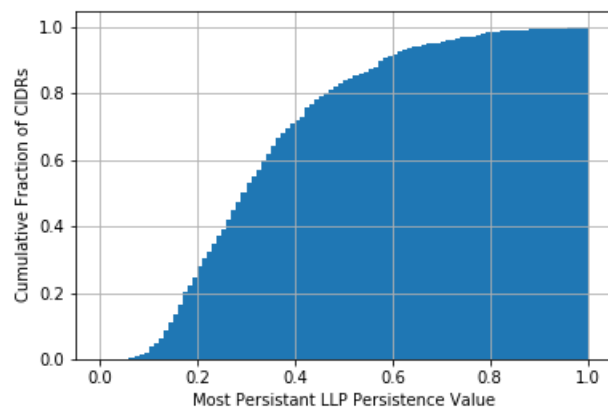
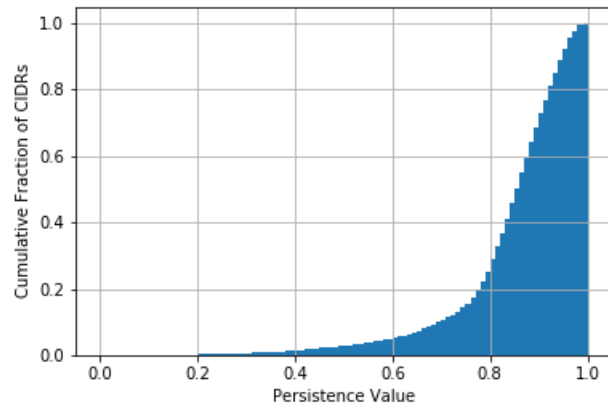


Figure 4.15. CIDR persistence.

Data Observation: CIDR moves tend to intersperse long periods of no movement. The average CIDR can be found at its last reported LLP 84% of the time (in the time window of this corpus). Yet, it is rare for a CIDR to stay in one place over a long period of time, they tend to move, sit and persist, and then move again.

Prevalence

prevalence is –for the most repeated (dominant) LLP of a particular CIDR– the fraction of times its LLP matched a previously observed LLP (it repeated an LLP) over all possible opportunities it had to repeat an LLP (its total number of data points minus one). It is a measure associated with the dominant LLP indicating how likely you are to find that CIDR at that LLP if you were to randomly select a LLP from a CIDR’s location history. Prevalence is calculated using the `max_repeat_val` (see Section 3.3.2).

Measurement	units	mean	std	min	25%	50%	75%	max
prevalence	none	0.38	0.19	0	0.24	0.35	0.49	1

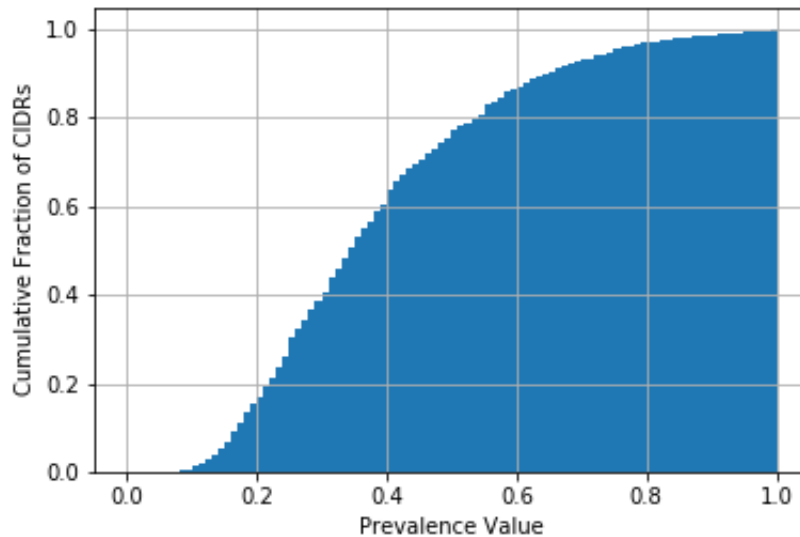


Figure 4.16. CIDR prevalence.

Data Observation: Again, the majority of CIDRs move, but at least 75% of CIDRs have a

location that accounts for nearly a quarter (24%) of their LLPs.

Persistence/Prevalence Scatter Plots

These scatter plots are included as exploratory findings. They are occluded but may be useful in future work, of interest are the disconnected clusters.

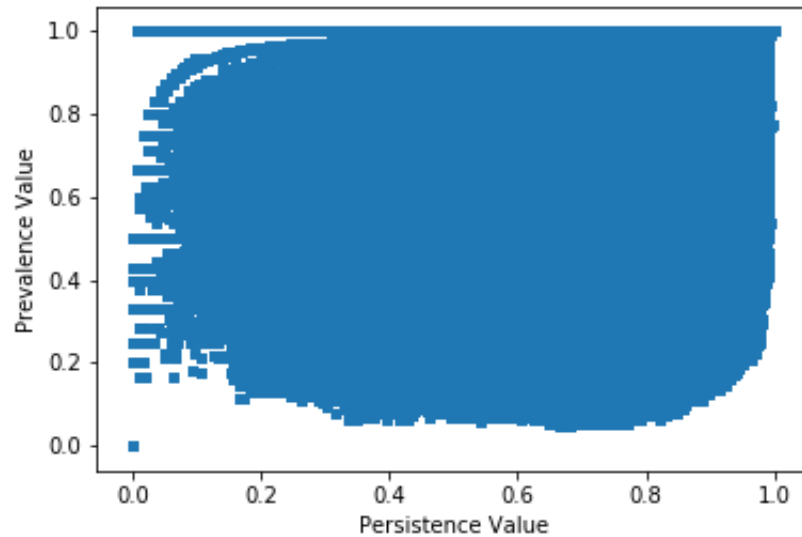


Figure 4.17. Corpus persistence/prevalence scatter plot.

4.3 Applied Measurement Results

This section applies our measurement scheme to ASs by type and to continents, plotting CDFs of the results.

4.3.1 AS Type Measurements

First a definition of an AS from RFC 1930 [36] defines an AS as:

“An AS is a connected group of one or more IP prefixes run by one or more network operators which has a SINGLE and CLEARLY DEFINED routing policy.”

The main task here is to create a mapping of CIDRs to unique AS identification numbers (ASNs) to AS type and then map all the CIDRs in the corpus. In doing this, we will discover the “stability”, based on our persistence and prevalence measurement scheme, of ASs by type. By graphing these results, we can make comparisons of the stability of networks by their types.

Mapping ASNs to AS types

The Center for Applied Internet Data Analysis (CAIDA)’s [27] website hosts downloadable copies of PeeringDB’s AS database. This database contains information on roughly $\frac{1}{3}$ of all ASs [37]. We acquired a snapshot of the AS peering database dated 20160101 and analyzed it to extract and pair unique AS identification numbers (ASNs) to AS types. The AS type field was not always reported so ASNs with missing AS types were ignored. The CAIDA taxonomy includes six AS types: Cable/DSL/ISP, Content, Educational/Research, Enterprise, Network Service Provider (NSP), Non-Profit, Not Disclosed.

Description	Filtered (% of 8,405)	Remaining
Count of valid PeeringDB entries		8,405
Entries missing ASN	2,775 (33.0%)	5,630
Entries missing AS type	537 (6.4%)	5,093
Malformed data	787 (9.4%)	4,306

Table 4.2. Data filtering of PeeringDB entries for AS type analysis [27].

Mapping CIDRs to ASNs

This mapping was accomplished by use of a Routing Information Base (RIB) file [38] acquired from *Routeviews* [39] dated 20160101 [38]. These files are dumps of Border Gateway Protocol (BGP) data that include CIDRs and AS routes. AS routes can be parsed to determine the AS (the ASN) to which that CIDR belongs. By building a radix trie from the RIB file CIDRs and populating with the ASNs, we were able to lookup the 27,672,206 CIDRs (see Table 4.1) from the corpus and associate them with their ASN.

