TU WIEN Informatics

Relevance of multiple vantage points for Internet topology measurements

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Nihad Abou-Zid, BSc
Matrikelnummer 01426835

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Privatdoz. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Edgar Weippl
Mitwirkung: DI Dr.techn. Johanna Ullrich, BSc

Wien, 4. September 2025

Nihad Abou-Zid	Edgar Weippl

TU WIEN Informatics

Relevance of multiple vantage points for Internet topology measurements

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Nihad Abou-Zid, BSc
Registration Number 01426835

to the Faculty of Informatics

at the TU Wien

Advisor: Privatdoz. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Edgar Weippl
Assistance: DI Dr.techn. Johanna Ullrich, BSc

Vienna, September 4, 2025

Nihad Abou-Zid	Edgar Weippl

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Nihad Abou-Zid, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 4. September 2025

Nihad Abou-Zid

Acknowledgements

The completion of this work is a significant milestone for me and my future. The writing of this thesis would not have been possible without the support and contributions of various persons and institutions. Their guidance has been crucial in shaping this work into its final form.

First, I express my deep gratitude to Dr. DI Johanna Ullrich, BSc, for her exceptional supervision throughout this project. Her technical explanations, detailed feedback, and the provision of helpful resources have been of great value. Her mentorship and expertise have significantly increased the quality and depth of my research.

I am particularly grateful to Mag. Peter Brandstätter from the Fachhochschule Steyr for his recommendations regarding structure and organization, especially that brainstorming afternoon at the beginning, which has helped to start and shape this work. Special thanks go to Lisa Scelli, MA, my wife, whose organizational support and motivation enhancement, but also patience, have been essential during challenging periods of this research.

The Technische Universität Wien also deserves deep recognition for providing both the resources and the opportunity to pursue a high-quality education at a well-renowned institution. I would like to specifically acknowledge Privatdoz. Dipl.-Ing. Mag.rer.soc.oec. Dr.techn. Edgar Weippl for making this master thesis possible.

The Center for Applied Internet Data Analysis has been fundamental in this research through their exchange and support regarding relevant data for this work.

In addition, I am deeply grateful to my family and friends who have provided support throughout this journey. Their encouragement and understanding have been essential to maintain the focus and determination needed to complete this master thesis.

This work represents not just individual effort, but is a result of support, guidance, mentorship, and resources from an entire network of individuals and institutions. To everyone who has contributed to this achievement, whether explicitly mentioned or not, I am deeply grateful.

Kurzfassung

Das Scannen von Netzwerken hat unter anderem den Zweck Netzwerküberlastungen, Ausfälle, und sicherheitsrelevante Probleme zu erkennen. Nur wenige wisschenschaftliche Ressourcen, die den Kompromiss zwischen der Anzahl der Messpunkte (Vantage points) und der gefundenen Netzwerktopologie analysieren, existieren. Diese Arbeit vergleicht und analysiert die unterschiedlichen Ergebnisse bei Netzwerkscans mit einem Messpunkt, gegenüber Scans mit mehreren Messpunkten. Die Methodik des Systematischen Literaturreviews wird verwendet um den durch die Literaturauswahl entstehenden Bias zu minimieren. Bereits existierende Daten werden verwendet und kombiniert um zu neuen Erkenntnissen zu gelangen. Dazu werden weitere Graphstrukturen verwendet, mit welchen das Internet modelliert wird. Um eine wissenschaftliche Systematik und Planbarkeit sicherzustellen wird der Cross Industry Standard Process for Data Mining (CRISP-DM) Ansatz für den Analyse-Prozess angewandt. Die gesammelten und aggregierten Daten werden wie Daten aus einem Feldexperiment behandelt. Die Limitierungen bezüglich der Interpretation, Validität und Generalisierbarkeit werden am Ende der Arbeit, durch Anwendung von Wissen aus experimentellen Methodologien, analysiert.

Abstract

Internet network scanning serves the purpose of dealing with network congestion, outages, and security-related matters, to name only a few. There is little scientific evidence on the trade-off between the number of vantage points and topology coverage. This study analyzes the differences between measurements with only one vantage point and those with multiple ones as well as how the measurement results behave in relation to the number of vantage points. Some parts of the systematic survey methodology are used for the exploration of the available scientific literature. Existing data are used and combined to gain new insights into patterns of different types of Internet topology and the structure of the Internet. Graph structures are used to represent the Internet, and the analysis is performed on the basis of those created graphs. The CRISP-DM approach is applied to ensure scientific rigor and predictability in the analysis process. The gathered and aggregated data is treated as data coming from a field experiment. Possible bias, threats to validity, and generalizability are analyzed using the knowledge from experimental methodologies.

Contents

CHAPTER 1

Introduction

This chapter gives an overview of what internet topology is, what problems it needs to handle, and why it is relevant both for research and industry. The end of this chapter states what wants to be achieved with this work.

1.1 Problem statement

Under *network topology* one understands a model of connections between different nodes of a network. As the Internet is a network, the *Internet topology* is a specific form of a network topology [1]. A topology is often represented as a graph and usually also includes a classification of the nodes that indicates to which part of the network they belong. This work focuses only on the discovery of the internet topology. More specifically, it focuses on the marginal utility of additional Vantage Point (VP)s. The marginal utility focuses on the coverage, which measures how many real nodes are present in the discovered topology. However, the problem is that the real set of nodes is not known. That is why the marginal coverage is analyzed instead of the total one. Other aspects that are considered for the utility mentioned above are characteristics such as node classification quality [2].
The topology of the Internet can be seen at the following four levels:

- **Internet Protocol (IP) interface level**: IP router and end system interfaces are modeled entities. The topology is obtained using data collected by a testing tool like the one discussed in chapter 4.1.1. Different approaches of how to gather these data exist, each having different advantages and disadvantages, as well as implications for the interpretation of the resulting data.
- **Router level**: Routers are the modeled entities. In order to achieve this representation, different IP interfaces of the same router are aggregated into one router. This

aggregation technique is called *alias resolution* and is further discussed in chapter 4.1.3. Its accuracy has a high influence on the observed topological characteristics.

- **Point-of-Presence (PoP) level**: Here, routers or interfaces which are found to be geographically co-located, are aggregated into a PoP level topology.
- **Autonomous System (AS) level**: This level provides information about the connectivity of ASes. The data for this level are primarily not drawn from active measurements but gathered from inter-domain routing information and address databases.

The internet topology is primarily created using data from active measurements. Topology discovery tools face challenges such as non-responding routers, also called *anonymous router*s, firewalls blocking traffic, and load balancing leading to different paths taken for different packets. In order to evaluate the collected data, one has to deal with those topics as they have a direct impact on the data [1]. Another challenge is the choice and the number of VPs for which not only the geographical location but also other characteristics are relevant [2]. The number of VPs is related to the efficiency of the measurements. There are multiple scanning campaigns that scan the Internet on a regular basis. This leads to a high network load and is often inefficient and costly [3]. The main disadvantage of active measurements is that they can only discover forward paths towards a given destination. Tools such as Traceroute, presented in Chapter 4.1.1, support the discovery of the reverse path, inverting destination and source. The reverse and forward paths would not necessarily consist of the same nodes, especially when load balancing techniques are used. Active measurements also miss backup paths, which are paths that are only used when the primary path is broken. This issue may eventually be solved by using larger time windows for probing and more VPs. Traceroute has been found to systematically malfunction when traffic is faced with load balancing. Another limitation is Multiprotocol Label Switching (MPLS), which gives an AS the ability to hide the internal topology. [1]

1.2 Motivation

A sound knowledge of the topology of the different levels of the Internet is necessary in order to perform network diagnostics and resource management. It can help when deciding where to add new routers in order to reduce bottlenecks, for instance, but also to figure out if the employed devices are configured correctly. It is also used for simulation and evaluation of proposed protocol changes as well as for modeling and monitoring purposes. The gathered topology data can also be used to build a graph whose characteristics can then be analyzed using graph methods. An example would be the resilience of a graph to failure or the routing efficiency. The acquired knowledge of appropriate metric values might influence the engineering of future topologies, repair strategies for eventual failures, and understanding of the fundamental properties of existing networks. Properties derived from internet graphs can be used as input to

simulations. Because of the complexity of the Internet simulations play an important role when trying to determine how different parts of the system behave under certain conditions. Network maps can be useful to monitor the connectivity of the internet like, for example, a connectivity comparison before and after an attack on the network infrastructure. Network topology information can also be helpful in deciding on the locations of replication servers used to store distributed data [1].

1.3 Goal

The aim of this work is to analyze the problem of finding a topology of the Internet and the choice of a suitable number of VPs depending on the desired coverage. Not only is the number, but also the different characteristics of the individual VPs taken into account. In contrast to existing research on the topic of marginal utility of VPs for internet topology measurements, this work evaluates the utility quantitatively using graph comparison methods. The hypothesis that there is a positive correlation between the number of VPs and the graph coverage is verified.

CHAPTER 2

Approach and structure

2.1 Approach

The following pages will give information about the methodologies used for the thesis: Systematic Literature reviews and Cross Industry Standard Process for Data Mining (CRISP-DM). Knowledge of experiments has been used throughout the process in order to evaluate the literature resources, the gathered data, and their further use for the thesis, as well as the limitations of the results.

2.1.1 Experiments

Experiments analyze whether and how different characteristics of a system relate to each other. They try to identify the cause and effect of different properties of a system. Most often, the term **inus condition** is more accurate for a cause. [4] defines the term as "Insufficient but non-redundant part of an unnecessary but sufficient condition". The analysis of cause and effect is done by the manipulation of certain factors followed by a systematic observation of the outcome. Usually many factors contribute to the outcome of which many are often unknown. This is one of the reasons why causal relationships are dependent on the context and not deterministic but just increase the probability for a certain outcome. Cause and effect are both properties, states or events in any kind of system. The term **effect** can be explained using the counterfactual model. Counterfactual is the knowledge of what would have happened if there had not been any treatment. The difference between the two different outcomes is the effect. Usually the counterfactual model cannot be tested on the same test objects at the same time and the test has to be approximated. It is valid to say that a causal relationship exists between a cause and an effect if

- The cause chronologically happened before the effect

- There is a connection between the cause and the effect
- No other explanation can be found as cause
- When the cause is manipulated, the outcome changes according to the change made to the cause
- Different methods are used in order to minimize the probability for a different cause leading to the observed effect

Experiments are good for building causal descriptions which state how a different treatment affects the outcome. They do less well in helping to build causal explanations which name the limitations and causal conditions under which the effects can be observed. Causal explanations are important to reason about the transferability of causal effects features. [5, p. 2-10].
Different types of experiments can be distinguished based on the assumptions made, the chosen test objects and different characteristics that will be explained in more detail in the following sections.

- **True/Lab/Controlled experiments** are the ideal form of experiments. In this case researchers have complete control of the experiment setup. This means that the test objects can be chosen randomly and that groups of objects or participants are completely isolated from each other. The test objects should be mostly homogeneous - they should not differ in characteristics being relevant for the study. Most of the time only a subset of the relevant characteristics is known. Furthermore there is complete control over extraneous and moderating variables as well as a constant and accurate treatment and outcome measures [6]. This setting isolates causal relationships which reduces the generalizability [7].
- **Quasi/Field-experiments** are less restrictive on the context and the assumptions of the setup. However, this means that the results have more threats to the different types of validity [6]. If statistical hypothesis tests are used, the test population has to be representative of the target population. Another challenge is that the environment and toolset should resemble the reality as much as possible. Usually, trade-offs are made in order to reduce the factors that influence the treatment. The most realistic setting for software engineers is their real environment which is usually at an office or at home. There is no control over interruptions, communication with other participants of the test and other influencing factors.

The following forms of validity can be distinguished:

- **Internal validity** refers to the correctness of the reasoning about the connection of cause and effect. It is defined as the validity of the implications that a manipulation of the treatment-variable A led to the change of the output-variable B. The change could have been caused by other factors which are then called confounding factors.

- **External validity/Generalizability** defines how much causal relationships hold under different settings, with different persons, treatments and outcomes.

Both types of validity are negatively correlated to each other [7].

All experiments have in common that the analysis is based on data. The measurements used to collect the data should be as accurate as possible and reproducible. Often, multiple studies on one or more closely related research questions are conducted. This is done to deepen the knowledge, improve the measurements, and experimental methods as well as to increase the reliability and validity. Measurements can be used to predict certain outcomes such as the required development and testing effort for a software project based on a detailed design. Another use case is to provide feedback to software engineers about the quality of the software under development. The reliability of empirical studies indicates how accurate and replicable the measurements are. In order to further evaluate and assess the quality of measurements, different forms of validity exist:

- **Content validity** relates to the thoroughness with which a problem domain has been covered and of which an inclusive definition is required.
- **Predictive validity** explains to what extent measurements can be used to predict the outcome of a related event. It is dependent on the relationship between the measurement and the criterion (outcome) [8].

Compared with other scientific areas like physics, chemistry, or medicine, the percentage of controlled experiments in software engineering is low and there is is no or little empirical validation. Often the information about which tools have been used is missing in the experiment's report. Another observation is that the set of participants is not representative for the industry as most of the participants of experiments are students and the relative number of professionals is low. Usually the participants are selected using convenience sampling which means that participants meeting the required characteristics and that are easy to find are selected. An example of this are students that are taken for an experiment performed by a teacher [7].

Experiments are used a lot in hard sciences like physics, chemistry, and medicine to gain new knowledge and make decisions based on evidence. In software engineering the decisions about which tools and languages to use are mostly based on personal experiences and preferences which introduces a large bias [9]. An example of medicine during the 1980s shows how negative the effects of scientific work without formal methods can be. Studies discovered that estimations of specialists are more adequate if they are based on experimental studies and systematic reviews. Experimental studies are time-intensive, complex to realize and generate a lot of data. There are tools available that support the design and creation process of experiments in the field of software engineering. One such tool has been built by the Experimental Software Engineering group of COPPE/UFRJ

(see [10]). The tool includes features for knowledge management, experiment-support tools like a software development environment and is suitable for distributed teams. The common understanding in the software engineering community is that experiments are necessary in order to create and improve methods, tools as well as processes for the development and maintenance of software systems. The difference to the hard sciences is that software projects largely differ in characteristics like size, tools, programming languages, environment and the resources just to name a few [9].
The different areas of science can be distinguished by how theoretical instead of correlational the analytical approach to the respective subject is. The latter explains the phenomena of perceivable events through relations, while theoretical studies explain relations using principles and constructs that are often multiple levels of abstractions from the empirical data. The scientific evolution of knowledge works by connecting theoretical concepts to perceivable data. In less developed sciences those concepts are not formally connected with each other. The existence of the relations is assumed and described verbally which makes them more difficult to test [8].

2.1.2 Systematic literature reviews

Systematic Literature Review (SLR)s, also called systematic reviews, are a certain type of study. Below is a brief overview of the main characterization of study types followed by a more detailed description of SLRs.

- **Primary studies** can be used to observe trends and behaviors, verify hypotheses, and evaluate technological solutions. An example of this category is a usability study. The following subcategories can be distinguished:
  - In vivo: experiment is conducted in the real environment
  - In vitro: a controlled environment is used - university, research center
  - In virtuo: the focus of interest is the interaction between participants or small groups of participants. A digital model of reality is used. The environment is a controlled one.
  - In silico: the participants and the environment are abstracted by a digital model. There is no human intervention with the environment.
- **Secondary studies** are used in order to identify, evaluate and interpret scientific works being relevant for a research question. They analyze the results of primary studies and are able to identify scientific gaps. They are not an alternative, but a complement to primary studies. SLRs are one type of secondary studies [9].
- **Tertiary studies** analyze secondary studies using the methodology of software reviews in the case of a tertiary review. It can be less time consuming but requires existing systematic reviews of the related topic [11].

The purpose of a SLR is to evaluate and interpret all scientific literature and data relevant for a certain research question. The choice of the research question is the most important part of a SLR since the rest of the process is based on it [12].
The review protocol specifies how the review should be conducted. The protocol should be reviewed by external people. A review protocol usually specifies the following points:

- Reasons for the review
- Research questions: a mapping study done beforehand might help forming the questions
- Search strategy: decide on specific search terms, libraries, and types of literature
- Selection criteria: try the criteria on a subset of primary studies
- Study selection procedures: how many reviewers should be used per study and how will disagreements be handled? When deciding to exclude or include a certain research artifact, the decision should be documented, including the reason for the decision. Usually a bibliography management tool is used for this.
- Study quality checklists: evaluate bias, internal and external validity
- Data extraction strategy: a proper validation process should be ensured
- Synthesis of extracted data: whether a meta-analysis should be performed - quantitative analysis of the data using statistical methods
- Dissemination strategy: how should the results be communicated (press-release, web page, academic journals, conference, ...)
- Project timetable

The process is usually not strictly sequential. It is therefore possible to go back to previous steps to improve something due to new insights [11] p. 6 ff.
SLRs are due to the high effort required, usually done by multiple researchers. In such a case it is advisable for all of those researchers to participate in the creation of the protocol in order to fully understand it. All participants must follow the protocol to reduce potential bias or errors. In order to define the search terms the research question is disassembled in the different parts. Furthermore the evaluation method and the response measure are identified. Those keywords together with those from primary studies are then used to define the main search terms. Afterwards the main terms are connected by an AND. Synonyms of the main terms are introduced using OR-connections. Different sources should be used for the query as one source usually only finds a subset of the relevant resources. Different authors recommend to not only rely on the title and abstract for a primary selection but to also use the conclusion as they claim that the quality of abstracts in software engineering is usually bad and not sufficient to evaluate the

relevance of the resource. For the extraction of data, extraction forms should be used [12].

When formulating the research questions, the following aspects should be kept in mind:

- The population to which the research questions refer to
- The type and method of intervention - artefact (software tool/technology/procedure) to address a certain problem
- The comparison - with which artefact is the intervention compared - it is also called the control treatment and it has to be described in detail as well as the intervention.
- Outcomes - The change in characteristics that are important to the target population like for example the fault tolerance, performance, development costs of a certain software.
- Context - Treatments performed in a certain (e.g. academic) context might not be transferrable to an industry context.

Concluding it can be said that SLRs are well suited for performing research with less bias and leading to a good overview of the existing already performed research regarding a certain scientific field. There is one type of bias considered worth mentioning that SLRs may suffer from: Publication bias. It refers to the fact that researchers are more likely to publish positive results than negative ones [11].

2.1.3 Cross Industry Standard Process for Data mining

CRISP-DM is a process model that provides a framework for data mining projects. Data mining is a complex process whose success depends on the right mix of appropriate tools and experienced analysts. CRISP-DM is independent of the industry sector and the technology used. It aims at making large data mining projects less costly, more reliable, repeatable, and faster. Furthermore, it helps in having reasonable expectations of the data mining process. CRISP-DM helps to structure the project, provides advice for each task, as well as the communication and documentation of the results, the latter part being the most important.

The process model defines a small number of phases, each consisting of generic tasks which are designed to be as complete and stable as possible. Each of those tasks consists of specialized tasks. One such generic task could be "build model" with one specialized task "build response model". Another more detailed level is the process instance level, which records actions, decisions, and results of actual data mining engagements. The process instance level is organized according to the tasks defined at higher levels while

representing what actually happened.

The lifecycle of a data mining project adhering to CRISP-DM is made up of the following phases. Due to data mining's iterative nature, the sequence and focus of phases depends on the outcomes of previous phases.

1. Business understanding: This phase focuses on the project objectives and requirements which are then converted into a data mining problem definition and a preliminary project plan.

2. Data understanding: During this phase, initial data collection is performed together with activities to become familiar with the data. Eventual data quality problems are identified and first insights are gained, which may lead to the detection of interesting subsets to form hypotheses.

3. Data preparation: Here, the final dataset is constructed. This phase is usually performed multiple times. It can contain tasks such as attribute selection, cleaning, transformation, and forming new attributes.

4. Modeling: The modeling techniques are selected and applied. The parameters of the chosen models are optimized.

5. Evaluation: This phase serves the purpose of evaluating the model and should ensure that it properly achieves the business objectives.

6. Deployment: In general, the creation of the model is not the end of the project. Since end users are usually not the data analyst, it should be clear to the user upfront what actions should be performed to make use of the models [13].

2.1.4 Approach

The approach for the thesis is inspired by the methodology of systematic reviews, while only a subset of the methodology will be used for the thesis. This is done to reduce bias while keeping the effort at a reasonable level. Based on the research question "Relevance of multiple VPs for Internet topology measurements" the following search string has been defined:

Known primary studies have been used to identify the main terms and their synonyms. A search using the constructed string has been be performed using following digital libraries: ACM Digital Library [14] and IEEE Xplore [15]. The search string at 2.1 uses the syntax required by the IEEE Xplore Command Search. The metadata field includes the abstract, keywords and bibliographic citation data (document title, publication data, etc.) [16]. The search string for the ACM Advanced Search looks more complex as there is no field comparable to the metadata field used in the search string for IEEE Xplore. The clauses

Algorithm 2.1: Search string for IEEE Xplore

```
("All Metadata":"Traceroute" OR "All Metadata":"traceroute*" OR "All
Metadata":"Scan*" OR "All Metadata":"scan*") AND ("All
Metadata":"Network*" OR "All Metadata":"network*" OR "All
Metadata":"Internet" OR "All Metadata":"internet") AND ("All
Metadata":"measure*" OR "All Metadata":"Measure*" OR "All
Metadata":"utility" OR "All Metadata":"coverage") AND ("All
Metadata":"graph" OR "All Metadata":"topolog*")
```

for the main keywords each contain further clauses for searching in the abstract, title, and keyword fields. The search string can be found in the appendix chapter of the thesis. The search has been restricted to the following types of research literature: conference papers, journal articles, and requests for comments. Only literature written in English has been considered.

The primary selection of relevant work has been done by looking at the titles. Literature that is obviously not relevant because it focuses on a different scientific field such as medicine is excluded during this step. The second selection has been performed based on the abstract, and if needed for deciding on the relevance also the conclusion has been taken into account. The decisions of both phases of the selection process have been documented. This was necessary to be able to verify if certain works from the search result have been processed already or not. The reason for documenting in the case of this thesis is therefore not repeatability of the systematic review but just verifiability during the writing of the thesis. The documentation is therefore not included in the thesis. For documentation only the title has been recorded together with the decision and an optional comment explaining the choice of including the document or not. For the search, no restriction has been made on the publication year. The following additional topic-based exclusion criteria have been used:

- Focus of the work is a machine learning network, a topic of information security like attack detection
- Focus of the work is a specific type of network only e.g. Internet of Things
- Focus of the work is path selection or path quality measures

The existing data available for analysis have been used together with data collected in the context of this study. The data analyzed has been considered to come from a field experiment as the scans from which the data stem have not been conducted in a controlled environment but in the Internet. In order to evaluate the validity, the choice of origin nodes will be analyzed. Multiple factors are known to be relevant for the choice. For the purpose of evaluating the validity, the following factors have been considered:

- Geographical location, proximity to other VPs
- If the VP is often used for scanning
- Topological location - wether the source node is situated in an area with high traffic loads or not
- Are there certain policies applied to the traffic affecting the source node or network - i.e. censoring, load-balancing.

2.2 Structure of the work

The remaining part of this work is structured as follows. Chapter 3 explains the fundamentals necessary in order to understand the following parts of the work. The chapter covers the necessary theory of topology measurement itself as well as the technical foundations such as, for example, IP, Border Gateway Protocol (BGP), and MPLS. Chapter 4 gives an overview of the state of the art in Internet topology discovery. Chapter 5 explains the collection, processing, and analysis of the existing data provided by Cooperative Association for Internet Data Analysis (CAIDA) [17]. Chapter 6 evaluates and discusses the results obtained. The chapter concludes with recommendations regarding the number and characteristics of VPs. Chapter 7 summarizes the previous findings and, based on those, draws conclusions about the research question and hypothesis.

CHAPTER 3

Background

This chapter provides the background information needed to understand the subsequent chapters. Explanations about topology measurements are given as well as summaries of different protocols used for communication on the Internet. Only specifications not deprecated are included in the explanations.

3.1 Network foundations

As the research topic has a lot to do with network technologies, this chapter provides an overview of protocols that are, for example, used for routing, data transfer of different data, address schemes and name resolution.

3.1.1 Domain Name System

Domain Name System (DNS) is a query response protocol whose messages have the same format in both directions. There is no single commonly agreed on definition of DNS. It can be considered as a combination of the following:

- DNS is a commonly used naming scheme and system for objects on the Internet.
- DNS is a distributed database for names and properties of those objects.

Global DNS

What follows is the explanation of some important concepts of the DNS, especially the global DNS. A **label** in this context is an ordered list of possibly multiple octets that have up to 63 octets in length. The root label is the only one with length 0. A label identifies one node in a network.

A **domain name** is an ordered list of one or more labels. A domain name where the last label identifies the root label of the network is fully qualified.

The following three name formats exist:

- The **wire format** is a list of labels ordered by decreasing the distance to the root, with the root label in the last position. Each label is preceded by a length octet.
- The **presentation format** is a list of labels ordered as in the wire format. The labels are encoded as American Standard Code for Information Interchange (ASCII), delimited by a ' . ' (dot) character. The presentation format includes the root label and the associated delimiter. An example is: *example.com.*
- The **common display format** is like the presentation format with the difference that the root label and the associated rightmost delimiter are optional. An example is: *example.com*
  Note the missing of the visible dot character in the second example in contrast to the previous one.

A domain name can have zero or multiple resource records associated with it. Every name being published in the DNS appears as a set of resource records.

All nodes having the same distance to the root are said to be in the same **zone**. The Top-Level Domain (TLD) is the zone one level below the root, such as *at, com, de, net.*

A domain contained within another domain is called **subdomain**. To give an example: *sub.example.com* is a subdomain of *example.com.*

The administration of domain names is done by delegation. The root zone is administered by Internet Corporation for Assigned Names and Numbers (ICANN). Administration policies may differ for different zones. An important use case of DNS is to resolve domain names in addresses. This process is called *forward lookup.* Depending on the zone, it is also possible to obtain a name by using its address. This process is called *reverse DNS lookup*, or just *reverse lookup.*

Private DNS

Names that use the protocol defined by [18] but do not rely on the root DNS zone or names that are not available on the Internet are said to be part of the private DNS. Systems may use the global and one or multiple private DNS systems. Domain names not appearing in the DNS and with the intention never to be looked up using the DNS protocol are not part of a global or private DNS.

Multicast DNS

The Multicast Domain Name System (mDNS) provides the ability to perform DNS-like operations on the local link without the need for any conventional unicast DNS server.

The mDNS provides a portion of the domain name namespace for local use without any fees, the need for delegations, or a conventional DNS server [19].

3.1.2 Internet Protocol

The IP provides users of the protocol logic for transmitting blocks of data called *Internet datagrams* from sources to destinations. The protocol also supports fragmentation and reassembly of packets in case it is necessary or desired that the packets are transported in smaller parts. The IP resides on a level where it is invoked by higher level protocols such as the Transmission Control Protocol (TCP) or the Uniform Data Protocol (UDP) discussed in the chapters 3.1.6 and 3.1.5. The IP itself invokes a local *net module* to transmit packets within a local network. The basic functions of the IP are addressing and fragmentation. Every host engaged in Internet communication has an Internet module which shares common rules for parameter interpretation. The IP uses the following key mechanisms in providing its service:

- The **type of service** indicates the desired quality of service. This includes parameters such as reliability and the importance of speed. Depending on the chosen parameters, the packets are treated differently during transmission throughout the Internet system.
- The **Time-to-Live (TTL)** allows the specification of an upper bound of a packet's lifetime measured in seconds. Every node forwarding the datagram has to decrement the TTL value by at least 1 unit even if less than 1 second passes during processing at this node. The reason for maintaining this information is to be able to discard undeliverable packages or those taking too long.
- The **options** mechanism provides a control for functions that are useful in some situations but not needed in most cases. This includes security options, special routing information, and the possibility of recording the information of the route taken.
- The **Header checksum** mechanism provides a means to verify that the information used to process the Internet datagram has been transmitted correctly.
- The **Identification** header field is a value of 16 bits length. The sender sets it to allow for the assembly of datagram fragments.

The IP does not provide a reliable communication facility. Errors during communication can be reported through Internet Control Message Protocol (ICMP) being implemented in the Internet module. ICMP is discussed in Chapter 3.1.4. The IP mainly deals with addresses. The Internet module maps Internet addresses to local net addresses. The IP has to support the use case of one host having multiple physical interfaces to a network where each interface has a different logical Internet address. Fragmentation occurs when

a packet is too large for transport through the network. Versions 4 and 6 of the IP handle fragmentation in different ways. More information about the different versions is given in the chapters 3.1.2 and 3.1.2. Datagrams between networks are forwarded by gateways. Gateways do not need to implement higher-level protocols such as Hypertext Transfer Protocol (HTTP) but only the IP extended with the functionality necessary for exchanging routing information with other networks [20].

Version 4

Version 4 of the IP, also denoted by IPv4, specifies that the source and destination are identified by fixed-length addresses of 32 bits [20]. Fragmentation may be performed by any node in the path when necessary as long as the node has enough resources. The Internet options specified by IPv4, estimated to be the most relevant for the following topics of this work, are the following:

- Record Route: This option is used to trace the route that an Internet datagram takes as long as there is enough space in the IP header.
- Internet timestamp: The timestamp of every node on the taken path is recorded as long as there is enough space.
- Loose Source and Record Route (LSRR): This option allows senders to provide routing information to the gateways, which can treat the routing information as a suggestion. The path gets recorded as with the formerly mentioned Record Route option.
- Strict Source and Record Route (SSRR): This option is similar to LSRR with the difference that the gateway or host IP must send the datagram directly to the next address of the source route [20]

Classless Inter-domain Routing

Classless Inter-domain Routing (CIDR) is an address assignment strategy in the context of the IPv4 address space. The strategy used before CIDR was to have three classes of networks that could be assigned to organizations. Depending on the class (A, B, or C), a different number of address bits is fixed. Class A addresses have the first Most significant bit (MSB) set to *0* (MSB) and leave the remaining 7 MSBs to identify 128 networks. An organization with such an address block assigned can then further split this address block into smaller parts or use the addresses directly. Each of the class A networks has 16 777 216 addresses that can be assigned. A class B address has the two MSBs set to *10* followed by 14 bits for the network identifier.

The class-based addressing scheme has the following disadvantages:

1. The flexibility of splitting the address block was quite limited as only class B and class C ranges can be used by organizations where either 16 or 24 of the MSBs

were fixed. In particular, the strategy lacked a suitable address block size for organizations of medium size somewhere between 254 and 65 534 addresses.

2. The size of routing tables in Internet routers has grown rapidly along with the expansion of Internet hosts.

3. The probable exhaustion of the available IP v4 addresses in the early 1990s.

CIDR solved the first and second problem and reduced the consumption rate of the IPv4 addresses. The approach is that the assigned prefixes should roughly follow the underlying Internet topology. The approach has the ability to fix from 0 to 32 of the most significant bits of the IPv4 address. Those fixed bits are called the *prefix*, which is written in the same format as an IPv4 address. The prefix is followed by the '/' character as well as a decimal number indicating the prefix length.

To give an example, a class B address block is used: The IPv4 address is *172.16.0.0* and the CIDR notation specifies this address block as *172.16.0.0/16*. It also makes sense to look at the binary representation: *10101100.00010000.00000000.00000000*

Internet Assigned Numbers Authority (IANA) allocates free /8 prefix address blocks to Regional Internet Registry (RIR)s which then assign parts of those blocks to Local Internet Registry (LIR)s which then directly use them or reassign smaller portions of the block. Depending on the size of the LIR, different prefix sizes are used [21].

Version 6

Version 6 of the IP, also denoted by IPv6 uses 128 bits for addressing instead of 32 [22]. The address scheme IPv6 has the following three address types:

- A **unicast address** identifies a single interface. It might be assigned to multiple physical interfaces, which is useful for load balancing. A unicast address consists of a subnet prefix and an interface ID. The interface ID must be unique for at least the given subnet prefix. Unicast addresses can have global or local scopes denoted by a flag in the address. There are two special types of unicast addresses: the *unspecified address* representing the absence of an address and the *loopback address* identifying the node itself. Both must not be assigned to any physical interface. An IPv6 unicast address may have an embedded IPv4 address.

- An **anycast address** identifies a set of interfaces that may belong to different nodes. The packets with an anycast address as destination are sent to one of the specified interfaces. Depending on the applied routing protocol, for example, the nearest node might be taken. Syntactically, anycast addresses are not distinguishable from unicast addresses. The interfaces assigned to the unicast address have to be explicitly configured to know that it is an anycast address. Global anycast

addresses are technically possible, though not ideal for scaling. The reason is that a separate routing entry has to be present in the routing table for the network where the anycast address should take effect, which in this case would be the whole Internet. Support for this use case may therefore be restricted.

- A **multicast address** identifies a set of interfaces where packets are distributed to all included interfaces. The multicast addresses are identified by the 8 MSBs set to *0*. The 8 following bits are flags that, for example, indicate if the multicast address is permanently assigned or not, followed by the scope.

IPv6 addresses are assigned to interfaces, while a single interface may have multiple IPv6 addresses [23]. In IPv6 the TTL header field has been renamed to *hop limit* which is no longer time-related. IPv6 offers the *routing header* which is similar to the 3.1.2 LSRR option. In contrast to 3.1.2, the fragmentation is only done by the source node [22]. The IPv6 specification recommends that nodes implement the *Path Maximum Transmission unit (PMTU) Discovery protocol* defined by [24]. The PMTU is the maximum size that a packet can have for transmission on a given path. It is equal to the minimum of the PMTUs of all nodes on the given path. The *PMTU Discovery protocol* defines mechanisms that allow a node to discover the PMTU of any path [24].

3.1.3 IP forwarding

IP forwarding is a mechanism in which each router chooses the next hop for the current packet based on the header analysis and the results of the routing algorithm. For choosing the next hop, two functions are used:

1. A function that partitions a set of possible packets into a set of Forwarding Equivalence Class (FEC)s.
2. A function that maps each FEC to the next hop. The router will usually consider two packets to be in the same FEC if there is an address prefix X in the routing tables of the router such that X has the longest match for each packet's destination address. Those steps are performed on every hop [25].

3.1.4 Internet Control Message Protocol

ICMP messages are used to provide feedback about problems in the network. However, this protocol is not used to improve the reliability of IP. No ICMP messages are sent about the ICMP messages themselves. This means that the feedback might not reach its destination nodes. The ICMP also includes a TTL header field as ICMP messages use the basic IP header. An example of a use case for an ICMP message, namely the *Time Exceeded Message*, is that a gateway receives a datagram containing a TTL value of 0. In this case, the node must discard the datagram as specified by the 3.1.2. The gateway may send an ICMP message to the source node. A gateway may also discard Internet

datagrams if it does not have the capacity to process the packet. The gateway would notify the source node using a *Source Quench Message* in such a case. Gateways may also send those messages when approaching the capacity limit, meaning that the packet may nevertheless be delivered. The protocol further specifies messages for verifying if a host is responsive using the *Echo and Echo Reply Messages* [26].

3.1.5 Uniform Data Protocol

The UDP is a protocol with minimal overhead and uses the IP as an underlying protocol. It is transaction-oriented and does not provide mechanisms for ensuring the message delivery or duplicate avoidance. It has a checksum that includes information from the IP and UDP header and the data itself. The checksum allows nodes to perform data integrity checks [27]. The UDP does not establish end-to-end connections between the source and destination hosts. In addition, state handling is kept to a minimum, making UDP a very efficient data transport protocol [28].

3.1.6 Transmission Control Protocol

The TCP in contrast to the 3.1.5 provides a reliable in-order bytestream service to applications. Each TCP segment is sent as an IP datagram. The TCP detects packet loss and errors and handles them by retransmission. The TCP is connection oriented, which means that an end-to-end connection is established between the source and the destination host. The data flow is supported biderectionally. This protocol uses port numbers to identify application services. Similar to other protocols it offers some flags, also known as *control bits* of which a subset is needed to handle the different states of the connection lifetime. A TCP connection requires maintaining the state of several variables. They are stored in a so-called Transmission Control Block (TCB) that contains local and remote IP addresses, port numbers, and more. A TCP connection progresses from one state to another in response to events such as user calls, incoming segments, particularly those containing the flags *SYN, ACK, RST, and FIN* and timeouts. The *OPEN* call specifies whether the connection establishment should be actively pursued or passively awaited. The *RST* flag can be sent from any state and resets the connection. This usually occurs on different errors, such as an incoming segment, when a connection is in the *CLOSED* state. Every octet of data sent over TCP has a sequence number. Each of those octets can be acknowledged. The acknowledgment mechanism is cumulative, meaning that all octets up to the acknowledged one (excluding) are indicated to have been received. This leads to a straightforward duplicate detection. A connection is defined by a pair of sockets and can be reused. In order to identify duplicate segments from previous connections, the implementation of the TCP does the following: it defines a *TIME-WAIT* state to limit the rate of connection reuse. The initial sequence numbers are generated from a monotonically increasing sequence, which should increase roughly every four milliseconds. This sequence is called *microsecond timer*. The state of the sequence does not need to be persisted across reboots of the host machine. The value of the microsecond timer is added to the value of a Pseudorandom function (PRF) taking

as input the IP addresses, port values, and a secret key. The value of the PRF must not be computable from the outside. The connection establishment is done through a three-way handshake [29].