Mapping CIDRs to ASNs to AS types

Finally, corpus CIDR to AS type mapping was performed with 10,490,348 successes. The measurements for these mappings were compiled and graphed into the following CDFs.

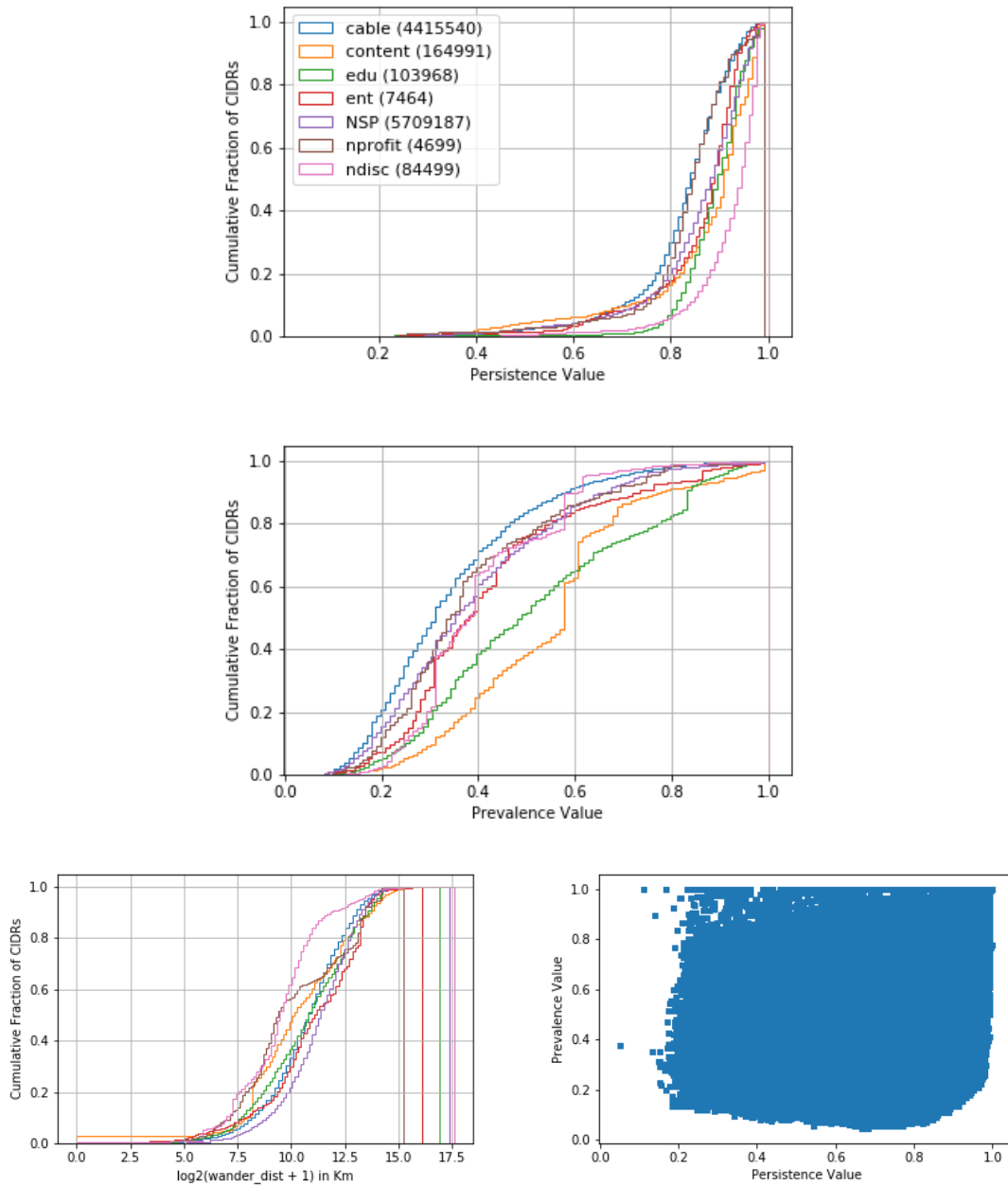


Figure 4.18. Analysis by AS type.

In Figure 4.18 most “edu” ASs have both a high persistence and high prevalence value, consistent with the intuition that they tend to establish a physical location and headquarters for their networks, and stay where established. This same behavior is shown in content provider ASs, which parallels the intuition of establishing a data center and providing content consistently from there. Why cable providing AS prevalence would be so starkly different from content providers is not understood but perhaps it has to do with how dynamic the market space. Perhaps content providers have lower stability measurements because they move more often due to their businesses being bought or sold or changing the service provider of their IT infrastructure (e.g., movement into the cloud). They may be web and app based companies who do not have or need strategically selected geographic infrastructure due to the ability to deliver information quickly over the internet. Conversely, cable/DSL/ISP ASs tend to be larger companies with geographic ties to their infrastructure and consumers that experience a less dynamic market space – yielding higher stability values. A proper answer to this question requires more research. The scatter plots here and elsewhere included for completeness.

4.3.2 Continent Measurements

Is there a relationship between stability values and geographic location?

That is the question this section set out to answer and to do so required the corpus CIDRs to be sorted geographically. Here, with continental accuracy of IPGDs higher than country level accuracy, the continent level was chosen. As IPGs could potentially change continent over time, we selected the same snapshot in time (early January 2016) as Section 4.3.1 to maintain some relation between the applied measurements. Because the Locations file contained a country designation for each LLP, all that was necessary was to map those to the proper continents. We used an ISO 3166 [40] file provided by MaxMind [41]. The process was straightforward and included the following steps:

1. Create a radix trie from the 20160105 GLC data files.
2. Create a dictionary keyed to country code from the continent mapping file (ISO3166).
3. Look up all CIDRs in the corpus in the trie, and use the returned country code to look up the Continent in the dictionary.
4. Output the compiled data to a CSV for eventual graphing.

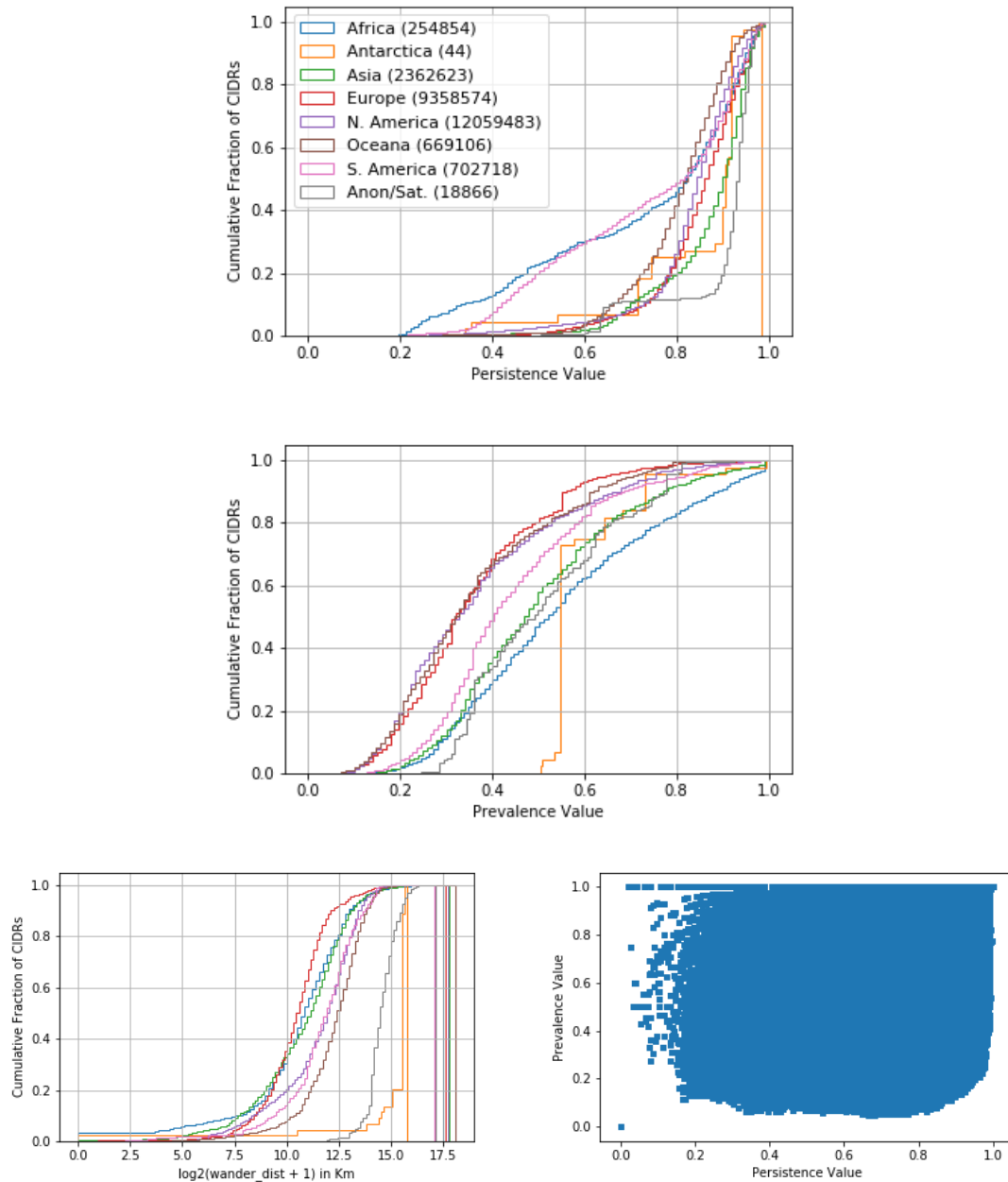


Figure 4.19. Analysis by continent.

In Figure 4.19, persistence by continent indicates that Africa and South America have very dynamic location histories, presumably due to weaker telecommunication infrastructure and poorer administrative record keeping (as these would impact the locations recorded in

an IPGD). (Note Anon/Sat. means anonymous/satellite).

4.3.3 Study Impact

Shown is a random sample of 20 IP addresses selected from the the corpus of addresses used for Eriksson’s 2010 study [12]. The gray lines show a ± 3 month window of time around April 1, 2009– the approximate date data was collected from the MaxMind database for the study. Note not only the dynamic environment after the window, but the dynamic environment inside. The date the database was polled for these IPGs clearly affected the outputs.

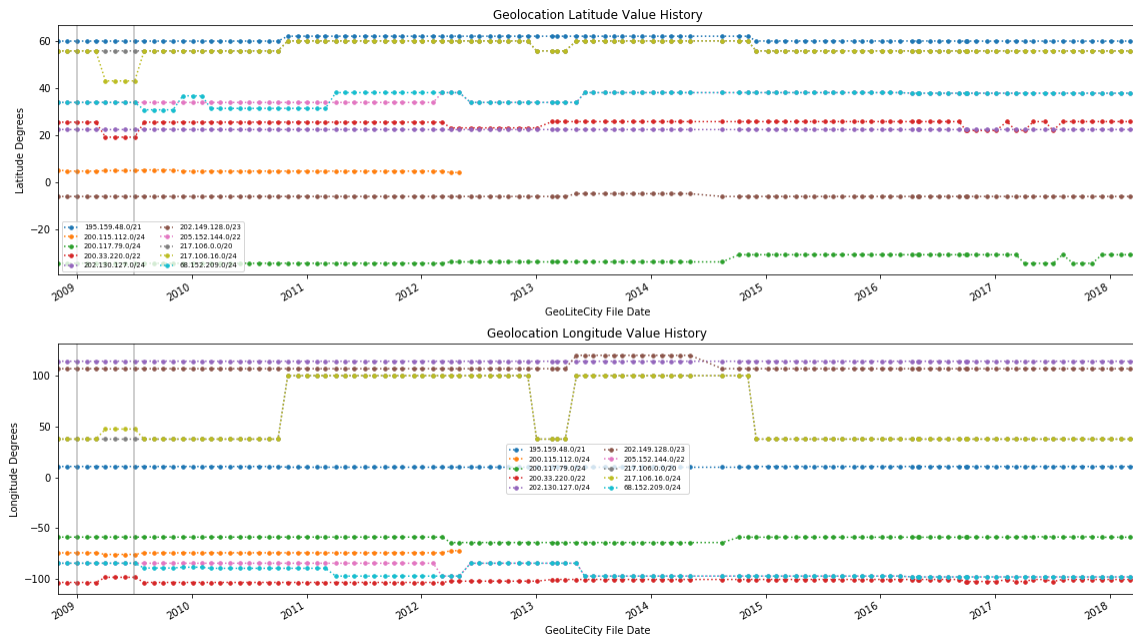


Figure 4.20. Eriksson study sample set 1 [12].

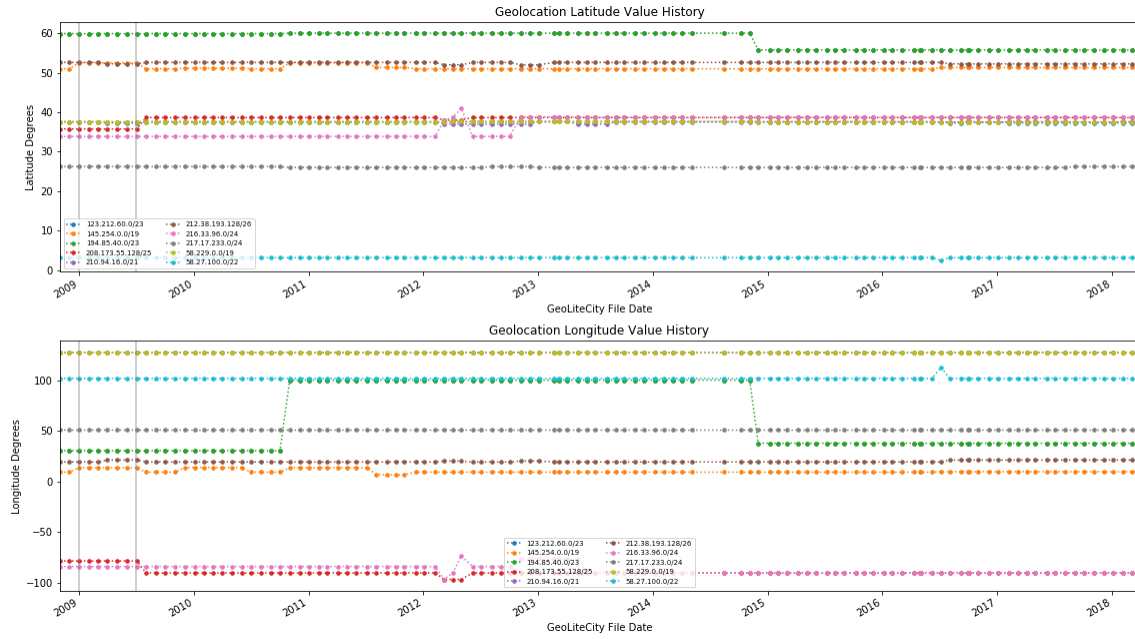


Figure 4.21. Eriksson study sample set 2 [12].

In the first set, there are jumps occurring in the window. For red, it has future jumps in latitude but its longitude is very stable. In the second set, brown shifts during the window (and latitude and longitude seem to have a mirrored shift). Meanwhile, orange appears to be in an oscillating series (where movement occasionally occurs in longitude but not latitude and vice versa) while red jumps right after the window closes. A more in depth analysis of the types of networks and the reasons (if explainable) for the movements is indicated here, and is one of the goals of our study – to illuminate these opportunities.