3.1.7 Border Gateway Protocol

The BGP is an inter-AS routing protocol. It is used for the exchange of network reachability information between different ASs. The BGP only supports destination-based forwarding for which the router looks at the IP header. This means that BGP does not support specifying a fixed route a packet should take. The BGP uses the TCP over port 179. Updates are communicated as soon as the routing table changes. Therefore, periodic checks are not necessary. If an error occurs during a connection, a *NOTIFICATION* message is sent, and the connection is closed. An AS has one or multiple BGP speakers. The BGP used within an AS is called Interior Gateway Protocol (IGP) and provides a consistent view of the interior routes of ASs even in the case of multiple BGP speakers. The routes between speakers are communicated via *UPDATE* messages. Routes are stored in Routing Information Base (RIB)s. of which three different parts or types are distinguished:

- The **Adj-RIBs-In** contains all incoming route information.
- The **Loc-RIB** contains the routes selected by applying its local policies to the former part of the RIB. The next hop for each of the routes in this part have to be resolvable via the local speaker's routing table.
- The **Adj-RIBs-Out** contains all the routes the speaker selected for communication with other BGP speakers.

The routing information that the speaker uses to forward packets is kept in the routing table. It accumulates routes to directly connected networks, static routes, and those learned from the IGP and the BGP. How the different parts of the RIB are implemented and which routes are selected for application is not specified by the protocol, but is a local policy decision. The information communicated through *UPDATE* messages can be used to construct a graph that can then be analyzed to detect routing information loops and other routing anomalies in the AS.

In order to withdraw a route, a speaker can:

1. Close the connection, which causes all routes that have been communicated between connected speakers to be removed from service.
2. Send a replacement route in an *UPDATE* message with the same Network Reachability Information (NRLI).
3. Send an *UPDATE* message with the IP prefix for the destination contained in a *WITHDRAWN ROUTES* field [25].

3.1.8 Multiprotocol Label Switching

In contrast to 3.1.3, the assignment of a packet to a FEC is done only once. Subsequent hops don't analyze the network layer anymore. The assignment of a packet to a FEC is done based on a label. Compared to the classic IP forwarding explained in Chapter 3.1.3, MPLS has the following advantages:

- Forwarding can be done by switches that can perform label lookups and switching but are not able to analyze the network layer headers.
- The router that assigns the label can use any information it has about a packet. This router is the ingress router that is usually at the entry point of the network. An example of such information that can be used in the case of MPLS but not with IP forwarding is the port number.
- It provides the ability to label packets differently depending on which ingress router is used, leading to forwarding decisions that depend on the ingress router.
- Packets can be forced to take particular paths chosen before or when entering the network. MPLS forwarding does this using a label that represents the route instead of encoding the route together with the packet.

A router that supports MPLS is called Label Switching Router (LSR). The label is a short identifier for a FEC. Labels are usually downstream-assigned and labels distributed from downstream to upstream routers. When two routers A and B decide to add the label L to a packet flowing from A to B, B is called the downstream LSR and A the upstream LSR. The labels of a packet are managed in a label stack. Different label distribution protocols (extending existing ones) exist for propagation of the labels within a network. As in conventional 3.1.3, TTL is used to suppress forwarding loops. It can also be used to implement other functionality such as limiting the scope of a packet [30]. MPLS maintains a separate TTL value called *Label stack entry (LSE)-TTL*. This value is encapsulated in the MPLS header. The way in which the LSE-TTL value is handled varies:

- The TTL value is taken from the network layer on the ingress router and decremented before encapsulating it in the MPLS header.
- The ingress router assigns a random TTL value.

In both cases, the value gets decremented on every hop. At the egress router the packet has two TTL values which will likely be different. MPLS needs to copy one of them in the IP header before forwarding the packet as a normal IP packet. In order to ensure that the outgoing TTL value is not greater than the incoming one, the minimum of both TTL values is taken at the egress router [31].

Sometimes, a router A decides to route a packet to router C which is not an adjacent router of A on the path for the packet and C is not the final destination. This can be done by encapsulating the packet inside a network layer packet that has the destination address C. This creates a *tunnel* from A to C. If the packet follows the usual path from A to C it is called a *Hop-by-Hop Routed Tunnel*. If the packet takes another path, it is called an *Explicitly Routed Tunnel*. Tunnels can also be implemented using a Label Switching Path (LSP). In the latter case, label switching is used instead of network layer encapsulation. [30]

3.2 Load balancing

Load balancing is used to improve the reliability and utilization of resources of a service [32]. Most commercial routers have load balancing capabilities and the common routers (like those of router vendors Cisco or Juniper) support all of the three types of load balancing mentioned below. Between any given pair of PoPs, usually several link-disjoint and PoP-disjoint paths are observed. Link-disjoint paths are paths in which no shared link is present. Typical Internet Service Provider (ISP)s build redundancy into their infrastructure. An ISP partitions traffic across multiple links, which leads to a diversity in the paths discovered by the scanning tools. Commercial interests guide today's Internet routing policies in ways that often yield inferior end-to-end paths as measured by delay, loss rate, or bandwidth.

The following types of load balancing are distinguished:

- **Per-destination**: Network packets are classified based on the destination address. This is the most common form of load balancing. [33] observed 70% of their 771 795 source-destination pairs to traverse a per-destination load balancer.
- **Per-flow**: Network packets are classified based on the following fields in the IP and UDP/TCP packet headers: IP source and destination address, protocol, source and destination port. Varying any field in the first four octets of the transport-layer header leads to a change of the flow identifier of the different packets. 39% of 1 010 256 source destination pairs had at least one load balancer in the path. The size of the datasets used by [33] for analyzing the per-destination and per-flow load balancing behavior differ. This is due to the fact that the per-destination dataset has 11 sources in contrast to 15 for the per-flow dataset.
- **Per-packet**: The packets are distributed evenly over multiple connections. 1,9% of the source destination pairs had at least one of such load balancers in the path. The same data set as for the per-flow analysis was used for the per-packet analysis. Large ISPs usually avoid this type of load balancing as it could have a negative impact on TCP connections. Most per-flow load balancers affect only a small share of the paths because they are located close to the destination. In the analyzed

set 85% of the per-flow load balancers were located at a maximum of 2 hops from destination.

There is a large disparity between a comparably small number of load balancers and a large number of load-balanced paths. In the case of per-flow and per-packet load balancing, the 50 load balancers that occur in paths the most often affect at least 78% of paths with load balancing observed by one VP. In the case of per-packet load balancing, a maximum of almost 60% is reached.

Furthermore, intra- and inter-domain load balancing can be distinguished: the former one only affects internal routing decisions of the respective AS. In this case the paths diverge after entering the AS and converge before exiting it. This behavior is realized through intra-domain routes in the forwarding tables of the routers. The BGP (being the protocol used for inter-domain routing) does not allow a router to install more than one next hop for a destination prefix. However, router vendors Juniper and Cisco offer BGP-multipath capabilities [33].

3.3 Topology measurements

The term *topology* refers to the edge-based and structural attributes of a network or a graph (Zaki and Meira of [34]) - the various entities, such as routers and their interconnections [34]. Although some information can be retrieved from passive measurements, active measurement approaches are used more often. Much of the Internet is hidden and unknown, as it is a large and complex system not managed by a single authoritative entity [1]. Internet topology measurements serve, among others, the purpose of gaining insight into the following topics: available services, outages, security issues, host availability, ISP behavior, and restrictions due to censorship. Often more than one VP is used for scanning. This usually means that multiple VPs simultaneously scan the same destination set [2]. Chapter 4.2 gives more information on the different possibilities for selecting VPs. The affects of measures such as Traffic Engineering or the usage of MPLS should be taken into account and dealt with when interpreting the data and especially when comparing and evaluating the performance of a routing method [35]. It has been shown that the past scanning behavior can influence the coverage of a VP [2]. The network graph can be analyzed at different granularities. Often the following are used:

- The **link layer topology** reflects the physical connections between different nodes. It is a prerequisite for tasks such as network diagnostics and resource management.
- The **Internet topology** reflects the logical connections between the different nodes. It can be analyzed itself at the following levels:
  - The **IP interface layer** directly models the IP addresses of the routers and destinations. The data is obtained by tools such as traceroute.

  - The **router layer** can be created from the IP interface layer by aggregating IP addresses of the same router. This step, called *alias resolution* is not straightforward as router information is not provided by routers or the respective networks themselves. It has to be inferred using heuristics and is as such best effort.
  - The **PoP layer** topology aggregates routers per AS per geographical location such as a city. It can be obtained by using reverse DNS lookups of IP addresses of the IP interface layer topology.
  - The **AS layer** models the connections between different ASs. It cannot be created directly from measurements, but from looking up routing information address databases [1].

- The **overlay topology**: Overlay networks are virtual networks built on top existing networks. Often, the IP interface layer is taken as a basis. End-to-end hosts exchange measurement results in order to decide on the path of traffic they want to send [36].

This work focuses on the IP interface and router layers of the Internet topology. Measurement tools are confrontend with numerous challenges some of which are:

- Many protocols are not applied consistently all the time. A specific example is ICMP. ISPs filter ICMP probes using rate limiters for security and performance reasons.
- ISPs are subject to frequent routing changes [35].

An introduction to some common measurement approaches is given in chapter 4.1.

3.4 Graphs

A graph is a type of data structure that models interactions between a collection of agents [37]. It is a foundation for the mathematical representation of different types of complex systems [38].

3.4.1 Notation

As different graph notations are used by different resources, this chapter specifies the notation used throughout this work. The notation used here is close to that used by [37].

The definition of a graph G is given by its sets of vertex and edges. The terms vertex and node are used interchangeably in the rest of this work. A graph G is defined as: $G = (V, E)$, vertex set $V = 1, ..., n$ and edge set $E \subset V \times V$. For measuring the size of a graph, there are many different options. Common measures are the number of vertices

$n = |V|$ and the number of edges $m \stackrel{\text{def}}{=} |E|$. The degree d_i of a vertex i is the number of edges incident (connected) to node i.

For $i \in V$ and $j \in V$, $i \sim j$ indicates that nodes i and j are adjacent, which is the case if $(i, j) \in E$. The matrix A is called the *adjacency matrix* and is defined as

$$A_{i,j} \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } i \sim j \\ 0 & \text{otherwise} \end{cases}$$

The *degree matrix* D is defined as the diagonal matrix of degrees, which means that $D_{i,i} = d_i$ and $D_{i,j} = 0$ for $i \neq j$. The *Laplacian matrix* L is then defined as $D - A$ [37]. The eigenvector v is defined by the eigenvalue equation: $\lambda v = Av$. λ is a scalar, called the *eigenvalue* [39]. The sorted sequence of a matrix's eigenvalues is called *spectrum*, whereas the sorting direction depends on the use case. The *adjacency spectrum* is the spectrum of matrix A is denoted as $\lambda_1^A \geq \lambda_2^A \geq \lambda_k^A \geq ... \geq \lambda_n^A$. In this case, the eigenvector consists of n sorted (descending order in the case of the adjacency spectrum) eigenvalues [37].

3.4.2 Community structure

Community structure is the tendency of vertices to build groups. The vertices within the groups are densely connected, whereas the connections between the different groups are sparse. This is a common pattern for many networks, including information networks such as the Internet. Depending on the use case, it can be important to analyze the community structure of a graph, as the properties on the community level can be different from those of the entire network. Ignoring the community structure might result in the loss of possibly relevant features. An example is a network that models sexual contacts with different communities that have high- and low-activity individuals. Analyzing the measures per community provides more possibilities with regard to the features and the questions that can be answered than when only the overall measures are used. In addition, the position of a node has been shown to possibly influence its assumed function. Communities in a network can be used to form a higher-level meta-network [38].

3.4.3 Traversal types

Depending on the graph's use case and it's context, different graph traversal patterns can be observed. The following different flows of things traversing nodes of the graph are distinguished here (this is not necessarily an exhaustive enumeration):

- A **walk** is defined as an unrestricted sequence of incident links. An example of this flow type is money, where coins and bills can have arbitrary link sequences.
- A **trail** is defined as multiple incident links with all links occurring only once. Every trail is implicitly also a walk. An example of this flow type is the case of a paperback novel where each physical object can only be at one place at a time.

The printed novel object might change owner from time to time. An owner might possess it more than once (receiving it from different persons), but not via the same link (unless the book is read more than once or lent to other people). Another example could be email messages warning of a computer virus. One person might receive the same email multiple times (technically speaking, they are still separate emails with the same content), the sender will usually be different, though. In contrast to the former example of trails, objects propagate by replication instead of being transferred from one node to another.

- A **path** is defined as multiple incident links where the links and nodes are unique. Every path is also implicitly also a trail. An example is a postal package delivery process where the shortest path is usually known. The shortest path is called the *geodesic path*.

Different characteristics can be used to further classify graphs:

- Mechanics of dyadic diffusion - wether the the items are replicated or transferred
- If the propagation of the items happens in a serial (gossip via one-to-one is one example) or parallel (a television show is an example) manner. This distinction can only be made for replication-based flows.
- If the path is chosen on a deterministic basis or if it is chosen randomly. A typical example for the deterministic type is to take the best way. What the best means depends on the context.
- If the item flows even follow graph-theoretic walks, trails, or paths [39].

Another characteristic that one can look at is whether a graph is sparse or dense. A sparse graph is one where the number of egdes grows linearly with an increasing number of nodes. One can also distinguish between directed and undirected graphs, as well as if the edges have weights assigned or not [37]. Only undirected, unweighted graphs are considered/used in this work. Furthermore, the graphs do not contain self-loops (connections from a node to itself).

3.4.4 Coverage

Coverage usually refers to the ratio between a complete graph and a subgraph. The coverage refers to the nodes or the edges, and its definition is given below. Let the complete graph be given by $G_1 = (V_1, E_1)$ and the subgraph by $G_2 = (V_2, E_2)$. The node coverage is given by $\frac{V_1}{V_2}$ and the edge coverage by $\frac{E_1}{E_2}$. The ratio is a number in the interval [0, 1] as long as G_1 is complete. Complete in this context means that the graph contains all nodes and edges of the modeled network. If the graph G_2 contains nodes not present in G_1 the fraction can be greater than one [2].

3.4.5 Graph comparison

Many possibilities of comparing graphs exist. One can compare the graphs' characteristics mentioned above. Only pairwise graph comparison methods are included in the remainder of the work. As graphs often have 10^4 to 10^8 vertices, the performance of the comparison methods plays a crucial role. When comparing graphs of these sizes or larger, algorithms whose runtime scales linearly or near-linearly with the number of vertices are considered to have a feasible runtime [37]. Usually, there is a trade-off between computational efficiency, effectiveness of the results, and the interpretability. Graph comparison methods measuring the similarity of graphs, root on the *graph isomorphism problem*. This problem specifies that two (undirected and unweighted) graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ are said to be isomorphic if there is a one-to-one mapping Φ from the node set V_1 to V_2. The mapping must satisfy the following condition: the edge $(u, v) \in E_1$ if and only if the edge $(\Phi(u), \Phi(v)) \in E_2$. This graph comparison is an *exact graph matching* method leading to a binary outcome only evaluating to true if the graphs have exactly the same structure. Graph isomorphism is not the same as equality as two graphs can be isomorph but have different vertex and edge labels. It has proven to be more useful to also have *inexact graph matching* methods. Often a real-valued distance metric approaching 0 when the graphs approach isomorphism is used [40]. Those measures can be split into two categories: spectral distances and matrix distances. One measure of the first category is the Adjacency Spectral Distance (ASD) being defined as follows:

$$d_A(G, G^*) \stackrel{\text{def}}{=} \sqrt{\sum_{n=1}^{n} (\lambda_1^A - \lambda_1^{A^*})^2}$$

Intuitively speaking, this measure calculates the pairwise distances between the eigenvalues of two adjacency matrices. It is the distance between the two spectra in the l_2 metric. Different l_p metrics can be used while p is in the range $[0, \infty]$. The larger p, the larger the impact of outliers. Spectral distances can also be defined on different graph matrices. Oftentimes when comparing graphs only the first k eigenvalues are compared. In the case of the adjacency spectral distance, the k largest eigenvalues are used. The choice of k affects the comparison, result, and interpretation in the following way: the larger k, the more information about the local structure is used. With small k values, the local structure is mostly ignored, only focusing on the community structure. The interpretability of the eigenvalues' distribution depends on the matrix representation. In the case of the Laplacian matrix, the following holds: the multiplicity of the zero eigenvalue equals the number of connected components of the graph. That means that if $0 = \lambda_k^L < \lambda_{k+1}^L$ then there exist exactly k connected graph components. Based on the Laplacian matrix, the Normalized Laplacian Spectral Distance (NLSD) and Laplacian Spectral Distance (LSD) can be defined in a similar way. The NLSD has the property that it can be used on graphs of different sizes. In contrast to the ASD, the LSDs compare the k smallest eigenvalues.

Matrix distances use pairwise affinities between vertices in a graph. The affinities are often put in matrices which then get compaired (for instance, the entry-wise l_p norm). All matrix distance measures require *node correspondence* [37]. Node correspondence, sometimes also referred to as Known Node-Correspondence (KNC), means that the set of vertices of two graphs are the same (or at least have a common subset) and the pairwise node correspondence is known to be true. However, only the common subset fulfills the node correspondence criterion, which might make the distance measure less usable if the common subset's coverage is low.

A distance measure between two vertices $v_1, v_2 \in V$ of a single graph is denoted as $\delta(v_1, v_2)$. Examples are the shortest path distance, effective graph distance, and variations of the random-walk distances. Let a generic distance on a graph be denoted by $\delta : V \times V \to \mathbb{R}$

Given two matrices M_1 and M_2 of pairwise distances of the vertices within the graphs $G_1 and G_2$. The matrix distance of the two graphs is then defined as

$$d(G_1, G_2) \stackrel{\text{def}}{=} ||M_1 - M_2||$$

The usefulness of the distance depends on the matrices used. How well the matrices reflect the topological structure of the graph has a large impact on the further use. Some distances detect changes in the volume of the graph, which is typically not desired. In contrast, the resistance distance detects connectivity changes between graphs. The Fast belief propagation models the diffusion of information throughout a graph. In theory, it is capable of perceiving global and local structure, but no comparison of this metric with alternatives has been made [37].

Another important graph concept not mentioned so far is the centrality. It is one of the concepts the most appearing in research in the context of social network analysis. Different concrete measures for this concept exist: degree centrality, closeness, betweenness, eigenvector centrality, information centrality, flow betweenness, rush index, etc. They have implicit assumptions about how the things within the graph flow. Some make the assumption that the things flowing are indivisible like packets, while other measures assume that the things can take multiple paths at the same time. Making the wrong assumptions for a measure or applying a measure in the wrong context can lead to a loss of at least some of the measure's interpretability or lead to poor answers. Therefore, it is necessary to also know the assumptions of a measure. In the following, some centrality measures are briefly discussed.

The **node's closeness centrality** is the sum of the graph-theoretic distances from all other nodes. Which distance measure to use depends on the use case. Examples for such distance measures are as follows.