CHAPTER 5:

Conclusions and Future Work

We developed a framework to examine the longitudinal stability in an IPGD by extending Paxson’s notion of *persistence* and *prevalence*. In doing so, we discovered interesting behaviors in an IPGD (see Table 3.5). For example, a “cycle”, where an IPG jumps from, and subsequently returns to, a specific location (in some cases several times) and a “blink” where an IPG present in the record goes missing for some period of time (the specific CIDR is not present in the Blocks file(s)), and then re-appears. The framework identifies these interesting behaviors, but work remains to explain and understand them. Additionally, our framework can quantify the dynamic behaviors of IPG history in an IPGD, which offers a new way to compare IPGDs.

We provide two examples (AS type and continent) using the framework to measure and gain deeper insight into how well the IPGD performs in different contexts. Certainly, more analyses of this type can be performed.

We showed that researchers should question and validate location data from IPGDs as the selection of one sample will likely contain unexpected perturbations in the location information that can be *significantly* different compared to location information reported even shortly in the past (or future).

We have not deeply analyzed all of our findings, so work remains on deriving latent insights from wander distance and the scatter plots among other measurement results. For example, wander distance may have correlation to a geographic region’s IPG density and, what do the isolated fragments on the scatter plots indicate?

Topics discovered during this study that may warrant future work include:

- **Impact on Previous Academic Work**

One of the loftier research questions at the outset of this study that has yet to be answered beyond our demonstration involved the application of this methodology to previous IPGD studies to examine if and how the results of those studies could have been affected by the date of the IPGD data collection. In other words, if the

researcher selected data from a certain date or a small number of dates, *assuming* a general stability within the database, what is the scope of error that could be, or in fact was, a consequence?

- **“Movement” Correlation**

It is not necessarily the case that a CIDR’s movement in an IPGD means it *actually* moved geographically. A location change could be due to one of three occurrences: an input to the IPG algorithm changed, the algorithm itself changed, or the device hosting the network was moved physically or logically (or a combination of these occurrences). We do not currently have a means of identifying and de-tangling this mix of possible causes but the effort could involve a collaborative effort with IPGD providers on the inputs and perhaps input weightings to their proprietary IPG algorithms. A completely independent effort could also be made involving monitoring available sources of IPG data inputs (discussed in Section 2.2) on known CIDRs/Blocks and comparing changes in the available data for IPG to changes in the published IPGD information.

- **Effects of Down-sampling**

We note an observation that may weaken the measurements we obtained from the GLC database. If it is the case that MaxMind did use down-sampling to consolidate Blocks in their GLC product, then, in addition the noise in the data due to not determining the cause of the change of a CIDRs IPG, there also exists the noise created by the down-sampling method itself. One potential avenue of research may be to obtain MaxMind’s sister product (*GeoIPCity*) to GLC and perform measurements as we have across the same window of time, thereby isolating the down-sampling noise.

- **Other IPGDs**

There exists an opportunity for follow-on studies that can utilize the framework developed to compare against more recent MaxMind products as well as other IPGDs.

- **Further Investigate Odd Behaviors**

Some anomalies, e.g., blinks and cycles, were revealed that may be interesting and also may reveal methods and faults/features within MaxMind’s (and perhaps others’)

IPG methodology. What are the actual inputs to the reported location in an IPGD (in our case MaxMind's GLC)? These are regarded as proprietary trade secrets, but perhaps some insight could be gained by examining behaviors in IPG histories that seem to have a pattern to them. There are also opportunities for greater visual analysis, graphing the geolocations on a map with a time progression to get a better sense of their movement (both for individual CIRDs but also dynamic properties of the set of CIRDs) and peculiarities. In general, referencing the geolocations beyond LLPs to both the physical locations available in the IPGDs and the actual geographic locations indicated by the LLPs may reveal more insights.

- **A Study of IPGD Address Space Coverage**

A wider chronological view of IPv4 address space coverage and allocation could be created in order to understand how well IPGDs have kept up with the growth of the internet and how assignments and allocations of blocks by administrative organizations affect IPGDs.

- **Applying and Expanding the Methodology to IPv6**

Our framework and provides a conceptual base to analyzing how IPGDs perform in the IPv6 address space.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX: File Dates and Python Code

A.1 Corpus Filedates

The following includes all the filedates of the corpus in chronological order in format YYYYMMDD.

```
1 date, source
2 20081101, GeoLiteCity
3 20081201, GeoLiteCity
4 20090101, GeoLiteCity
5 20090201, GeoLiteCity
6 20090301, GeoLiteCity
7 20090401, GeoLiteCity
8 20090501, GeoLiteCity
9 20090602, GeoLiteCity
10 20090703, GeoLiteCity
11 20090802, GeoLiteCity
12 20090902, GeoLiteCity
13 20091002, GeoLiteCity
14 20091102, GeoLiteCity
15 20091203, GeoLiteCity
16 20100104, GeoLiteCity
17 20100202, GeoLiteCity
18 20100302, GeoLiteCity
19 20100402, GeoLiteCity
20 20100503, GeoLiteCity
21 20100602, GeoLiteCity
22 20100702, GeoLiteCity
23 20100802, GeoLiteCity
24 20100902, GeoLiteCity
25 20101002, GeoLiteCity
26 20101102, GeoLiteCity
27 20101204, GeoLiteCity
28 20110103, GeoLiteCity
29 20110202, GeoLiteCity
30 20110302, GeoLiteCity
31 20110404, GeoLiteCity
```

32 20110502, GeoLiteCity
33 20110602, GeoLiteCity
34 20110706, GeoLiteCity
35 20110805, GeoLiteCity
36 20110909, GeoLiteCity
37 20111007, GeoLiteCity
38 20111102, GeoLiteCity
39 20111208, GeoLiteCity
40 20120104, GeoLiteCity
41 20120208, GeoLiteCity
42 20120307, GeoLiteCity
43 20120404, GeoLiteCity
44 20120501, GeoLiteCity
45 20120606, GeoLiteCity
46 20120704, GeoLiteCity
47 20120808, GeoLiteCity
48 20120905, GeoLiteCity
49 20121002, GeoLiteCity
50 20121107, GeoLiteCity
51 20121205, GeoLiteCity
52 20130102, GeoLiteCity
53 20130220, GeoLiteCity
54 20130307, GeoLiteCity
55 20130403, GeoLiteCity
56 20130509, GeoLiteCity
57 20130604, GeoLiteCity
58 20130702, GeoLiteCity
59 20130807, GeoLiteCity
60 20130904, GeoLiteCity
61 20131001, GeoLiteCity
62 20131106, GeoLiteCity
63 20131205, GeoLiteCity
64 20140108, GeoLiteCity
65 20140205, GeoLiteCity
66 20140305, GeoLiteCity
67 20140402, GeoLiteCity
68 20140506, GeoLiteCity
69 20140816, GeoLiteCity
70 20141008, GeoLiteCity
71 20141105, GeoLiteCity

72 20141202, GeoLiteCity
73 20150107, GeoLiteCity
74 20150204, GeoLiteCity
75 20150303, GeoLiteCity
76 20150407, GeoLiteCity
77 20150506, GeoLiteCity
78 20150603, GeoLiteCity
79 20150708, GeoLiteCity
80 20150805, GeoLiteCity
81 20150902, GeoLiteCity
82 20151009, GeoLiteCity
83 20151103, GeoLiteCity
84 20151201, GeoLiteCity
85 20160105, GeoLiteCity
86 20160202, GeoLiteCity
87 20160303, GeoLiteCity
88 20160412, GeoLiteCity
89 20160429, GeoLiteCity
90 20160503, GeoLiteCity
91 20160607, GeoLiteCity
92 20160706, GeoLiteCity
93 20160802, GeoLiteCity
94 20160907, GeoLiteCity
95 20160930, GeoLiteCity
96 20161004, GeoLiteCity
97 20161101, GeoLiteCity
98 20161207, GeoLiteCity
99 20170104, GeoLiteCity
100 20170207, GeoLiteCity
101 20170307, GeoLiteCity
102 20170404, GeoLiteCity
103 20170502, GeoLiteCity
104 20170608, GeoLiteCity
105 20170704, GeoLiteCity
106 20170803, GeoLiteCity
107 20170906, GeoLiteCity
108 20171004, GeoLiteCity
109 20171106, GeoLiteCity
110 20171206, GeoLiteCity
111 20180105, GeoLiteCity

```
112 20180205,GeoLiteCity
113 20180307,GeoLiteCity
114 20180327,GeoLiteCity
```