- The number of links of the shortest path from one node to another

- The expected travel time between two nodes

If the traffic takes the shortest path, the closeness centrality can be interpreted as an index of the expected time until arrival. In simultaneous settings the traffic can take all possible paths which includes the shortest one. In this case, it also makes sense to use the closeness centrality measure.

The **betweenness centrality** for a node v_1 is defined as follows: share of the times a node v_2 needs the node v_1 to reach another node v_3 taking the shortest path. It represents the exclusivity of v_1. The assumptions of this measure are that the traffic is indivisible. When choosing between different equally short paths, it does so randomly. Traffic travels only along the shortest paths. The characteristics match the package delivery process but not flows of gossip, for instance.

The **eigenvector centrality** is defined as the principal eigenvector of the adjacency matrix. The interpretation is that a node that has a high eigenvector score is adjacent to other nodes that also have high scores. The eigenvector centrality provides a model of nodal risk. It is a function of the risk level of its contacts. The cells of the matrix powers give the number of walks of length k between two nodes. The eigenvector centrality measure assumes the traffic to be able to move via unrestricted walks without the restrictions of trails, paths, or a geodesic path. There is no assumption that the things flow to a neighbor one at a time but rather by parallel duplication. The measure is ideally suited for influence type processes such as when a speaker changes the attitude of many other persons at the same time.

The **degree centrality** measures the number of ties of a node. It is equal to the sum of each row of the adjacency matrix. It is similar to the eigenvector centrality with the difference that the latter measures long-term direct and indirect risk, while the first measures immediate risk only.

Yet another measure is the degree of a node on its own. It models the frequency of visits of the modeled entity taking an infinitely long random walk through the network. An example in which it makes sense to use this measure is the money exchange process [39].

The last measure mentioned here is the **connectivity**. The following two types can be distinguished:

- Edge connectivity: It is defined as the minimum number of edges whose removal would result in a loss of connectivity of the graph G. It is denoted by $e(G)$.
- Vertex connectivity: It is defined similarly to the edge connectivity. The difference is that the vertices and edges are removed together. It is denoted by $v(G)$ [41].

CHAPTER 4

Related work

When performing Internet topology measurements, one is confronted with the situation that the measurement targets are plentiful while the number of sources is relatively scarce. Each path found by traceroute (see 4.1.1) forms a directed graph. The various graphs are then combined. Often, it is desired to relate different interfaces of the same router to this router itself. The core part of nodes is comparably small and is used by most of the end-to-end (source-target) paths. A network graph is clearly not a random network, but consists of two distinct parts: the central routing core and the links that feed into the backbone nodes [2].

[2] used data from CAIDA Skitter project [42]. The data period analyzed is May 2000 and includes measurement sources in North America, Europe, and Asia. It contains 8 sources and 1277 destinations. Four types of nodes are distinguished: leaf, stub, border, and backbone. Half of the discovered nodes are backbone nodes, less than 10% border nodes.

[2] tries to build an undirected graph and defines node and edge coverage (p. 7). [2] analyzes the benefit of conducting additional measurements. This is done using the expected coverage of the resulting subgraph. [2] mentions that some regions are difficult to cover using traceroute. [2] mentions **stub** domains which are the domains where traffic ends or originates. [2] further mentions the **transit** domains as being part of the highly connected backbone. Routes with non-responding destinations have been discarded, while routes with hosts that did not respond to ICMP requests have been included. Further classification of nodes that correspond to routers and internet hosts can be done: core muters, border routers, stub routers, and leaf nodes (p. 9). The authors were primarily interested in characterizing the backbone. Unfortunately, the total coverage cannot be measured, as the complete graph is not known. Additional measurements might have good coverage, but do not necessarily uncover unknown parts of the graph. Almost 20% of the overall nodes that have been found can be classified as backbone nodes. Those nodes used multiple interfaces with different IP-addresses. [2]

mentions the following approach: when transmitting an ICMP message, a router usually transmits the packet with the source address equal to the one of the outbound interface on which the packet is sent. 10% of the core routers never responded to UDP messages transmitted to unknown ports. If instead routers respond with the source address equal to the UDP destination address, the technique is useless. Many backbone nodes are misclassified as stub or border nodes with small number of sources. Once a node is classified as backbone, the classification is no longer changed. The order in which the sources are added has a significant impact on the overall results. The behavior of routing algorithms on the Internet is not deterministic. [2] observe rapidly diminishing returns for adding sources and constant returns when adding destinations. When adding new sources, mainly new backbone nodes and interfaces are uncovered. A good balance seems to have a large number of passive targets and a small number of active measurement points. The second data set consists of 12 sources and more than 313 000 destinations. The large size makes it more difficult to disambiguate interfaces. They define the marginal utility of experiment Sn (p. 14) both online and offline (not biased by ordering) and distinguish experiments where sources are added vs destinations. Conclusions sensitive to choice of source nodes.

4.1 Measurement approaches

Depending on the type of topology to be found, the desired information, and the subsequent use of the collected data, there are a variety of measurement approaches, while even more are still created targeting different issues of existing solutions [1].

4.1.1 Traceroute

Traceroute is a tool used to track the route packets take from a source to a destination host in an IP network. Traceroute uses an increasing value of the IP TTL header field and waits for ICMP *time exceeded* replies. It starts with a TTL value of one and increases the value until either traceroute reaches its maximum hop limit (being 30 per default), an ICMP *port unreachable* or a TCP *reset* message is received. The latter two messages signify that the destination host has been reached. By default, traceroute sends 3 UDP probes per TTL value. Table 4.1 shows the output of *tracert*, which is a similar tool to traceroute but available on the Windows Operating System (OS). Each row in the table refers to a different TTL value visible in the first column. The second to fourth column show the Round Trip Time (RTT) values of the three probes per TTL value. The last columns show the IP address, sometimes together with its associated name. If for a TTL value different probes are answered by different hosts, all of those hosts are listed in the last column. If no response is received after sending a probe within the configurable *waittime* of traceroute, which is 5 seconds per default, an asterisk is printed for that probe. This case is visible in the lines with TTL values of two and three. When all three probes remain without response, traceroute does not give any information about the hop. Instead, the "Request Timed Out" message is shown. If

TTL	RTT1	RTT2	RTT3	Hop IP / name
1	5 ms	3 ms	3 ms	172.20.10.1
2	*	*	*	Request timed out.
3	*	43 ms	27 ms	10.10.15.0.winrou1.mrsn.at [10.10.15.0]
4	78 ms	42 ms	47 ms	10.10.15.151
5	46 ms	38 ms	42 ms	google.peering.cz [185.0.20.170]
6	53 ms	39 ms	59 ms	108.170.245.33
7	60 ms	43 ms	47 ms	209.85.245.247
8	54 ms	39 ms	49 ms	dns.google [8.8.8.8]

Table 4.1: Tracert output from the IP address 62.240.134.144 to 8.8.8.8

traceroute does not receive any response for most of the probes, it will stop the execution. In order to reduce the likelihood that the destination host processes the probe, an unlikely value for the destination port is used. Alternatives for sending probes are ICMP *echo* requests and TCP *SYN* requests. Modern networking services such as firewalls often block communication using such unlikely UDP ports as well as ICMP *echo* requests. In order to speed up the work, traceroute sends multiple probes simultaneously per default, which may result in a loss of responses. To cope with this risk, the traceroute implementation provides the option to decrease or completely disable simultaneous probing [43]. The options provided by the tool differ by exact OS and version of the tool. Therefore, no further information concerning the options of traceroute is given in this work.

When using traceroute different topological anomalies might be observable. The main reason for those anomalies is load balancing (see Chapter 3.2). The following anomalies are distinguished:

- **Loops**: One or more nodes appear at least twice in at least one route from a source to a destination. This means that the same node appears with different TTL values in a Traceroute record. Possible reasons for loops other than load balancing are the following:
  - Probes with different TTL values take different routes due to path changes during the execution of traceroute leading to the same final host.
  - Zero-TTL forwarding: Misconfigured routers often forward packets that have a TTL value of zero while they should drop those packets instead. This leads to the case where a packet with a TTL value of n reaches the same host as a packet with a TTL value of n + 1 if the faulty router is the $(n+1)^th$ one.
  - Address rewriting: Gateway routers may replace the source address field of all ICMP packets leaving the subnetwork to which the gateway is attached by a single IP address.
  - *Unreachability* message: This case happens when a router is not able to forward probes for whatever reason. At an initial TTL value where this router is the

last node, it sends the expected ICMP *Time Exceeded* message. In subsequent probes with larger TTL values, the same node also answers, while it does so with an ICMP *Destination Unreachable* message.

[44] distinguishes the following types of loops: *information loop, forwarding loop, and traceroute loop.* The former type is the case where a router makes decisions based on routing information the router itself has propagated earlier. Forwarding loops occur when packets forwarded by a router return to this router later in the path. When a traceroute record contains a sequence of routers multiple times, it is called a traceroute loop.

- Cycles: A route is cyclic if at least one router R in the path appears twice. The occurrences of router R have to be separated by at least one occurrence of a different router. The possible causes for cycles are load balancing as well as an *unreachability* message as explained above. Furthermore, packets might be caught in a forwarding loop during the convergence of route changes.

- Diamonds: In contrast to loops and cycles, diamonds can only appear when multiple probes are used per TTL value. Diamonds refer to the case where routes discovered by different probes observe paths with at least one intermediate router. On the left part of the figure 4.1 one can see multiple different paths from nodes L to G. On the right side of the same figure, a possible outcome of traceroute is shown. Given the traceroute outcome depicted, examples of diamonds are the following pair of nodes: $(L_0, E_0), (L_0, G_0), (B_0, G_0)$. However, notice that the pair of nodes (C_0, G_0) is not a diamond as only one path has been observed from C_0 to G_0. As it may not be obvious how the path L_0, B_0, E_0 could be observed as A and E are on distinct paths on the left part of the figure 4.1 an explanation is given. When probing with a TTL value of six, the load balancer chooses to take the path that contains node A. In this case, A is the final round. In the next probing round the load balancer chooses to route the packets through the path containing node E where E is the final node for this probing round, resulting in the above-mentioned observed path. The link from node A to node E is a false one. It is assumed that most observed diamonds result from such false links. Diamonds are observed towards 79% of the destinations followed by loops occurring on routes towards 18% of the destinations. Cycles appear on routes to 11% of the destinations. The main reason for loops and diamonds is per-flow load balancing [32].

4.1.2 Paris Traceroute

Paris traceroute is based on traceroute and adds functionality to improve the handling of observed anomalies using the classical traceroute tool discussed in chapter 4.1.1. The classical traceroute suffers from the discovery of false links, while it may also fail to discover true nodes and links. The reason for the latter problem is that load balancers may simply decide not to use certain paths for the probes sent by traceroute for whatever reason. The problem related to per-flow load balancing is due to traceroute's own

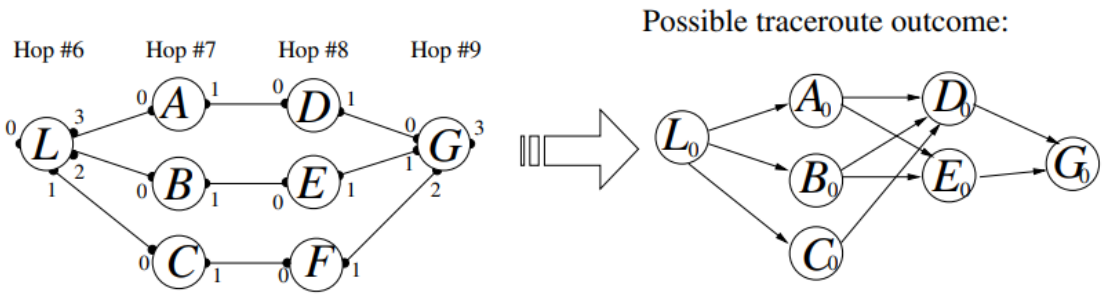

Figure 4.1: Network with diamonds
Source: [32]

behavior, as it varies the UDP destination port and the sequence number of ICMP *echo* probes.
Paris traceroute only varies header fields that are not used for load balancing. When sending probes via UDP, it changes the checksum field, making it necessary to adapt the payload so that the checksum is correct. For ICMP probes, Paris traceroute varies the *identifier* field in addition to the *Sequence Number* field. Although it does not discover all paths every time, it considerably increases the coverage compared to the classical traceroute tool.

4.1.3 Alias resolution

The alias resolution process, briefly mentioned in Chapter 3.3 groups IP interfaces belonging to the same router. Different approaches exist, some of which are mentioned in the following. [35] presents *Anaximander* which maps specific ISPs at the router level while adapting to the characteristics of the ISPbeing mapped. This leads to more efficient probing. [35] tries to achieve the same coverage with smaller communication costs than the common brute-force approach.

Anaximander collects information from RIBs, followed by a strategy phase where three core principles are applied:

1. Find an initial pool of targets expected to transit the ISP of interest. Include prefixes from BGP entries where the AS appears in the *AS_PATH* attribute as probes to this prefix will probably go through the same AS. Additionally, add internal prefixes to the AS. The data only gives a view relative to the BGP speakers.

2. Reduce the initial target pool by avoiding probes on prefixes where BGP will not decide for a route that includes the AS of interest. Anaximander tries to approximate local BGP decisions as there is no access to the local policy. It approximates the policy with the current practice in today's Internet, namely *no-valley and prefer customer routing policy*. It specifies to prefer routes through

a customer AS over a peer AS over a provider AS for economical reasons. This leads to the case where 4.1.1 tends to discover customer-provider links more easily than peer-to-peer links. In case this policy cannot be applied or there is a tie, the algorithm applies the following logic: include the prefix if the AS is in the shortest *AS_PATH*. Overlay reduction: Several related prefixes can be grouped in certain cases. The algorithm tries to retain only the most specific prefix. The reason is that forwarding on the Internet is usually done on a longest prefix match basis. Exceptions to this statement exist, like MPLS forwarding, for example, which has been discussed in chapter 3.1.8. Prefixes of different spceficity have following use cases: advertise reachability information, Traffic Engineering and overlays. The last one is of special interest, as the differences serve no purpose in this context, and a reduction can be made here.

3. Sort and schedule the remaining targets. The targets are divided into four groups: internal prefixes, prefixes to direct neighbors, one-hop neighbors, and other ASs. A router gets counted as discovered if at least two of its addresses are discovered. The group with the internal prefixes plays a major role for the coverage. The algorithm therefore starts probing with this group. The second group is further divided into customers, peers, and providers. With this group, the algorithm starts to probe the customer ASs. Whether to proceed with providers or peers is not important. Small ASs are probed before larger ones. The additional coverage contribution of the other two groups is at most five percent.

During the discovery phase, Anaximander jumps to the next AS when the discovery rate falls below a threshold [35].

Probing tools based on 4.1.1 often rely on caching to reduce traffic [35]. [45] is a well-known intra-domain topology discovery tool that relies on the following techniques to reduce traffic:

- **Ingress reduction**: Considers that probes that go to a given destination should converge when they enter the targeted ISP at the same ingress node.
- **Egress reduction**: Traces from the same ingress to any prefix beyond the same egress should traverse the same path.

[46] presents *bdrmapIT*, a tool that does not map the entire Internet but only the ISPs. The objective is to delimit ISPs as accurately as possible from the rest of the Internet. [47] presents *MIDAR* which can be used in combination with *bdrmapIT* [35]. MIDAR sends probes to possible routers on its own and uses the IP Identifier header value. Packets that have not been forwarded by routers, but for which the router is the origin, are of special interest for alias resolution. Routers may send packets themselves when responding to ping or traceroute requests, for example. There is no specification of how to set the unique value of the IP Identifier (ID) field. Many routers use a counter to

assign this value. Due to the field's size of 16 bits, the value range is 0 to 65 535. If a router uses IP ID counter shared across multiple interface addresses, their IP ID time series form a monotonically increasing sequence when merged. The IP ID time series refers to the sequence of IP ID values collected over time. MIDAR must be able to handle the case when a router, for instance, does not use a counter but a random number. A router might also use separate counters for each of its interface addresses. Tools that rely on the IP ID values might in this case miss those routers. MIDAR can produce the following three results with respect to the shared counter:

- positive shared counter and positive aliases
- negative shared counter and inconclusive aliases
- inconclusive shared counter and inconclusive aliases

When confronted with a high number of traceroute records, such as multiple millions, alias resolution tools relying on the IP ID values face two challenges:

- **Runtime**: Different time series must be tested against each other, resulting in an asymptotic runtime of $O(N^2)$ unless a more sophisticated approach is used.
- **Potential of high false positive rate**: Many records will have similar values by chance. Comparable tools such as Ally or RadarGun suffer from a false positive rate that is too high to provide reliable results for millions of records.

MIDAR uses the Monotonic Bounds Test (MBT) to test the closeness of two time series in order to verify if they satisfy the monotonicity requirement. Due to this test, MIDAR reaches near perfect sensitivity. Sensitivity is a statistical measure that indicates the share of correctly predicted values out of the real positives. Its formula is $truepositives/(truepositives + falsenegatives)$.

MIDAR considers a time series unusable when one or multiple of the following conditions are true.

- Less than 75% of the probes to a target resulted in a response.
- At least 25% of the samples are degenerate, which means they either use a constant IP ID value or simply echo the IP ID value of the probes.
- The time series cannot be modeled as monotonically increasing. This can be either the case because random numbers are used or because the counter counts too quickly, resulting in too many overflows. In this context, the term *negative delta* is important. Let the values v_1 and v_2 be two IP ID values of the two timestamps t_1 and t_2. The values belong to the same time series and $t_1 < t_2$. The delta value is

defined as $v_2 - v_1$. A negative delta occurs when v_2 is smaller than v_1. This can happen due to an overflow or a random-value assignment. 30% is the maximum allowed fraction for negative deltas that leads to the exclusion of 98,8% of random time series.

For reliably detecting counter overflows, also called counter wraps, a sampling interval shorter than the wrapping period is used. The wrapping period refers to the time between two counter wraps. The sampling interval refers to the length of a sample of an IP ID time series. The maximum acceptable sampling interval is defined as the largest interval that still ensures a maximum negative delta fraction of 30%.

MIDAR sends its probes sequentially. This means that it waits for the response of a probe to a target before sending the next probe to the same target interface. Therefore, there is never any uncertainty about the ordering of the samples within a single time series. However, different interfaces are probed in parallel, which may cause overlapping time series for a single router. However, this does not negatively impact the sensitivity [47].

4.2 Location of vantage points

Publicly available BGP path information does not cover the entire Internet due to visibility constraints, route aggregation, hidden suboptimal paths, and policy filtering. Being an active measurement technique, traceroute can be designed to potentially cover the entire Internet assuming that a large enough number of VPs is used. The following categorization can be made for the approach of scanning the internet:

- Scan from within the Internet core: This is the top-down approach, starting from the side of the ISPs.
- Scan from the edges of the Internet: This is the bottom-up approach, starting from the user's side [48].