A.2 Master File Construction Code

The following is the python code used to construct the master file from which measurements were taken.

```
1 # Jon Culbert Nov 2019
2 # This file works from last column to first in the all_glc.txt file
3 # looking up any (-,-) in the trie constructed for that filedate (column
   ). It
4 # will populate the location of the returned longest-prefix-match, if
   there is no match
5 # it will leave a '-' indicating that the trie would have returned a "
   None" because there
6 # was matching prefix of any length for that CIDR in the trie.
7
8 #import pandas as pd
9 import time
10 import json
11 import pickle
12 #import radix
13 import os
14 import gc
15
16 start_time = time.time()
17
18 chunk_size = 2700000
19
20
21 #Create a list of file dates (in reverse order) to help crawl through
   directory
22 filedate_list = []
23 with open('geolitecity_fileset.csv', 'r') as glc_files:
24     glc_files.readline()
25
26     for line in glc_files:
27         date = line.split(',')[0]
```

```

28         filedate_list.append(date)
29
30 inv_filedate_list = filedate_list[::-1]
31 print('Processing these glc_filedates:\n', inv_filedate_list)
32
33
34
35 # Update all entries in all_glc.txt data file for given date
36 #   - If no entry '(-,-)' , fill in with LPM from trie (lat,long)
37 #   - If no LPM in trie, fill with: 0
38 # Write out new line to new file
39
40 entry_ctr      = 0
41 header_line    = ''
42 #first_trip    = True
43 #test_ctr      = 0
44 #test_cycles   = 5
45
46 #offset_all_glc_ = 0
47
48
49 # Load all_glc.txt into memory (all_glc_matrix) line by line
50 print(round((time.time() - start_time), 2), 'sec - Loading all_glc.txt
    to memory...')
51 with open('all_glc.txt', 'r') as all_glc:
52     header_line = all_glc.readline()
53     #     offset_all_glc += len(header_line)
54
55     # Setup output file
56     with open('all_glc_.txt', 'w') as all_glc_:
57         all_glc_.write(header_line+'\n')
58
59
60     chunk_list = [(0, 2767942), (2767942, 5535884), (5535884,
61     8303826), (8303826, 11071768), (11071768, 13839710),
62     (13839710, 16607652), (16607652, 19375594),
63     (19375594, 22143536), (22143536, 24911478), (24911478, 27679413)]
64
65     for chunk in chunk_list:

```

```

65     all_glc_matrix      = []
66
67     for line in range (chunk[0],chunk[1]):
68         entry            = all_glc.readline()
69         #                offset_all_glc_ += len(entry)
70
71         entry_list       = entry.strip().strip('\n').split('*')
72         cidr              = entry_list[0]
73         dp_list           = eval(entry_list[1])
74         num_data_points   = len(dp_list)
75
76         # entry length check
77         if num_data_points != len(filedate_list):
78             print('BAD ENTRY', entry)
79             continue
80
81         # Populate Matrix with entries
82         # matrix = [ [cidr1, dp1, dp2,...],
83         #             [ [cidr2, dp1, dp2,...], ...      ]
84         dp_list.insert(0,cidr)
85         all_glc_matrix.append(dp_list)
86
87         entry_ctr += 1
88         if entry_ctr%100000 == 0: print(round((time.time() -
start_time), 2), 'sec - Loaded',entry_ctr/100000,'of 27.7 entries.')
89
90         print(round((time.time() - start_time), 2), 'sec - Completed
loading all_glc.txt chunk', chunk_list.index(chunk))
91
92
93
94         # Select date for processing (trie date, and column in
matrix)
95         for date in inv_filedate_list:
96
97             #test_ctr += 1
98             #if test_ctr == test_cycles: break
99
100            date_index = inv_filedate_list.index(date)
101            print('Date Index:', date_index)

```

```

102         #print(round((time.time() - start_time), 2), 'sec -
Processing matrix column dated', date, 'index', date_index, 'of',len
(filedate_list))
103         # Load the appropriate trie
104         trie_filename = date+'_geolitecity_prefix_tree_pickle'
105         #print(round((time.time() - start_time), 2), 'sec -
Loading', trie_filename)
106
107         with open(trie_filename, 'rb') as fd_trie:
108             trie = pickle.load(fd_trie)
109             #print(round((time.time() - start_time), 2), 'sec -
Completed Loading', trie_filename)
110
111             # Make matrix dp's reflect trie lookups, one date (
column) at a time.
112             # date_index = 0 ==>  matrix[:, -1]  all rows, last
column
113             # date_index = 1 ==>  matrix[:, -2]  all rows, second
to last column  etc.
114
115             # Get entries from matrix, perform lookups and
replacements
116             row_index = 0
117             for row in all_glc_matrix:
118                 # as date index increments, dp_list index
decrements
119                 cidr          = str((row[0].split('/')[0]))
120                 entry_dp      = str(row[-date_index - 1])
121                 #print( type(entry_dp),entry_dp)
122                 #print('Fixing Column:', -date_index-1)
123
124                 if entry_dp == str(('-', '-')):
125                     # return longest matching prefix
126                     rnode = trie.search_best(cidr)
127                     # Adjudicate trie return value
128                     if rnode is None:
129                         all_glc_matrix[row_index][-date_index -
1] = '-'
130
131                     else: # update matrix with LPM

```

```

132         rnode_data = rnode.data['ipg_data']
133         # ip_network_addr,prefix,date,country,
region,city,postal_code,latitude,longitude
134         rnode_data_list = rnode_data.strip().
strip('\n').split(',')
135
136         #print('lat/long:', (rnode_data_list[7],
rnode_data_list[8]))
137         all_glc_matrix[row_index][-date_index -
1] = (rnode_data_list[7],rnode_data_list[8])
138
139         row_index += 1
140
141
142         print(round((time.time() - start_time), 2), 'sec - Completed
updating all_glc_list, writing to all_glc_.txt')
143
144
145         # write the matrix to a text file
146         for row in all_glc_matrix:
147
148             cidr      = row[0]
149             dp_list = row[1:]
150
151             entry_str = cidr+'*'+str(dp_list)
152             all_glc_.write(entry_str+'\n')
153
154         print(round((time.time() - start_time), 2), 'sec - Completed
writing to all_glc_.txt,. Flushing and Collecting Garbage...')
155         all_glc_matrix = []
156         all_glc_.flush()
157         os.fsync(all_glc_)
158         gc.collect()
159
160
161     # Lookup File
162     fetch_dict    = {}
163     entry_ctr     = 0
164     offset        = 0
165     all_glc_.close()

```

```

166
167
168 with open('all_glc_.txt', 'r') as all_glc_:
169     header_line = all_glc_.readline()
170
171     offset      += len(header_line)                # account
172     for offset of header line
173     for entry in all_glc_:
174
175         entry_len      = len(entry)                # determine
176         length of the entry
177         ip_addr        = entry.split('*')[0]        # extract the
178         entry ip_addr
179
180         fetch_dict[ip_addr] = offset                # add it to
181         dict with the offset value
182
183         offset          += entry_len                # inc offset
184         value by the entry_len
185         entry_ctr        += 1                       # track
186         number of entries processed
187
188
189
190
191     if entry_ctr % 1000000 == 0:
192         print(round((time.time() - start_time), 2), 'sec - ', '(',
193               entry_ctr/1000000, 'of 27.6) Million entries processed')
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

A.3 Stats Tabulation Code

The following is the python code used to generate stats and measurements.

```
1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # will take about 50 minutes to complete
5
6  import json
7  import time
8  from haversine import haversine
9  from datetime import date
10
11
12  # "data point" is a (date, latitude, longitude) tuple instance
13  start_time = time.time()
14  entry_ctr    = 0
15  data_point_ctr = 0
16
17
18  # Make a list of the number of days between filedates
19  day_list     = [0]          # 0 is placeholder for first date
20  corpus_days = 0
21  with open('all_glc.txt', 'r') as all_geo:
22
23      hdr = all_geo.readline()
24      date_list = hdr.strip().split(',')[1:]
25
26
27      for idx in range(len(date_list)-1):
28
29          prev_yr    = int(date_list[idx][0:4])
30          prev_mn    = int(date_list[idx][4:6])
31          prev_dy    = int(date_list[idx][6:8])
32          prev_date  = date(prev_yr, prev_mn, prev_dy)
33
34          curr_yr = int(date_list[idx+1][0:4])
35          curr_mn = int(date_list[idx+1][4:6])
36          curr_dy = int(date_list[idx+1][6:8])
37          curr_date  = date(curr_yr, curr_mn, curr_dy)
```

```

38
39         day_list.append((curr_date - prev_date).days)
40
41     first_yr    = int(date_list[0][0:4])
42     first_mn    = int(date_list[0][4:6])
43     first_dy    = int(date_list[0][6:8])
44     first_date  = date(first_yr, first_mn, first_dy)
45     last_yr     = int(date_list[-1][0:4])
46     last_mn     = int(date_list[-1][4:6])
47     last_dy     = int(date_list[-1][6:8])
48     last_date   = date(last_yr, last_mn, last_dy)
49
50     corpus_days = (last_date - first_date).days
51
52
53     #print(day_list)
54     print(round((time.time() - start_time), 2), 'sec - Starting ...')
55
56
57     with open('all_glc.txt', 'r') as all_geo:
58         all_geo.readline()      # ditch first line
59
60         with open('all_glc_stats.csv', 'w') as all_stats:
61             all_stats.write('cidr,num_data_points,num_blink,num_unique,
num_repeat,num_jumps,num_cycles,wander_dist,pers_days,persistence,
persistence_most_pers,pers_lat,pers_long,max_repeat_val,prevalence,
prevl_lat,prevl_long,pers_loc_eq_prevl_loc_bool\n')
62
63             #test_runs = 100
64             # COMPILE STATISTICS FROM INPUT DATA
65             for entry in all_geo:
66
67                 #test_runs -= 1
68                 #if test_runs == 0: break
69
70                 entry_ctr      += 1
71                 if entry_ctr % 1000000 == 0:
72                     print(round((time.time() - start_time), 2), 'sec -', '('
, entry_ctr/1000000, 'of 27.6) entries processed')
73

```

```

74         # Acquire and prepare data for tabulation
75         entry_list      = entry.strip().strip('\n').split('*')
76         cidr             = entry_list[0]
77         data_point_list = eval(entry_list[1])
78
79
80         # Go though data_point_list chronologically and count every
event (once) where there was first a valid location and then
81         # the DB returned a "None" (it lost coverage over all
networks to which that IP address or CIDR belonged). The lost could
82         # be for a long period of time, but the occurrence of it
blinking out is only counted once for each time the criteria are met
83         # where it has a location, it loses it, and it resurfaces.
84         # this also counts the number of actual datapoints for that
CIDR in the whole corpus ( the amount of times the lookup did
85         # not return a "None".)
86
87
88         # Declare variables
89         num_data_points = 0
90         num_blink       = 0
91         num_unique      = 0
92         num_repeat      = 0
93         num_jumps       = 0
94         num_cycles      = 0
95         wander_dist     = 0
96
97         persistence     = 0.0
98         pers_days       = 0
99         prevalence      = 0.0
100        pers_eq_prevl_bool      = False
101
102        cycle_list        = []
103        lat_long_list     = []
104        repeat_dict       = {}    # keys are ip_addrs and value is a
tally of the number of repeats
105        streak_dict       = {}    # keys are ip_addrs and value is a
list of tallies of CONSECUTIVE repeats
106        # Each time the repeat chain (streak) is broken, a new
tally is started in the list so that

```

```

107         # at the end, the maximum streak value can be found where a
particular ip_addr has more than 1 streak.
108
109         first_data_point = True
110         blink_trigger     = False
111         index_counter     = 0
112
113
114
115
116     for data_point in data_point_list:
117
118         # Is it a valid data point?
119         if data_point != '-':
120             num_data_points += 1
121
122
123         # Retrieve lat/long data
124         lat      = data_point[0]
125         long     = data_point[1]
126         curr_point = (lat, long)
127
128
129         # For first data_point
130         if first_data_point == True:
131             num_unique          += 1
132             repeat_dict[curr_point] = 0
133             streak_dict[curr_point] = [[0,0]]
134             first_data_point      = False
135             blink_trigger         = True
136
137
138
139
140         else:    # For all data_points beyond the first
141             blink_trigger = True
142             prev_point = lat_long_list[-1]
143
144             # Add wander_dist(km)
145             try:

```

```

146         prevp_lat      = float(prev_point[0])
147         prevp_long     = float(prev_point[1])
148         prevp_flt      = (prevp_lat, prevp_long)
149
150         currp_lat      = float(curr_point[0])
151         currp_long     = float(curr_point[1])
152         currp_flt      = (currp_lat, currp_long)
153     except:
154         print('BAD llp: ',prev_point, curr_point ,
entry)
155         continue
156
157         wander_dist += int(haversine(prevp_flt,
currp_flt))
158
159
160     # curr_point HAS NOT been seen before
161     if curr_point not in lat_long_list:
162         num_unique += 1
163         num_jumps  += 1
164         cycle_list += [curr_point]
165         repeat_dict[curr_point] = 0
166         streak_dict[curr_point] = [[0,0]]
167
168
169     else:    # curr_point HAS been seen before
170         num_repeat += 1
171         repeat_dict[curr_point] += 1
172
173
174     # curr_point WAS the last point seen
175     if curr_point == prev_point:
176         # streak, inc last tally tuple of
prev_point tally list
177         streak_dict[prev_point][-1][0] += 1
178         # inc pers_days of streak dice tally
tuple list
179         streak_dict[prev_point][-1][1] +=
day_list[index_counter]

```

```

180                                     #print('idx:', index_counter, day_list[
index_counter])
181
182
183                                     else:      # curr_point WAS NOT the last
point seen
184                                     num_jumps  += 1
185
186                                     if curr_point in cycle_list:
187                                         cycle_list.remove(curr_point)
188                                         num_cycles += 1
189
190                                     # streak end, append new tally tuple to
prev_point tally list
191                                     streak_dict[prev_point]      += [[0,0]]
192
193                                     lat_long_list.append(curr_point)
194
195
196                                     else: # Its not the first data_point and the one before
it was a valid location (not ('-')), then its a blink
197
198                                     # print(data_point, 'BLINK CHECK', blink_trigger,
data_point_list[data_point_list.index(data_point)-1])
199                                     if blink_trigger == True and data_point_list[
index_counter - 1] != '-':
200                                         num_blink      += 1
201                                         blink_trigger = False
202
203
204                                     index_counter += 1
205
206                                     # end of FOR loop to process data_points
207
208 # ===== DETERMINE AND RECORD ADDITIONAL ENTRY STATS
209
210
211
212
213

```