The number, distribution and characteristics of the chosen VPs have a high impact on the obtained topology. [49] analyzes the effects of the VP distribution on the quality of the obtained topology. The work presents two methods for the selection of VPs.

- *omniscent*: This method is appropriate for optimizing measurement assignments that use statically deployed measuring devices.
- *oblivious*: This approach selects VPs at random and is suitable for the analysis of the coverage capacities of measurement efforts. The number of VPs is usually higher for this approach. Each agent only uses a subset of the total destination set. The probability that multiple agents are scanning the same address prefix is

low. DIMES [50] is a community-based example using this approach. Community-based approaches might suffer from irregular behavior due to the activity of the community.

[49] argues that increasing the number of VPs alone is not sufficient, but a diversity of location and type is necessary to reduce bias. The authors further mention that active measurement techniques have issues with the broadness, meaning that entire parts of the topology might be missing. Another issue may be the freshness of the destination list.

[49] mentions that the largest local topologies are most of the time discovered by VPs located in Europe or the United States. The number of ASs found remains mostly constant after a few VPs while the behavior is different for AS links, where increasing numbers are observed even with more than 80 VPs. Starting at around 150 VPs, VPs are observed that only contribute links that connect low-degree ASs to regional providers. The authors of [49] also observe that different graph measures of the obtained topology need a different amount of VPs to converge. The maximum degree AS that a VP has discovered indicates whether the VP detects core nodes of the Internet or not. The maximum degree converges after a few VPs, while the graph density converges slowly. When tools such as traceroute are used, the node degrees highly depend on the VP locations. The Internet topology graph's node degree distribution follows a power-law distribution with an exponent larger than two. Tree-like sampling processes produce systematic bias when applied to such graphs. However, the bias is negligible in magnitude. The work [49] notes that increasing the number of VPs does not guarantee unbiasedness.

[48] proposes another community-based system that uses smartphones as VPs. The system is split into client and server logic. The server logic runs on a central machine and represents the mind of the architecture. The server manages the different clients, supervises different measurement campaigns, and stores the measurement data collected. The server decides which clients are suitable targets for a certain campaign based on certain characteristics of the client. For this decision, the server logic takes into account the host target type: routers, fixed hosts, other clients. Also, the desired measurement type has an impact: capacity or RTT estimation are two examples. The server also decomposes a global task into smaller jobs that are distributed to the clients. The clients execute the measurement jobs communicated by the server, store the intermediate results as well as the user preferences, and show some information to the user in the user interface of the application. Both the server and client code are organized into different modules. The tool was restricted by the abstraction of the Java Application Programming Interface (API) as the manipulation of packet headers was only possible for the UDP. The algorithm is capable of estimating the route towards a given host, recognizing load-balancing routers, and enumerating their interfaces.

Paris traceroute presented in chapter 4.1.2 with its Multi-path Detection Algorithm (MDA) is able to discover the multiple next hop interfaces of per-flow load balancers. In

general, a path can be traced using one of the following protocols: ICMP, UDP and TCP. The approach of the mobile monitor is to emit a UDP probe with a given TTL value k. The k-th router will then discard the packet and reply with an ICMP *Time Exceeded error*. The application then stores the error and associates it with the respective socket. The implementation makes use of two types of threads:

- The **Breadth Explorer** creates a pool of depth explorers and waits for them to finish, aggregates the results, and updates the global data.
- The **Depth Explorer** synchronously sends a probe and waits for a reply. The depth explorers operate on different source ports. They also send probes to destination addresses that are adjacent to the target address but are still different from the addresses used by its siblings depth explorers. The use of adjacent addresses makes the detection of per-destination load balancing possible. Since also the source port is different, per-flow load balancers assign a different *flow id* to those packets. For more information on the detection of the load-balancing type, see Chapter 4.1.2 [48].

The tool analyzed 141 target destinations belonging to Italian research and education network (GARR). All interfaces are either located within PoPs or connected to backbone routers. The tool discovers exactly the same set of interfaces as traceroute. The average number of hops is 4.3. The total volume of IP traffic generated is 3.16 Megabyte (MB). The measured battery level reduction has in every case been at most 1%. The advantages of this approach are that the performance is observed at the edge of the network. The monitors also allow one to collect dynamical and geo-referenced information [48].

CHAPTER 5

Methodology

This chapter gives a detailed explanation of the different steps that make up the analysis of this work. It contains an explanation of the datasets used, how they have been gathered, processed, and visualized. Furthermore, it contains the analysis of the visualizations and some derived data.

5.1 Data sets

The data analyzed originate from measurements made on the Internet by CAIDA. The internet measurements are made with the tool *Scamper*, of which a brief explanation is given in the following chapter 5.1.1. The different data sets differ in whether and how the data have been processed and aggregated after collection.

5.1.1 Scamper

Scamper is a packet-probing tool designed for large-scale internet measurements. It supports IPv4 and IPv6 probing with MDA, different alias resolution techniques, as well as different traceroute features such as scanning with ICMP, UDP and TCP. Scamper does work on Linux, Windows, BSD, MaxOS X and Solaris. Scamper probes in parallel. The amount of parallelism is configurable via command line parameter specifying how many packets should be sent per second. Resources such as sockets are managed centrally and shared between tasks, if possible. Scamper also stores the completed measurements centrally. There are two ways in which the tool receives its probing endpoints:

- Static address list: Scamper receives the list of IP addresses via the command line or in a file.
- Run Scamper as *daemon* and receive the instructions dynamically [51].

5.1.2 Data description

All data used were obtained using traceroute. Depending on the exact data set, the traceroute results are included in their raw or aggregated format. The following publicly available files have been used:

- Warts files from 08.02.2022 until 23.02.2022[52]: In this context, a Warts file is a binary file generated by the tool *scamper*. CAIDA uses address prefixes announced via BGP for scanning. The prefix list is updated every seven days. In order to avoid overlapping prefixes, a single target address is created for every prefix [53].

- midar-iff.nodes.bz2[54]: This data set contains a list of nodes and interfaces. An example line shows the format of each line:

  $$nodeN5 : 59.152.205.1359.152.208.959.152.213.2159.152.218.5$$

  In this case, N5 is the router to which the four listed interfaces have been inferred to belong. Each line has exactly one router and at least one interface. Only the IPv4 interfaces are included in this dataset. Each router is only listed once.

- midar-iff.links.bz2[55]: This data set contains a list of links between routers and interfaces contained in the 5.1.2 dataset. Each link is an IP layer link that connects the nodes listed on the respective line. Multiple routers can share one link. An example of a line of this dataset is the following:

  $$linkL1 : N24635037 : 1.0.0.1N111892N111893$$

  This line indicates that the link with the logical id *L1* connects three routers. In case the router interface being connected to the link is known, the interface is noted together with the logical node id. This is the case for the first node listed in the example link.

The second and third datasets both only contain aggregated data. It is based on traceroute measurements collected from the 8^{th} to 23^{rd} of February, 2022. The dataset used as the base dataset is *The IPv4 Routed /24 Topology Dataset* also gathered by CAIDA [56]. This base dataset contains traceroute records gathered by Scamper. Random destinations are selected on a random basis from every routed IPv4 /24 prefix. In contrast to the warts files every address is only probed by one source per probing cycle. As random destinations are used, no consistent IPv4 address measurements are included. Scamper uses Paris traceroute (see 4.1.2) with ICMP for the probing.

VerticesId:ID(Vertices)	:LABEL
1	r
103.10.67.69	i

Table 5.1: Format of the vertices.csv file

5.2 Process description

The process goes from the data gathering to its visualization. Most of the process is automatic, but some initial manual steps are involved. As the process covers different use cases, it has a certain complexity. The use cases for example differ in the aggregation type, analyzed period as well as the base data used. The complexity also comes from the fact that the program has to deal with a large amount of data. A detailed description of the workflow is given in the following sections.

5.2.1 Data gathering

The datasets 5.1.2 and 5.1.2 (in the following denoted by midar datasets) are downloaded manually while warts files 5.1.2 are downloaded automatically during the main processing workflow 5.1. For the data contained in the warts files, the data is reduced such that only data in the range from 8^{th} to 23^{rd} of February 2022 are included. The logic for this data reduction is explained later in this chapter when the main processing workflow is discussed in detail. Some visualizations further reduce the date range. For the visualization where this is the case, it is explicitly specified. Before analyzing, any lines in the midar data sets not adhering to the above mentioned format need to be removed from the files. In order to preprocess the midar datasets, the files are imported into a Python Pandas dataframe and converted to the parquet format for more efficient processing. In order to save space, the *Node N* and *Link l* line prefixes are removed. In addition, the router prefixes in the links midar dataset 5.1.2 are removed. The router and link logical ids are then converted into integers. Both data sets are then transformed into Comma-Separated Values (CSV) files using a format that allows to import the data into a Neo4j database. An example of the data used for the CSV file that contains the node list is given by the table 5.1. The first line is the header line that provides the metadata for Neo4j. Every other line contains either a shortened logical router id or an IP address together with a label that indicates the node as router or interface.

An example of the data of the CSV file containing the edges is given by table 5.2. The first line of the file is a header line that contains some metadata for Neo4j. The data contains two types of connections: an interface belonging to a router and a router being connected to another router. The former type is denoted by *belongs_to_router* in the *type* column. The first two lines state that the interfaces *24.38.46.249* and *24.157.32.25* belong to the router with the logical shortened id *1*. The remaining two lines state that the nodes *71513356* and *83205372* are connected to the node *5410974* via the links

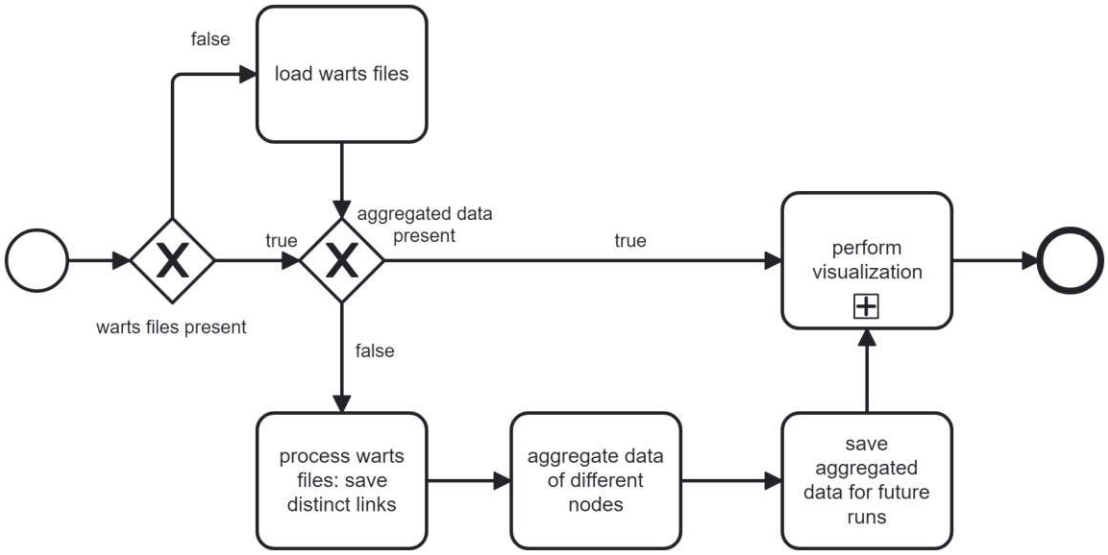

Figure 5.1: Main processing workflow

84524649 and *84524650*. There is no direction specified for the edges, as this information is not necessary for further analysis.

The data of the two resulting CSV files are then loaded into an empty Neo4j database. This is done manually by executing the following command of the Neo4j Command line interface (CLI).

```
sudo neo4j-admin database import full --nodes=
  ./data/vertices.csv --relationships=./data/edges.csv
```

The remaining logic for preparing the data for the analysis and creating different visualizations is done by a python program. The program has several options for controlling the behavior:

- max-files: receives the number of files that should be loaded and parsed. The default value is 300.
- reload: if some warts files already exist, they get deleted and all the necessary data gets loaded. This option is a boolean and is *false* per default.
- cores: Specifies the number of cores that the program should use for the parts where parallelization is implemented. The default value is 16.
- destinations: This is a boolean option with the default value *false*. If set to true, the figure 6.9 is generated. Otherwise, the figures 6.2 to 6.8 are generated.

- eigenvalues: This is also a boolean option being false by default. If true, the program generates the eigenvalue figures 6.11 to 6.12. It cannot be set to true together with the destinations-option.
- plot: This is a boolean option that is *false* by default. Depending on its value, the figures are created or not. The motivation for this option is the ability to only run the parsing of the warts files. It also loads the warts files if not done already or the reload option is true.

To compute the results provided in this work, the following commands have been used:

- Generate the figure 6.2 to 6.8 (also called standard figures from now on): nohup python3 main.py –plot &
- Generate the figure 6.9: nohup python3 main.py –plot –destinations &
- Generate the eigenvalue figures: nohup python3 main.py –eigenvalues &

The program ran on a virtual machine running the Ubuntu operating system with version 22.04.3. The Random-Access Memory (RAM) available for use was 294.73 GB. 16 virtual cores were available. For generating the standard figures the program needs around 12 hours to complete if the warts files have been loaded and parsed already. The command *nohup* and the trailing ampersand of the commands used allow the program to continue to run even if an Secure Shell (SSH) session gets terminated. This is recommended to use, as otherwise the process can get terminated due to network issues or timeout settings, for instance. For local development a smaller number of files and cores have been used. Oftentimes, when debugging, it is advisable to run the program with one single core. At the parsing phase, 195 files are processed while some of them might not be needed as files outside the desired date range get downloaded as well. This is because for most sources, a complete scan takes more than one day to finish. This can lead to the case where data from the desired start date may actually be stored in files from scans that began on earlier dates. The parsing phase takes almost 60 minutes for the standard and eigenvalue data.

The general workflow of this program is visualized by the previously mentioned figure 5.1. The first time the workflow logic is executed, the program downloads all warts files of a specified time interval. Unless specified via a command line option, the program does not download the data again the next time it is run. One warts file contains data from one source node of one complete scan. Not all warts files from a source contain the same amount of data per day. Some sources do have days without any data. No extrapolation of missing data is performed. In order to extract the data from the file the *scamper pywarts*[57] python library is used. As warts files are binary files, the content can most easily be extracted by using such a library. The warts data are processed record by record synchronously. To speed up the parsing depending on the value of the above-mentioned cores parameter of the Python program, multiple files are parsed in parallel. Each record

:START_ID(Vertices)	:END_ID(Vertices)	type
24.38.46.249	1	belongs_to_router
24.157.32.25	1	belongs_to_router
5410974	71513356	84524649
5410974	83205372	84524650

Table 5.2: Format of the edges.csv file

contains a source and destination address in the form of an IP-address. Additionally, the date at which the traceroute command has been started is contained as well as a list of all nodes traversed from source to destination, including the source and destination themselves. Only traceroute records with a start date in the inclusive range of 8^{th} to 23^{rd} of February 2022 are included. The hops are then stored in pairs per source address and date. This means that the combination of the address and the date serve as the key while the hop pairs represent the edges. Not all of the visualizations are generated during the same program run. Destination plots are created in a separate run. If the destination flag is passed to the program, the parsed warts data is stored per destination. In order to avoid duplicates, the hops are ordered before storing. Each file contains around one million traceroute records. The parsed data are saved to a file that can be easily restored by the program. At subsequent runs, the warts files are not parsed again as long as the data can be restored from the file.

5.2.2 Data preparation

The 5.1 contains the subprocess 5.2 that performs the visualization as a last step. This subprocess transforms the data in such a way that it has the desired format for plotting. Some figures require a specific preparation, while others just need a more general form of aggregation. The data on which this process starts to work are a set of interface links per source interface. These data are flattened and then put into a set. From this set, the respective routers are retrieved from the Neo4j database. This is done using the listing 5.2.2.

```
MATCH (n:i)-[e]-(r) where e.type = 'belongs_to_router' and
n.VerticesId in $interfaces return distinct r.VerticesId
```

The query does the following: match every node with the interface label that has a relationship to a router. The relationship type has to be *belongs_to_router* and the id of the node (*VerticesId*) has to be included in the *interfaces* parameter provided to the query. For each of the (router) nodes fulfilling those criteria, the id is returned. The *distinct* keyword ensures that no router id is listed more than once.

In the next step, it depends on which plot is needed. In this work, not all of the figures that can be created by the program are shown. However, all parts of the workflow are explained. The workflow for the figures 6.2 to 6.7 is explained in the first place. At

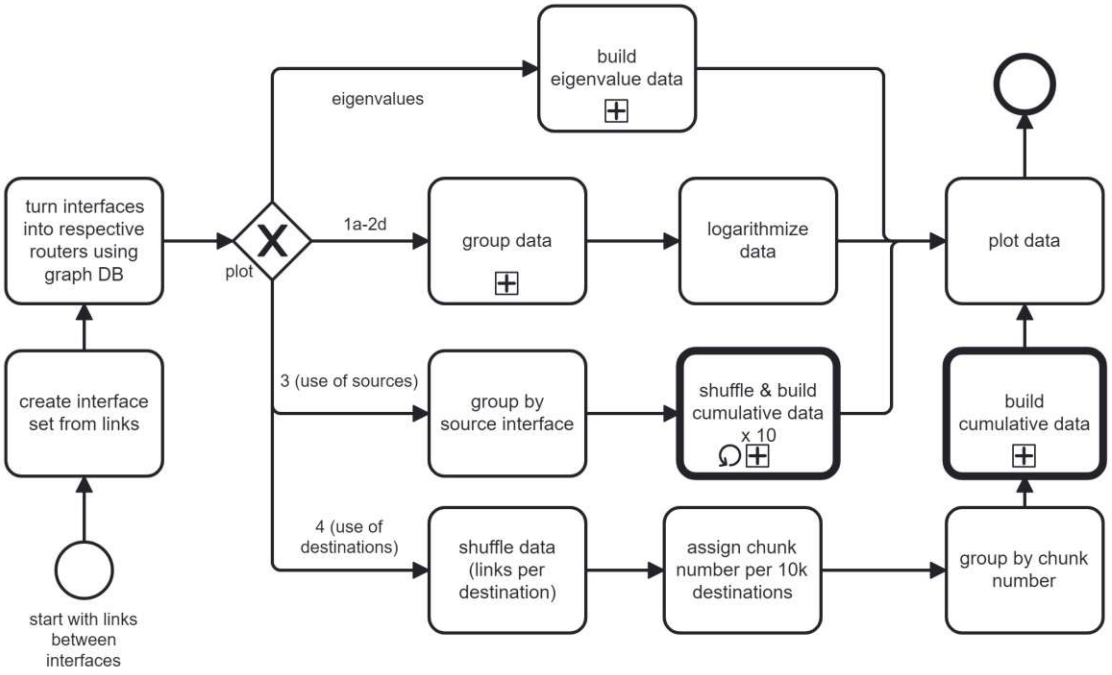

Figure 5.2: Visualization general workflow

first, different group operations are performed. This is visualized by the subprocess 5.3. Depending on the exact plot, the process starts working on the data where the sources are either interfaces or routers. Furthermore, a distinction can be made whether the plot should visualize the link or node data. However, the subsequent workflow is the same for the latter two cases. In the case where routers should be used as source instead of the interfaces, the needed data are grouped by router, as eventually routers could appear multiple times. The data is again aggregated using the union operation on the sets. Using the set datastructure ensures that no duplicate entries are stored. On the resulting data, the sizes of the sets are calculated. Afterwards, the resulting data is grouped per source or date. Set sizes are used to calculate the minimum, arithmetic mean, maximum, and sum metrics per group. For calculating the sum, a custom aggregation function is used. This function performs the union of multiple sets and returns the size of the set returned by the union operation. The resulting data are then plotted.