```

214         # PERSISTENCE
215         if len(streak_dict) > 0:
216             #     for item in streak_dict.items():
217                 #         print(item)
218
219             # Determine pers_days
220             for ip_addr_key,tally_list in streak_dict.items():
221                 for instance in tally_list:
222                     pers_days += instance[1]
223             #         print('instance:', instance, )
224
225             # Reduce the list of tally values in streak_dict to the
max value
226             for ip_addr_key,tally_list in streak_dict.items():
227                 streak_dict[ip_addr_key] = max(tally_list, key =
lambda count_days_list : count_days_list[1])
228
229             # Capture most persistent location data
230             pers_location, pers_values = max(streak_dict.items(),
key = lambda key_val_tuple : key_val_tuple[1][1])
231             #print(max_streak_list)
232             most_pers_streak, most_pers_days = pers_values
233             persistence_most_pers = round(
most_pers_days / corpus_days, 4)
234             pers_lat, pers_long = pers_location
235
236
237             try:
238                 persistence = round(pers_days / (corpus_days),
4)
239             except:
240                 persistence = -1
241
242             else: # there is no streak
243                 pers_location, pers_days, pers_lat, pers_long,
persistence = [-1, -1, -1, -1, -1]
244
245                 #print('Persistence:', persistence, persistence_most_pers, '
pers_days:', pers_days, 'corpus_days:', corpus_days)
246                 #print()

```

```

247
248
249     # PREVALENCE
250     if len(repeat_dict) > 0:
251         prevl_location, max_repeat_val = max(repeat_dict.items
252         (), key = lambda key_val_tuple : key_val_tuple[1])
253         prevl_lat, prevl_long          = prevl_location
254
255         try:
256             prevalence = round(max_repeat_val / (
257             num_data_points - 1), 4)
258         except:
259             prevalence = -1
260
261     else: # there are no repeats
262         prevl_location, max_repeat_val, prevl_lat, prevl_long,
263         prevalence = [-1, -1, -1, -1, -1]
264
265     if persistence == prevalence:
266         pers_eq_prevl_bool = True
267
268     if pers_location == prevl_location:
269         pers_loc_eq_prevl_loc_bool = True
270
271     # Write stats to csv
272
273     # cidr,num_data_points,num_blink,num_unique,num_repeat,
274     num_jumps,num_cycles,wander_dist,pers_days,persistence,
275     persistence_most_pers,pers_lat,pers_long,max_repeat_val,prevalence,
276     prevl_lat,prevl_long,pers_loc_eq_prevl_loc_bool
277     csv_line = str(cidr)+','+str(num_data_points)+','+str(
278     num_blink)+','+str(num_unique)+','+str(num_repeat)+','+str(num_jumps
279     )+','+str(num_cycles)+','+str(wander_dist)+','+str(pers_days)+','+
280     str(persistence)+','+str(persistence_most_pers)+','+str(pers_lat)+',
281     '+str(pers_long)+','+str(max_repeat_val)+','+str(prevalence)+','+str
282     (prevl_lat)+','+str(prevl_long)+','+str(int(

```

```

pers_loc_eq_prevl_loc_bool))
276         all_stats.write(csv_line+'\n')
277
278     # end of FOR loop to process entry
279
280
281
282
283 # RE WRITE THE JSON LOOKUP FILE
284 # # This code creates the lookup file into all_geolitecity_pp_stats.csv
285 #
286 #     - This only requires running to produce an updated file if
287 #       all_geolitecity_pp_stats.csv is changed
288 #     - This must be run in the same dir as all_geolitecity_pp_stats.csv
289
290
291 # EXTRACT OFFSETS OF ENTRIES IN all_geolitecity_pp_stats.csv INTO A
292 #     DICTIONARY
293 start_time = time.time()
294 print(round((time.time() - start_time), 2), 'sec - Starting to make
295 dictionary ...')
296
297 fetch_dict = {}
298 entry_ctr = 0
299 offset = 0
300
301 with open('all_glc_stats.csv', 'r') as all_glc_pp_stats:
302
303     for entry in all_glc_pp_stats:
304
305         entry_len = len(entry) # determine
306         length of the entry
307         ip_addr = entry.split(',')[0] # extract the
308         entry ip_addr
309
310         fetch_dict[ip_addr] = offset # add it to
311         dict with the offset value
312
313

```

```

308         offset                += entry_len                # inc offset
        value by the entry_len
309         entry_ctr              += 1                        # track
        number of entries processed
310
311
312         if entry_ctr % 1000000 == 0:
313             print(round((time.time() - start_time), 2), 'sec -', '(',
        entry_ctr/1000000, 'of 26.3) Million entries processed')
314
315
316 print('Writing to json file...')
317
318 # Create JSON lookup file from the dictionary
319 with open('all_glc_stats_lookup.json', 'w') as lookup_file:
320     json.dump(fetch_dict, lookup_file)

```

A.4 IPv4 Coverage Tabulation Code

The following is the python code used to generate IPv4 coverage.

```

1
2 # This file must be run in the same directory as all the source data
   files.
3 # It will concatenate all the entries from files of one type (i.e.
   geolitecity blocks and locations files)
4 # into one large file (i.e. all_geolitecity.csv) where each line has:
5 #   ip_network_addr,prefix,date,country,region,city,postal_code,latitude
   ,longitude
6
7
8 import glob
9 import ipaddress as ip
10 import time
11
12 start_time = time.time()
13
14
15 # CREATE A LIST OF ALL CSV FILES IN THIS DIRECTORY
16 extension = 'csv'

```

```

17 all_filenames = [i for i in glob.glob('*.{}'.format(extension))]
18
19
20 #DECLARE WORKING FILE LISTS
21 geolite_blocks_file_list = []
22
23
24
25 # CREATE RECEIVING CSV FILES
26 with open('all_glc_coverage.csv', 'w') as all_geolite:
27
28     # Write Header Line
29     all_geolite.write ('date,coverage,overlaps\n')
30
31
32 #=====
33 # 11111111111111111111 CHECK FILES AND SORT INTO WORKING LIST 11111111111111111111
34
35     for filename in all_filenames:
36
37         # Build blocks and location file lists:
38         filename_list = filename.split('-')
39
40         # Possible filename_lists :
41         #      0      1      2      3
42         # [ date, GeoIPCity, 134 or 534, (Blocks or
Location).csv]
43         # [ date, GeoLiteCity, (Blocks or Location).csv ]
44
45         if len(filename_list) > 1:
46
47             # Sorting GeoLiteCity
48             if filename_list[1] == 'GeoLiteCity':
49                 if filename_list[2] == 'Blocks.csv':
50                     geolite_blocks_file_list.append(filename)
51             else:
52                 print('ERROR - Unclassifiable File:', filename)
53
54
55     ip_num_list = []

```

```

56     print(round((time.time() - start_time), 2), 'sec elapsed - Building
array... ')
57     for x in range(2**32):
58         ip_num_list.append('0')
59     print(round((time.time() - start_time), 2), 'sec elapsed - Completed
array... ')
60 # -----
61 # START START START START START START START START START START START
62
63     ##### GEOLITE BLOCKS FILE ANALYSIS #####
64     for geolite_blocks_file in geolite_blocks_file_list:
65
66         # Capture date
67         geolite_date_str = geolite_blocks_file[:8]
68
69         # Prepare entry list
70
71         #####
72
73
74         # OPEN FILE, EXTRACT & CHECK DATA, WRITE TO CSV
75         with open(geolite_blocks_file, 'r', encoding='utf8', errors="
ignore") as glblk:
76
77             print(round((time.time() - start_time), 2), 'sec elapsed -
Loading file: ', geolite_blocks_file)
78
79             glblk_start_time = time.time()
80             overlap_ctr = 0
81
82             glblk.readline() # ditch header line
83             for line in glblk: # valid entry filter
84
85                 line_list = line.strip().replace('"', '').split(',')
86
87                 #print(type(line_list[0]), line_list[0], type(line_list
[1]), line_list[1])
88
89                 try:
90                     startIpNum = int(line_list[0])

```