In case the figure 6.8 should be created the data is grouped by source interface as the base data is per source per date. Afterwards, the cumulative data are built 10 times in a random way. This part is visualized by figure 5.4. The order in which the data are processed is relevant. Therefore, the data are shuffled before being passed to the subprocess. The results of each data row are needed to build the next data row. The subprocess starts with an empty set, iterates through the rows, adds the current set of links to the existing set, and copies the current cumulated set. After having iterated through all of the rows, a new dataframe exists with the number of rows equal to the

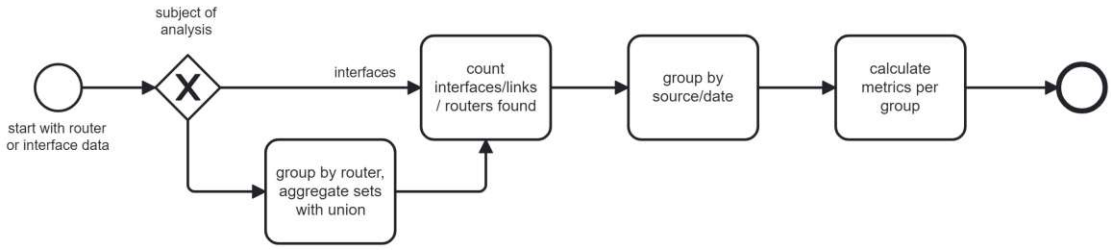

Figure 5.3: Grouping for visualization

input number of rows. Each row of the new dataframe holds a separate set, while the sets can only grow in size by growing index. Afterwards, separate interface and router sets are computed from the link sets. Then the sizes of the sets are calculated followed by an assignment of the row numbers representing the x-axis.

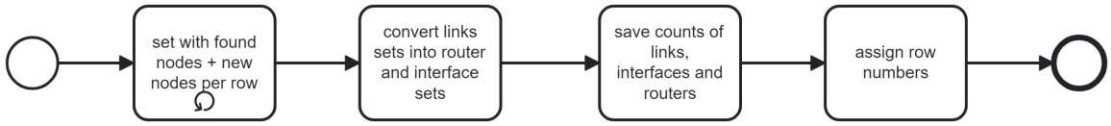

Figure 5.4: Build cumulative data for plots 6.8 and 6.9

Figure 6.9 includes data from all dates. As for the figure 6.8, 10 runs are made of which each performs the following logic: select 10 thousand destinations, each serving as a key, randomly. Associated with each destination, separate sets exist for the links, interfaces, and routers. Then those sets are unified with the sets of data already discovered. This part of the process is repeated until all destinations have been selected.

All eigenvalue workflows group the data by date or source interface. Afterwards, it depends on whether or not a cumulative plot should be generated. At first, the most simple process is explained. The eigenvalue figures 1a and 2a are not cumulative while the other eigenvalue figures are. After the grouping, there is one set of links for each group. Adjacency matrices are built for these sets. Sparse matrices are used for this. To do this, the program iterates through the links and converts the list of links between the nodes into the Compressed Row Storage (CRS) matrix format. This format is used by the sparse.csr_matrix function of the scipy library and is a common format to efficiently store sparse matrices (see [58] p.57). To give an example, let G_1 be a graph whose connections are visualized by figure 5.5. The adjacency matrix for G_1 is depicted by table 5.6. The set of links created by the Python program would look like this: $(A, B), (A, C), (C, D)$. Using the CRS format, the graph would then be represented in the following way: $rowIndices = [0, 0, 2], columnIndices = [1, 2, 3]$. The order in which the nodes are processed does not make any difference as the matrix eigenvalues are invariant to row/column permutations. Also, the exact labels used do not make any difference as the eigenvalues only depend on the structure ([59] p. 271-286). Based on this matrix

representation, the 10 largest algebraic eigenvalues (adjecency spectrum) are calculated using scipy.sparse.linalg.eigsh. Those values build the adjecency spectrum. Afterwards, the adjacency spectral distances are calculated using the *L2-Norm*. The distance of the subgraph is calculated against the complete graph of the current group. Afterwards, this data is plotted.

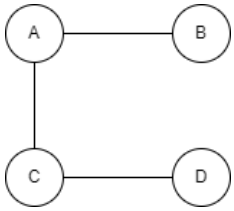

Figure 5.5: Graph

	A	B	C	D
A	0	1	1	0
B	1	0	0	0
C	1	0	0	1
D	0	0	1	0

Figure 5.6: Adjacency matrix

Next, the workflow for the eigenvalue plot 1b, which is a cumulative figure, is explained. The data is first again grouped by source and sorted so that the smallest date comes first. The algorithm then loops through the data rows, adding the current link set to a cumulated set. The current set is also copied into a separate list. Afterwards, the eigenvalues and distances are calculated as for the previous workflow type.

Finally, the workflow for the eigenvalue figures 2b_* which are also cumulative figures is explained. The '*' character represents the date, as a separate figure is created for each date in the range 8^{th} to 23^{rd} of February 2022. The program iterates through the dates and reduces the data so that only the date of the current iteration is included. The algorithm starts with a random selection of 10 different sources as 10 different runs are made per level. Let x be the number of unique source interfaces of the data of the current date. The algorithm then makes x-1 iterations representing the levels. In the first iteration, the previously mentioned initial random selection is stored. This includes the eigenvalues, the level and run numbers as well as the graph number which is an integer working as an alias to the source. This data row represents the first level. At the following iterations the existing graph selections will be extended by one further source. The program keeps track of the past selections as a string. The string contains the selected aliases separated by a single '-' character. At first, a random selection of the previous choices is made. For further explanation, it is called *existingSolution*.

existingSolution is then divided into its number parts. The list of unique source interface aliases is then reduced to the ones not used yet. A random selection of the available numbers is made and joined with *existingSelection* to form a new graph number. The numbers in the created graph number are sorted to be able to check for equality. Then it is verified that the new graph number is not yet in the current level. Otherwise, the process of choosing a new graph is run again. If no new graph could be found after 100 tries, the process is aborted. This happens if there are not enough sources available. An example of such a situation would be a reduced data set with only 3 sources. Then it is

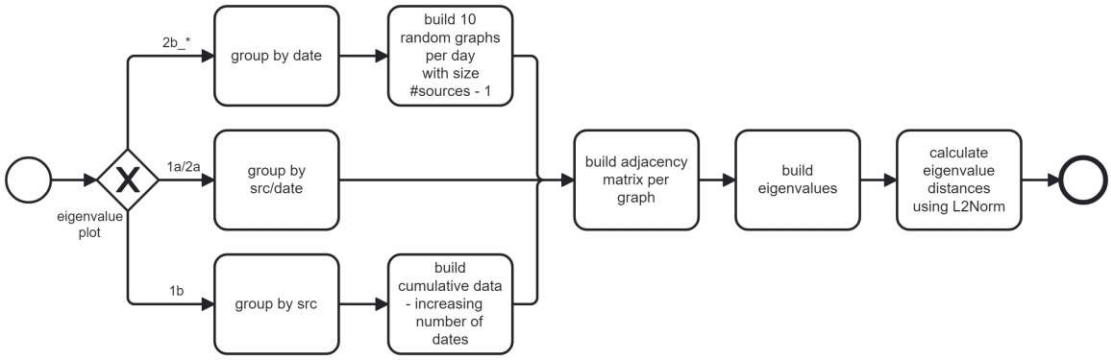

Figure 5.7: Build eigenvalue plots

not possible to come up with 10 different and unique random graphs, no matter which level. The link set of the previous level is then joined with the one from the current selection and recorded together with the new graph number. The eigenvalues are then calculated for each link set. Afterwards, the complete graph of the current date is built. In the next step, the adjacency spectral distances between the subgraph spectra and the complete graph spectrum are calculated. Intuitively speaking, the workflow creates 10 different graphs of level #sources - 1 per day and records all intermediary steps. As the size and therefore also the coverage increase with increasing level, the distance to the complete graph becomes smaller. For this figure type two subtypes exist:

- Plot the 10 lines, each representing one run.
- Plot the median distance for each level as well as the standard deviation of the 10 data points per day.

5.3 CRISP-DM Process Documentation

This section documents the most relevant parts of the iterative cycles and feedback loops encountered during data analysis using CRISP-DM. It shows the main decision points and the review of previous decisions and implementation approaches.

During the Business Understanding phase, it was crucial to have a solid understanding of the requirements. For this, a broad knowledge of Internet measurements, especially topology discovery, was necessary. To obtain this knowledge, the sources collected during the SLR have been analyzed and the background section has been written. Furthermore, the description of the data provided by CAIDA of the data has been used together with the early analysis of the available data. Uncertainties regarding requirements have been discussed and resolved during discussions with the assistant supervisor.

Moving to the Data Understanding and Preparation phases, initial implementation approaches followed with the aim of creating the first plots being the most simple ones.

During the implementation of the subsequent plots different approaches for handling the large amounts of data have been tested and evaluated. The first approach was to use NetworkX [60] to model the complete graph. This option turned out very early not to be the right choice due to issues encountered with first experiments with the data forming a large-scale graph. igraph [61] has been chosen as an alternative. This formed a feedback cycle back to Data Understanding as the performance issues revealed limitations in our initial approach to handling the great amounts of network data. This worked sufficiently well until the growing analysis complexity led to the fact that the host machine with 32GB of RAM and 12 cores needed six to seven hours to complete. Next, the choice has been made to internally use the parquet format instead of csv files. This reduced the run-time to four hours. It has been decided that igraph is not performant enough for the remaining analysis. Amazon Neptune [62] is a tool that was evaluated but turned out to be too expensive. Subsequently, Neo4j [63] was selected as the final choice for implementing the overall topology graph. further improvements to the runtime have been made by processing the warts files with multiple threads. A virtual machine running Ubuntu with 32GB RAM and 16 cores has been provided.

Good progress has been made with the plots. An ambiguity regarding how duplicate links and nodes should be counted when grouping sets has been discussed with the assistant supervisor, the result being that the sum should be size of the joined sets. Afterwards, the first plots 1a to 1d have been completed and an initial implementation of the plots 2a - 2d has been created. Further feedback had the suggestion of using the logarithm with base 10 instead of the natural one, to use a grid and consistent colors for the metrics. This has been adapted accordingly. This additional feedback cycle demonstrated the iterative part of the Modeling phase, where data preparation and presentation approaches are influenced by the requirements for the visualizations.

During subsequent implementation, the conversion from interfaces to routers using Neo4j caused problems. The investigations carried out included monitoring resource usage, comparison of Java and Python versions, analysis of query length, and the number of interfaces passed as query parameter. Due to the fact that no progress has been made, even though significant effort has been invested, a pragmatic decision was made to prioritize completion over optimization by running this part of the code only with a single thread.

During the Evaluation phase, the feedback regarding plot four was to perform 10 runs where the selection of starting graphs was random. The initial idea for modeling the subgraphs for the spectral distance plots was to use Neo4j. As the version of Neo4j used did not have the possibility to compute the eigenvalue distance between the graphs, SciPy is used to represent the graphs used for the distance analysis. The requirements for the distance plots have been concretized during discussions with the assistant supervisor. Afterwards, an algorithmic workflow was established and communicated. With the first

results being available, feedback included the issue of complex numbers being included in the results, which could be solved by using a symmetric adjacency matrix instead of a non-symmetric one.

Further feedback included that it would aid for analyzing and understanding the data when exporting the subgraph index, to single graph number mapping. The following analysis of the plots revealed a pattern in which some of the sources showed numbers that alternating every second day. Further investigation with the assistant supervisor and CAIDA led to the finding that this is due to the different packets per second (PPS) rates used by the sources.

After the analysis of the existing in order to give some recommendations discussions with the assistant supervisor resulted in the decision to plot additional information. Concretely, a plot visualizing the marginal gain rate of the number of interfaces, routers, and links has been created. The plot also includes three threshold lines that indicate different improvement rate values. The grouped eigenvalue plots were adapted to additionally visualize the distance's median improvement rate together with the standard deviation. Multiple analyses of the adapted grouped eigenvalue plots revealed some mistakes during the data transformation process for this type of plot. Due to the complexity of the plot, a significant effort was needed to correctly prepare the data.

CHAPTER 6

Evaluation and discussion

In this chapter the results obtained by running the created Python program as specified in the previous chapter are analyzed and interpreted. Finally, the initial hypothesis of this work is evaluated using the results, and a recommendation regarding the number and characteristic of VPs to use is provided.

6.1 Data analysis

Although the neo4j graph is not the primary result, it is considered beneficial to understand the data and the context to briefly analyze it. In total, the graph contains approximately 217 million edges between 166 million nodes. Figure 6.1 lists the five nodes with the maximum and minimum node degree.

IP-address	Degree
154.54.10.46	10 160
203.234.255.169	10 118
154.54.10.234	10 076
212.187.216.237	9 726
62.115.155.29	9 169

IP-address	Degree
211.240.84.1	1
191.99.102.1	1
71.247.208.1	1
108.56.199.1	1
103.66.57.1	1

Figure 6.1: 5 nodes with highest and smallest degree

For calculating the top five node degrees, the Cypher command 6.1 has been used. It matches all nodes of the graph and returns the value of the *VerticesId* attribute, as well as the calculated degree by counting all edges per node. The query is limited to five nodes, which in combination with the descending order gives the top five node degrees.

```
MATCH (n) RETURN n.VerticesId, count{(n)--()} AS degree
ORDER BY degree asc LIMIT 5
```

In order to get the five nodes with the lowest degree, an ascending sort direction has to be taken. When performing IPlookups for the top five nodes, one can see that they all belong to ISPs. The places one and three belong to Cogent Communications Inc., an ISP in the United States while the second place belongs to KT Corporation, an ISP in South Korea. The nodes with degree one belong to sim card providers such as Ritesim.com in the case of 103.66.57.1 but also ISP companies such as ElimNET Inc. in the case of 211.240.84.1. It should be kept in mind that there are many more nodes with degree one. This is only an arbitrary selection of five nodes with the minimum degree. In total, there exist 341 848 nodes with degree one. Those nodes cannot be intermediary routers as they have at least degree two. They can be source and destination nodes, while most of them are destinations as only 17 different sources have been used for scanning. The destinations can be edge routers/gateway routers, while there are also networks whose hosts directly are the destinations.

Figure 6.2 shows the interfaces found per day. As the data are grouped by day, this means that for each day there exist x data points, x being the number of source interfaces. For each date, the following metrics are calculated individually: maximum, minimum, sum, and arithmetic mean. As specified in chapter 5.2.2, the sum in this case is not the arithmetic sum but the number of interfaces found that day over all sources.

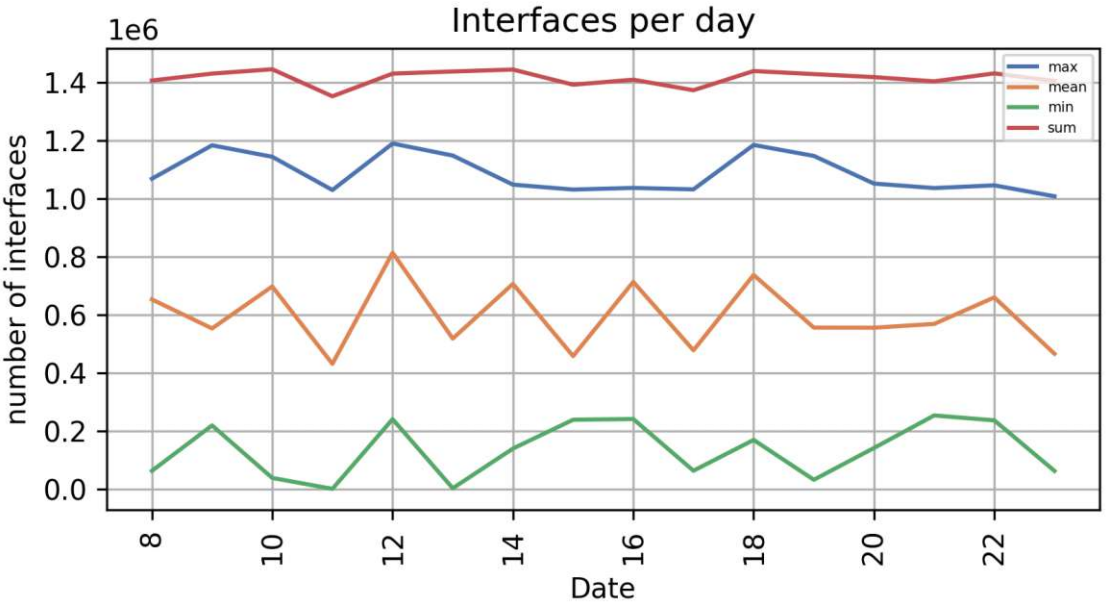

Figure 6.2: Interfaces found per day - plot 1a

One can see that the sum and maximum are more stable than the mean and minimum. The latter two have more variation than the former two. One can also see a certain periodicity, while the mean shows this pattern the clearest. The periodicity stems from the fact that a scan has not finished at the time the next scan should start, it does not start that day at all. Most of the sources need less than 48 hours but more than 24 hours

for a scan. That is why most of the sources have warts files available for every second day. The duration of the scan depends on many factors, such as latency and the number of timeouts that the sources receive. An important factor is the PPS rate which varies between sources. [64] mentions that different scan rates are used by different sources. An analysis of the warts files confirms this. Various numbers are calculated during warts file parsing:

1. Number of records: number of traceroute records in the file
2. Number of packets sent: each traceroute record has an attribute for the number of packets sent in the context of the respective record; all of the values are summed up per file
3. File name and source address
4. Minimum and maximum start-dates per file: a record only has the information of the start-date. The maximum start-date is therefore a lower bound for the end-date of the scan.