```

91         endIpNum         = int(line_list[1])
92
93         for ip_num in range(startIpNum, endIpNum):
94             x = ip_num_list[ip_num]
95             if x == '1':
96                 overlap_ctr += 1
97             else:
98                 ip_num_list[ip_num] = '1'
99         except:
100             print('BAD BLOCK LINE:', "file:",
geolite_blocks_file, line)
101
102
103
104         entry_str = geolite_date_str + str(round((len(ip_num_list)
/2**32),4)) + str(overlap_ctr)
105         all_geolite.write(entry_str+'\n')
106
107         print(round((time.time() - start_time), 2), 'COMPLETED: ',
geolite_blocks_file, 'overlaps =', overlap_ctr, round((ip_num_list.
count('1')/2**32),3), "% coverage")
108         print('Resetting array... ')
109         ip_num_list = ['0' for x in range(len(ip_num_list))]
110         print(round((time.time() - start_time), 2), 'sec elapsed - Array
Reset.')
111         print()

```

List of References

- [1] “IP geolocation in cybersecurity investigations,” accessed February 23, 2020. Available: <https://hackernoon.com/ip-geolocation-can-it-still-support-cybersecurity-investigations-34d7eee7b064>
- [2] K. Hill, “How cartographers for the U.S. military inadvertently created a house of horrors in South Africa,” accessed February 23, 2020. Available: <https://gizmodo.com/how-cartographers-for-the-u-s-military-inadvertently-c-1830758394>
- [3] K. Hill, “How an internet mapping glitch turned a random Kansas farm into a digital hell,” accessed February 23, 2020. Available: https://splinternews.com/how-an-internet-mapping-glitch-turned-a-random-kansas-f-1793856052#_ga=2.174663526.538810931.1546537765-410898999.1545173683
- [4] B. Huffaker, M. Fomenkov, and K. Claffy, “Geocompare: A Comparison of Public and Commercial Geolocation Databases,” *Proc. NMMC*, pp. 1–12, 2011.
- [5] S. Zander, “On the accuracy of IP geolocation based on IP allocation data,” *Swinburne University of Technology. Centre for Advanced Internet Architectures*, 2012.
- [6] Y. Shavitt and N. Zilberman, “A geolocation databases study,” *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 10, pp. 2044–2056, 2011.
- [7] D. Komosnỳ, M. Vozňák, and S. U. Rehman, “Location accuracy of commercial IP address geolocation databases,” *Kaunas University of Technology*, 2017.
- [8] A. hexasoft. 10 Applications of IP Address Geolocation. [Online]. Available: <https://blog.ip2location.com/knowledge-base/10-applications-of-ip-address-geolocation/>. accessed March 26, 2019.
- [9] P. Gill, Y. Ganjali, B. Wong, and D. Lie, “Dude, where’s that IP?: Circumventing Measurement-Based IP Geolocation,” in *Proceedings of the 19th USENIX conference on Security*. USENIX Association, 2010, pp. 16–16.
- [10] M. Crovella and B. Krishnamurthy, *Internet Measurement: Infrastructure, Traffic and Applications*. John Wiley & Sons, Inc., 2006.
- [11] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe, “Towards IP geolocation using delay and topology measurements,” in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM, 2006, pp. 71–84.

- [12] B. Eriksson, P. Barford, J. Sommers, and R. Nowak, "A learning-based approach for IP geolocation," in *International Conference on Passive and Active Network Measurement*. Springer, 2010, pp. 171–180.
- [13] B. Wong, I. Stoyanov, and E. G. Sirer, "Octant: A Comprehensive Framework for the Geolocalization of Internet Hosts." in *NSDI*, 2007, vol. 7, pp. 23–23.
- [14] Y. Wang, D. Burgener, M. Flores, A. Kuzmanovic, and C. Huang, "Towards street-level client-independent IP geolocation." in *NSDI*, 2011, vol. 11, pp. 27–27.
- [15] "American Registry for Internet Numbers," accessed October 13, 2019. Available: <https://www.arin.net/>
- [16] "Asia Pacific Network Information Centre," accessed October 13, 2019. Available: <https://www.apnic.net/>
- [17] "Maxmind, 2019," accessed October 13, 2019. Available: <https://www.maxmind.com/en/home>
- [18] "IP2Location, 2019," accessed October 13, 2019. Available: <https://www.ip2location.com/>
- [19] "World Wide Web Consortium (W3C), 2019," accessed October 13, 2019. Available: <https://www.w3.org/>
- [20] "Whois, 2019," accessed October 13, 2019. Available: <https://whois.net/>
- [21] B. Gueye, S. Uhlig, and S. Fdida, "Investigating the imprecision of IP block-based geolocation," in *International Conference on Passive and Active Network Measurement*. Springer, 2007, pp. 237–240.
- [22] S. Siwpersad, B. Gueye, and S. Uhlig, "Assessing the geographic resolution of exhaustive tabulation for geolocating internet hosts," in *International Conference on Passive and Active Network Measurement*. Springer, 2008, pp. 11–20.
- [23] I. Poesse, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye, "IP geolocation databases: Unreliable?" *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 53–56, 2011.
- [24] M. Gharaibeh, A. Shah, B. Huffaker, H. Zhang, R. Ensafi, and C. Papadopoulos, "A look at router geolocation in public and commercial databases," in *Proceedings of the 2017 Internet Measurement Conference*. ACM, 2017, pp. 463–469.
- [25] "Maxmind, 2009," accessed October 13, 2019. Available: <https://web.archive.org/web/20081216022214/http://www.maxmind.com/app/geolitecity>

- [26] B. Huffaker, M. Fomenkov *et al.*, “DRoP: DNS-based Router Positioning,” *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 5–13, 2014.
- [27] “CAIDA, 2019,” accessed October 13, 2019. Available: <https://www.caida.org/>
- [28] “Maxmind accuracy, 2019,” accessed October 13, 2019. Available: <https://www.maxmind.com/en/geoip2-city-accuracy-comparison>
- [29] “European Union’s General Data Protection Regulation (GDPR), 2019,” accessed October 13, 2019. Available: <https://gdpr-info.eu/>
- [30] “The WayBackMachine, 2019,” accessed October 13, 2019. Available: <https://archive.org/web/>
- [31] “ipaddress — ipv4/ipv6 manipulation library,” accessed October 30, 2019. Available: <https://docs.python.org/3/library/ipaddress.html>
- [32] “netaddr 0.7.19 documentation,” accessed October 30, 2019. Available: <https://netaddr.readthedocs.io/en/latest/>
- [33] V. Paxson, “End-to-end routing behavior in the internet,” *IEEE/ACM transactions on Networking*, vol. 5, no. 5, pp. 601–615, 1997.
- [34] W. R. Parkhurst, *Routing First-Step*. Cisco Press, 2004.
- [35] MaxMind Customer Service, “Maxmind request,” email, September 13, 2019.
- [36] J. Hakinson and T. Bates, “Guidelines for creation, selection, and registration of an Autonomous System (AS),” Internet Requests for Comments, RFC 1930, March 1996, accessed October 30, 2019. Available: <https://tools.ietf.org/html/rfc1930>
- [37] “PeeringDB, 2019,” accessed October 13, 2019. Available: <https://www.peeringdb.com>
- [38] “rib.20160101.0000.bz2, 2019,” accessed October 13, 2019. Available: <http://archive.routeviews.org/bgpdata/2016.12/RIBS/>
- [39] “University of Oregon Route Views project, 2019,” accessed October 13, 2019. Available: <http://www.routeviews.org/routeviews/>
- [40] “Iso 3166, 2019,” accessed December 22, 2019. Available: <https://www.iso.org/iso-3166-country-codes.html>
- [41] “Maxmind, 2019,” accessed December 22, 2019. Available: https://dev.maxmind.com/geoip/legacy/codes/country_continent/

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California