To estimate the PPS rate, the following formula can be used:
$|packets|/seconds(maxStart - minStart)$

Table 6.1 shows some of those measures, including the estimated PPS rate. The table includes data from two different sources. Each line represents data from one warts file. The data is ordered by source address and then by start time. One can see that the PPS rate is quite stable per source but differs a lot between these two sources. Regarding the estimated PPS rate, the sources with the minimum and maximum rates are shown here.

|Records|	Source	Start	Duration	PPS
926443	130.206.158.142	08.02.2022 23:20	01 01:02	297,18
925593	130.206.158.142	10.02.2022 01:00	01 00:53	297,22
926526	130.206.158.142	11.02.2022 23:20	01 00:51	297,16
926526	130.206.158.142	13.02.2022 01:00	01 00:49	297,15
926954	130.206.158.142	14.02.2022 23:20	01 00:57	297,15
924865	140.192.218.138	07.02.2022 02:00	02 20:18	99,91
925593	140.192.218.138	10.02.2022 02:00	02 19:52	99,91
926526	140.192.218.138	13.02.2022 02:00	02 19:40	99,90
926954	140.192.218.138	16.02.2022 02:00	02 20:13	99,90

Table 6.1: Traceroute scan statistics of 2 sources

During the same time period, the source *130.206.158.142* sends three times as many packets as the source *140.192.218.138*. For the total number of packets, this does not make a substantial difference as the destinations used per source are the same. Different sources just need a different amount of time to perform the scans. The duration is

presented in the following format: *dayshours* : *minutes*. One can see that the source *140.192.218.138* performs two scans in a time period of six days, while the source *130.206.158.142* performs four scans in the same period.

There does not seem to be a day that outperforms or underperforms all the other dates. The minimum is not as important as the other measures shown here. The number of interfaces found per day over all sources is around 1,4 million while the mean for a single source per day is around 0,6 million. As the arithmetic mean is used, outliers may have a large impact. It looks as if the minimum value is 0 at some points, but that is not the case. To showcase this, figure 6.3 is included showing the same data as 6.2 with the log_{10} transformation applied to the values of the y-axis.

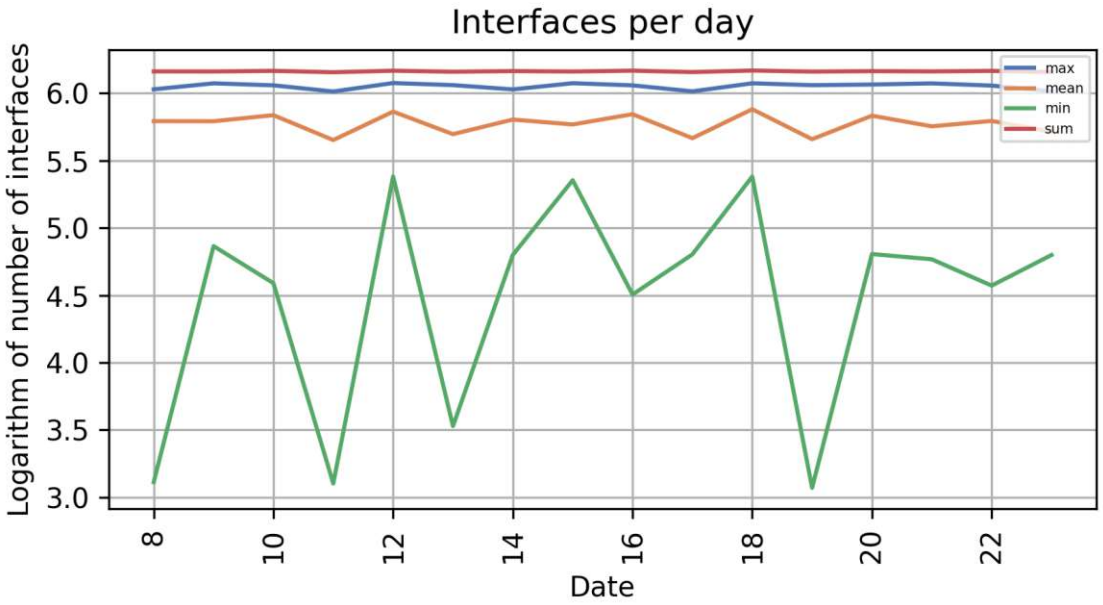

Figure 6.3: log_{10} interfaces found per day - plot 1a_log

Figure 6.4 shows almost the same data as 6.2 except that the y-axis shows the routers instead of the interfaces found. The y-values are slightly lower here, which is necessarily the case as some interfaces are found where multiple interfaces belong to one router. However, more interesting is that the sum is more than twice the mean. This means that on average one source finds almost half of the total routers found that day, while finding between 40 and 45% of the interfaces of the day. The difference between sum and max is larger (in relation to the total range) in figure 6.4 than in figure 6.2. Also, the range from minimum to maximum is smaller in figure 6.4. The number of total routers found per day is less than a third of the interfaces found. This might mean that on average 3 interfaces belong to one router. However, this hypothesis has not been verified and different possible explanations may exist.

Figure 6.5 shows the links found per day. The numbers are more than doubled compared to figure 6.2. Especially the sum is interesting. Between three and four million links are

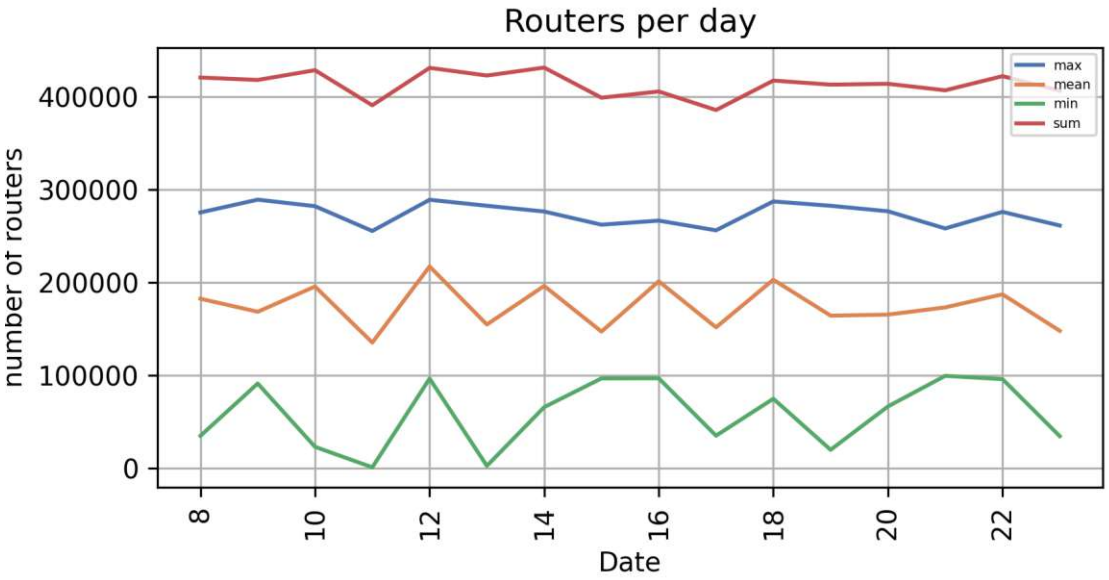

Figure 6.4: Routers found per day - plot 1b

found per day. For every interface, the scans find around 2.5 links, which shows a high number of redundant connections, though we do not know anything about the distribution of those links. It is also interesting that the gap between the sum and the other measures is much larger, which also means that the range from minimum to maximum is smaller in relation to the total data range. Load balancing is a possible explanation like, for instance, when traffic is routed based on the source IP-address, which means that a higher number of sources (or differnt IP-addresses) is more important for the link coverage than for the router and interface coverage. It seems that the affect of negative peaks is amplified for the sum which indicates that the mean is not drawn towards zero by an outlier but it rather seems that many data points contribute to negative peak as otherwise the sum would not have such a significant change. Furthermore, the sum is less stable than for the previous figures.

Figure 6.6 shows the same metrics as in the previous figure for the interfaces found per source interface. The overall sum is stable again, while the remaining metrics have much more variation. One can see that some sources seem to have less coverage per day than others, in general. Examples for such sources with a lower coverage are 200.27.115.62 (Universidad de Chile) and 192.43.244.202 (National Center for Atmospheric Research, Caida, Colorado) which have smaller maximum, mean and minimum values. Over all days, they still achieve roughly the same coverage as the other sources, which is visible when inspecting the sum. The 130.206.158.142 source (Universidad de Navarra, Spain) has at least one day with the best coverage, while it also has a day with quite bad coverage. Then there are 139.18.11.241 (caida.uni.leipzig.de), 140.192.218.138 (DePaul University, Illinois) and 150.183.95.135 (Caida South Korea), which have pretty stable coverage as all the three metrics are close together. They only achieve medium or even

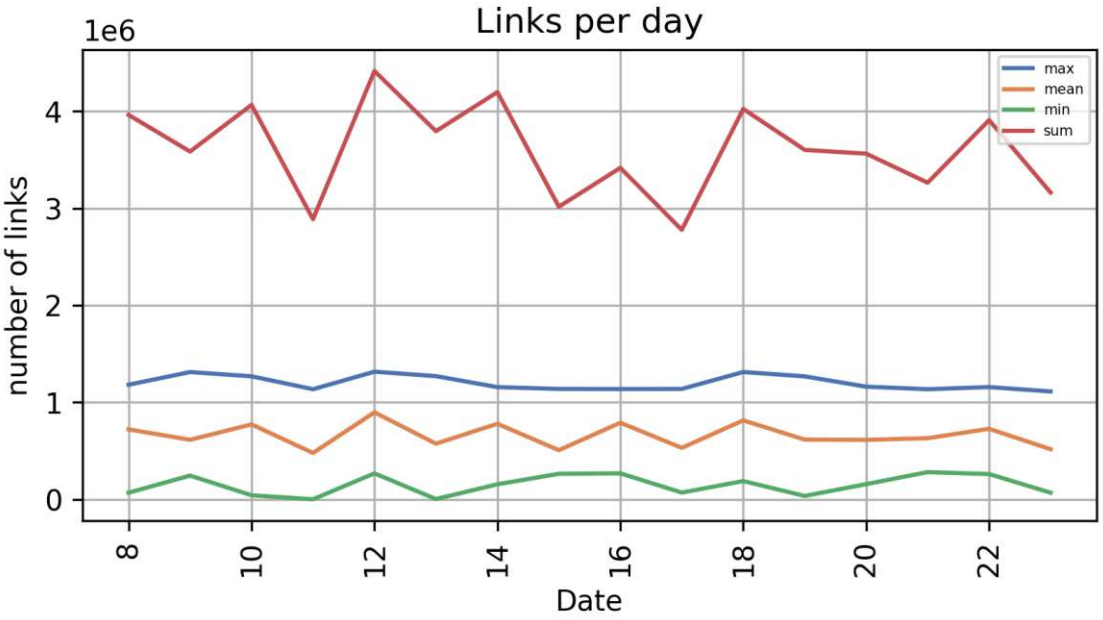

Figure 6.5: Interface links found per day - plot 1c

worse per day coverage.

Figure 6.7 looks pretty similar to figure 6.6, the data is different, though. It is the only plot where the data is grouped per source router. In practice, this does not make a difference, at least for the analysis performed for this work, as no two source interfaces belong to the same router. The fact that the number of distinct sources is 17 no matter whether the data is grouped by source interface or router confirms this statement. However, this is not necessarily the case in general, as some sources use a range of IP-addresses (see [64] p.2). Not only is the data on the x-axis different, but also the one on the y-axis, as it shows the number of links found. The general pattern resembles the one of figure 6.6. The sum is less stable as is the case with figure 6.5 where also the number of links is analyzed. One also again sees that the range from minimum to maximum is smaller for the data analyzing links than the ones showing the number of interfaces or routers found on the y-axis.

Figure 6.8 analyzes the data of the 17th of February only. The interfaces, routers, and links (between interfaces) found are shown in a cumulative plot. The x-axis shows the number of source interfaces taken into account. On that specific day data are available only from 10 sources. The order of the sources is random. The logic to compute the data for this and the following plot is explained in detail in chapter 5.2.2. Figure 6.8 only shows the aggregated metrics and not the single runs. One can see that the coverage of the interfaces and routers starts to stagnate at around five sources.

Figure 6.9 visualizes the cumulated number of interfaces, routers and links (between interfaces). It looks as if the line representing the number of routers is flat quite early but

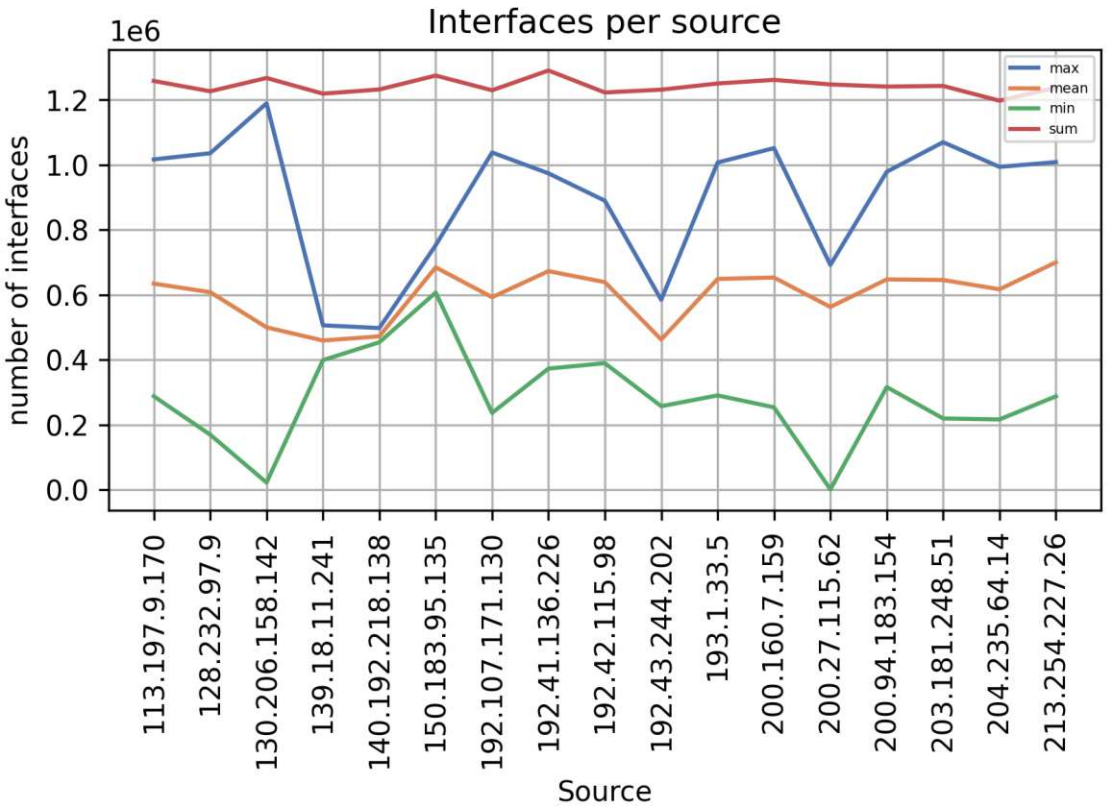

Figure 6.6: Interfaces found per source interface - plot 2a

it should be kept in mind that the magnitudes of the two axes is different. Even between 790 and 800 thousand destinations used the additional number of routers found is 2190. The slope of the curve representing the number of interfaces is greater than the former one. The reason is that the higher the number of routers found the higher the probability that a new interface found belongs to an already found router. The x-axis shows the number of destinations in steps of 10 thousand. The number of routers, interfaces, and links increases as the number of destinations used increases. This makes perfect sense. It is still surprising that the curve of the links does not seem to become flat. The reason for this could be load balancing.

Figure 6.10 shows the distance of the eigenvalues per source interface. It can be seen that the distance decreases in all cases when navigating from smaller to larger values on the x-axis. The distance is still quite different for some sources. Especially the source 204.235.64.14 can be pointed out here as the distance at the x-axis value *08* is 370,11 which is the highest value. The second highest value is 227,94 (192.107.171.130). The distances of both sources are largely greater than the other distances over most of the visualized date ranges. For most of the sources, the greatest decrease in distance can be observed from the first day to the second day. The two sources mentioned above are an exception to this, as they have a greater decrease at another time: from day 19 to 20 in

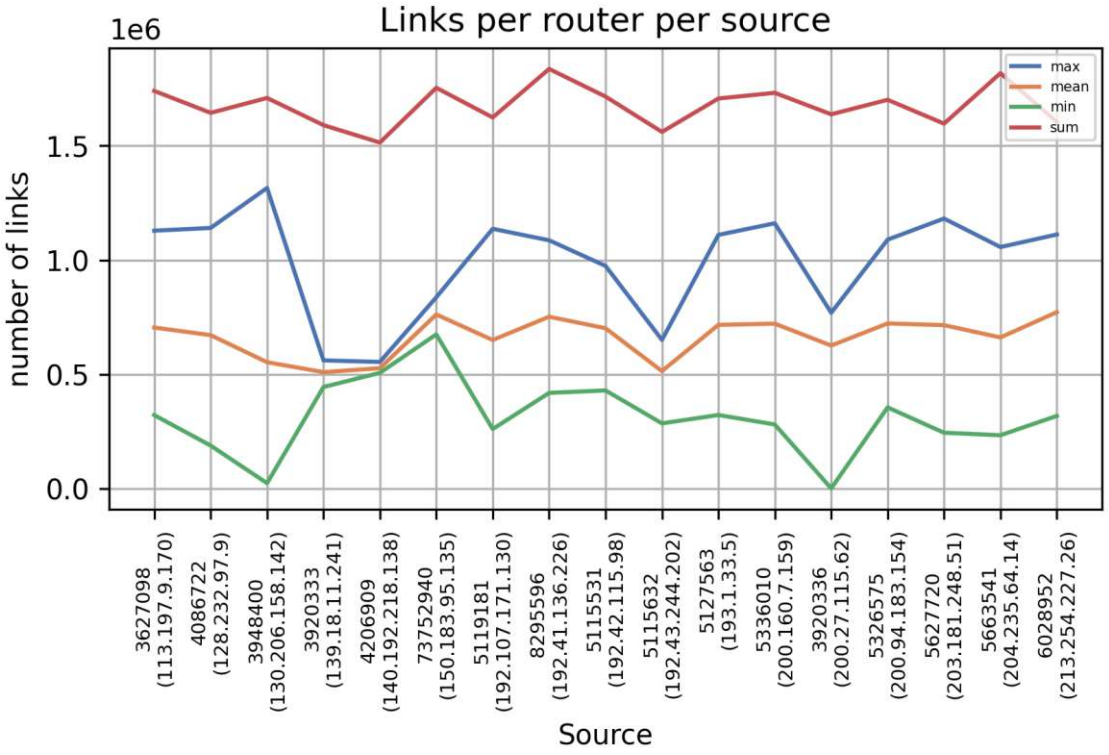

Figure 6.7: Interface links found per source router - plot 2d

the case of 192.107.171.130 and from day 21 to 22 in the case of 204.235.64.14. Another exception is the source with IP-address 139.18.11.241 which has no point in time during the analyzed period with such a great relative decrease. Other than one might conclude from the visualization, the distance at the data points at the x-axis value 08-23 is not zero or nearly zero in all cases: 130.206.158.142 has the largest value, 2,85 followed by 193.1.33.5 with 0,20. The third largest distance has 204.235.64.14 with 0,096.

Figure 6.11 shows the eigenvalue distance per source interface. The different data points per source represent data of different days from 8th to 23rd of February. One can see that some data points are missing. An example is the data from the source 150.183.95.135 on the very left of the figure, which indicates that the analyzed data do not contain records with the start date 8th of February. One can also see that the distance seems to be quite low for all dates where a data point exists for source 204.235.64.14 (headquarter of US army information systems engineering command, Fort Huachuca, Arizona). The results of day 22 seem quite bad, in general. For almost all sources, the distance is the highest that day. Some sources have quite stable results. The coverage seems to shift in relation to other sources (comparing nodes 113.197.9.170 and 192.42.115.98) with the relation of the values staying roughly the same. However, this is not true for all nodes. Node 128.232.97.9 for example has some days with the lowest distance but also some days with one of the highest distances.

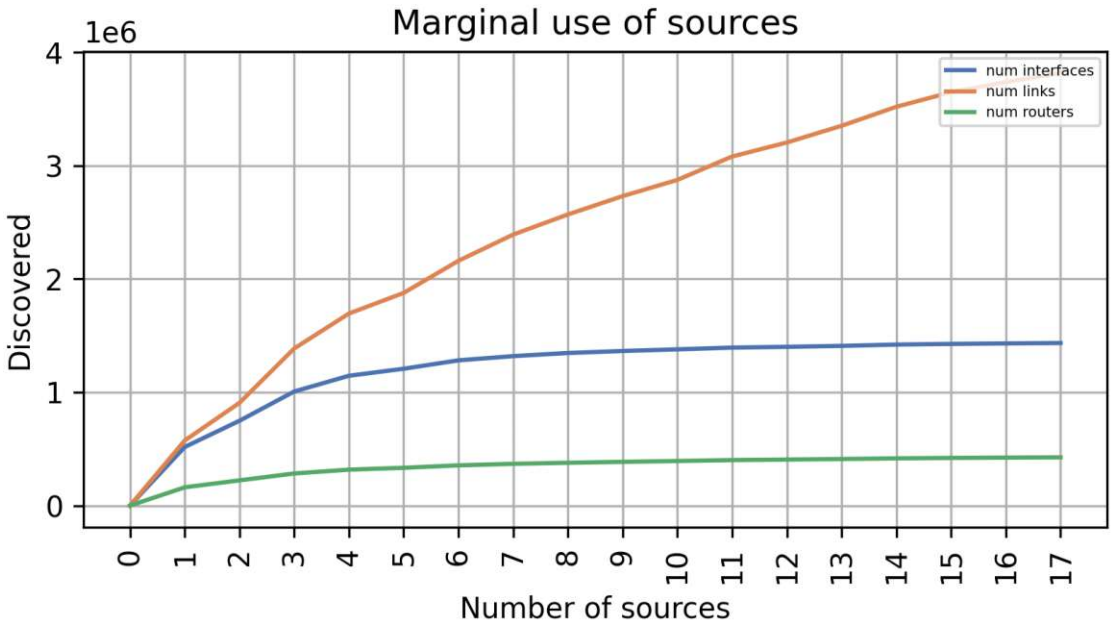

Figure 6.8: Marginal use of source interfaces - plot 3

Figure 6.12 shows the eigenvalue distance (ASD) based on traceroute records in warts files from 16 consecutive days in February 2022. The distance is calculated in relation to the entire graph for that day. The distance clearly decreases with an increasing level number representing the number of graphs. In addition, the interval between the respective minimum and maximum values is shown. When analyzing eigenvalue distance plots for different days, of which only 6.12 is shown here, it can be seen that the range between minimum and maximum values decreases with increasing levels, though not continuously. The range is generally quite centered around the median. The plot also shows the median improvement rate together with its standard deviation. The improvement rate is calculated based on the eigenvalue distance. The formula is defined as follows:

$$\frac{d_i - d_{i-1}}{d_0} * 100, i > 1$$

In human terms, this means that it calculates the decrease of the difference between two consecutive differences relative to the maximum distance. As the distance may only decrease or stay constant as the level increases, the improvement rate cannot be negative. The improvement rate starts around 15 % when looking at the median. The exact further trend of this line differs between the different dates and is largely dependent on the order in which the subgraphs are assembled. Due to the fact that this is done at random, the plot might look different to some extent each time. However, it can be said that the improvement rate is quite stable, meaning that the sources contribute evenly to the reduction of the distance. In the case of the improvement rate, the interval is not centered

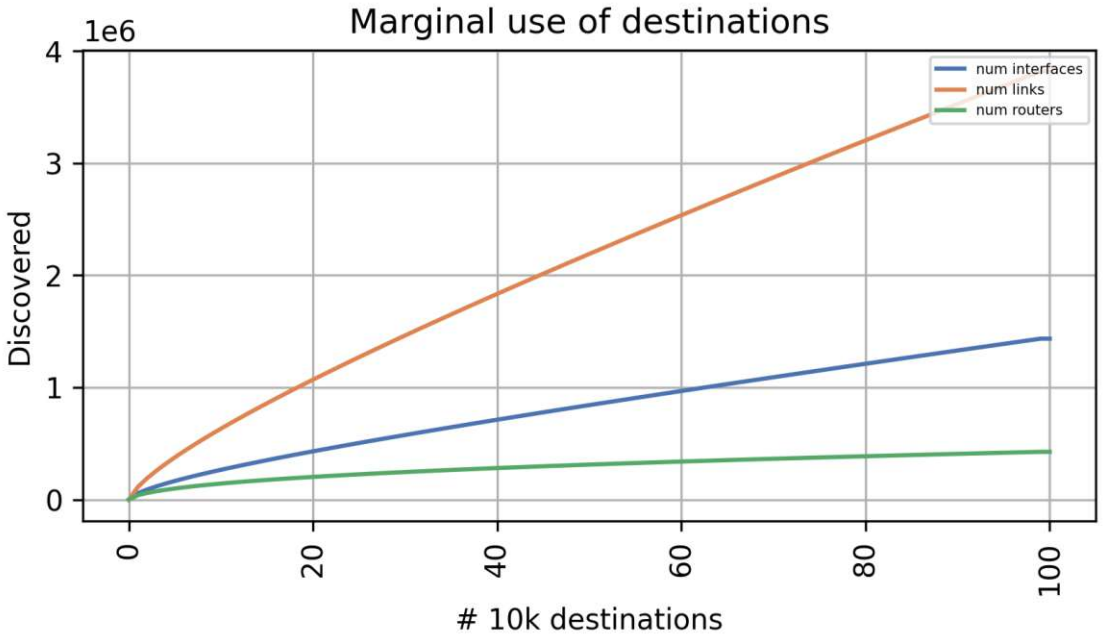

Figure 6.9: Marginal use of destinations - plot 4

around the median. It could be the case that this is a result of outliers. Another possible explanation is that the distribution is skewed towards low values. Given that some plots have a high variation of the improvement rate range with large spikes and small ranges in between, the first explanation makes more sense. Neither the improvement rate itself nor its minimum maximum range decreases in general.

6.2 Effect of the choice of vantage points on the coverage

The data analysis shows that the use of a larger number of VPs in different geographical locations has a positive impact on the coverage. Figures 6.8 to 6.12 show this in different variations. Related work like [2] also observe an increasing coverage when using more VPs. It can be said that the number of VPs and the coverage have a positive correlation. However, not only the geographic diversity is important. Other factors that have an impact are the frequency of scans performed from a certain source address. Yet another factor is the number of days at which the scans are performed. Moreover, the factors also depend on what type of typology is analyzed at which granularity. A distinction should be made between topologies that model links, interfaces, routers, ASs, or other entities.

VPs at different geographical locations show largely different coverage patterns. Figure 6.6 shows that the Universidad de Chile (200.27.115.62) and National Center for Atmospheric Research (NCAR) (193.43.244.202) VPs have a comparably low daily coverage while the total coverage, when taking into account the entire measurement period, is roughly the

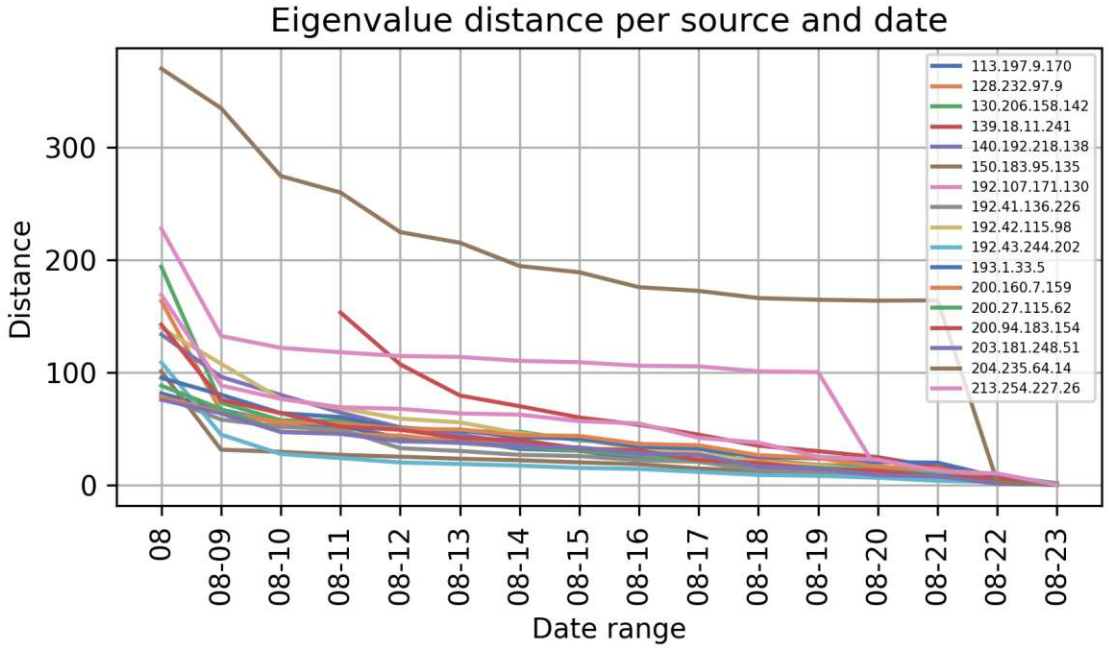

Figure 6.10: Eigenvalue distance per source and date range - plot eigenvalues_1b

same as the results of other VPs included in this analysis. This work does not claim that the differences are due to the geographic location of the VPs. The possible reasons for the lower coverage are not investigated in this work. However, it has been shown that the geographical location of a VP can have a large impact on the coverage. Although VPs can be restricted by the characteristics of the regional network, they still add a valuable contribution to the overall topology [49].

In contrast to the two previously mentioned VPs, the source caida.uni.leipzig.de (139.18.11.241) shows a different pattern. The minimum, maximum and average number of interfaces found are close together, indicating a lower variation than in the case of most other VPs. However, the total coverage achieved by this source is rather low. Sources like this one can be valuable to establish a measurement baseline.

Figure 6.10 visualizes the eigenvalue distance calculated between subgraphs of different sizes against the complete graph. The eigenvalue distance provides insight into the structural properties of the different subgraphs. The subgraphs as well as the complete graph are created per source. A low distance indicates that the structures of the subgraph and the complete graph are similar. Variations of the distance show that some VPs need more scans to build representative subgraphs than others.

When comparing the number of links, interfaces and routers found per day per source,

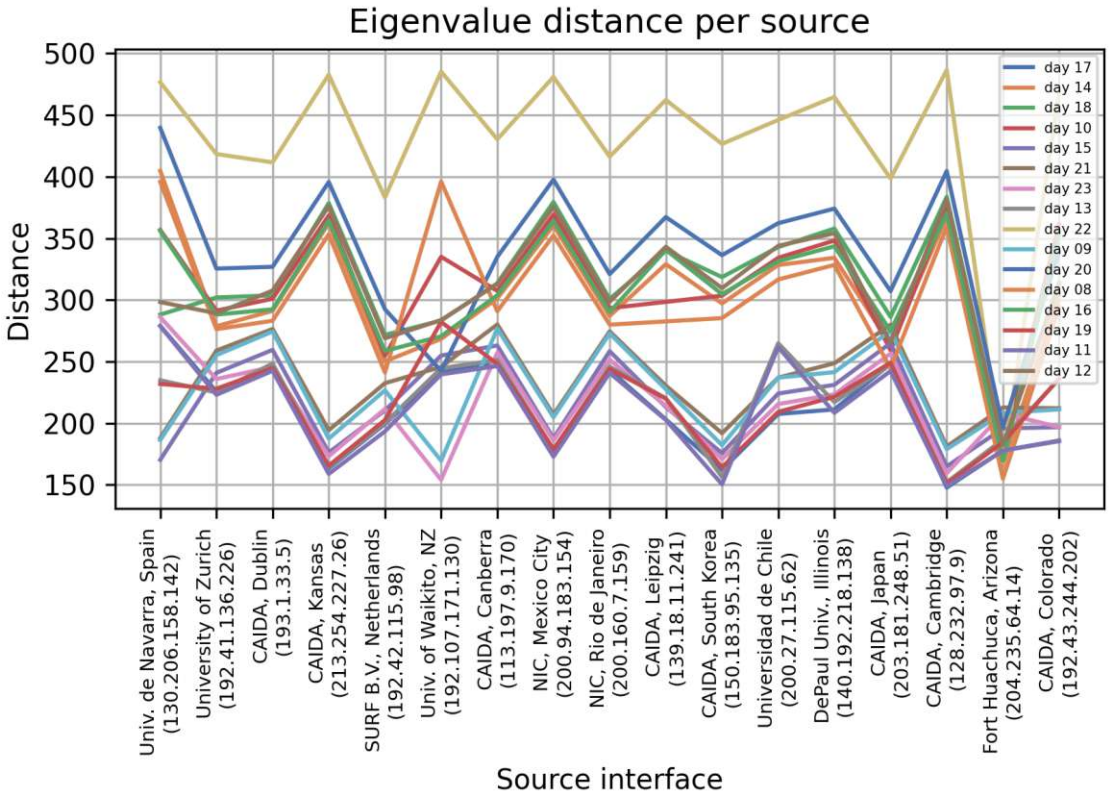

Figure 6.11: Eigenvalue distance per source - plot eigenvalues_2a

one can see that approximately 2.5 links are found for every discovered interface. Figure 6.8, visualizing the marginal coverage of source interfaces shows similar results. Although the marginal coverage decreases early for the number of routers and interfaces, no plateau is reached for the number of links. Related work like [49] observes similar patterns of the discovered links. The observed patterns indicate the use of load balancing, especially per-flow load balancing. However, this hypothesis is not verified in this work.

6.3 Recommendation regarding the choice of vantage points

Depending on the specific goals of the measurement studies, different recommendations can be made based on the results of the analysis. In order to give a scientifically valid recommendation, mainly the already mentioned figures visualizing the median eigenvalue distance and improvement rate together with the minimum and maximum interval are used. Furthermore, based on figure 6.8, figure 6.13 has been created. It visualizes how much the number of interfaces, routers, and links found increases by adding one source at a time. Three thresholds for values 2, 5 and 10% are shown in the figure, too. The

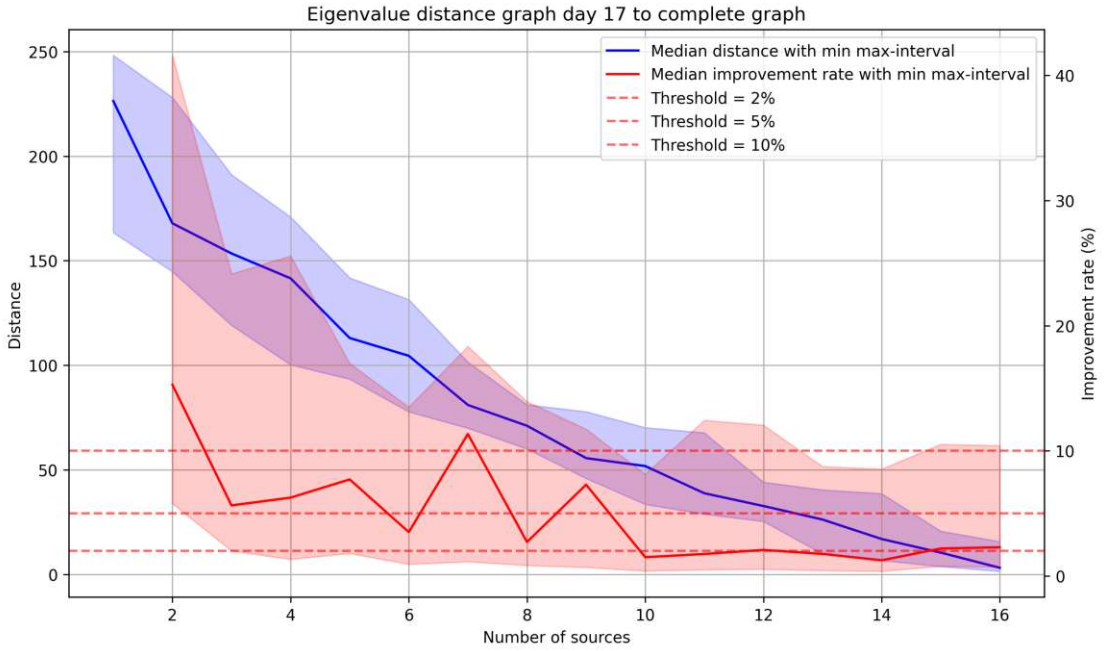

Figure 6.12: Eigenvalue distance day 17 - plot eigenvalues_2b_17_grouped

marginal gain rate starts with the highest value in the case of the number of links, while the number of routers starts with the lowest rate.

In case a basic router level topology is needed, four to five geographically diverse VPs should be sufficient to build a baseline topology. In this case, VPs that show consistent coverage should be prioritized.

In case a representative link layer or other detailed topology is needed, at least 10 to 15 VPs should be used. Here, VPs with different coverage patterns can be chosen. A high PPS rate is not expected to be necessary, especially with a larger number of VPs. It is also deemed reasonable to avoid unnecessary network load by preferring longer-lasting scanning campaigns that perform scans less often instead of those that last a shorter time but perform daily or near-daily scans during this period. VPs with similar patterns and also similar locations might make sense to account for failures, local outages, etc. Little is known about the semantics of the eigenvalue distance in the context of Internet topologies. The pattern of the median improvement rate of the egeinvalue distance differs substantially between the different days analyzed.

When interested in regional topologies, first VPs of that region should be included. Eventual relevant regional network characteristics should be taken into account if possible. The regional VPs should be extended by some stable sources external to the region of

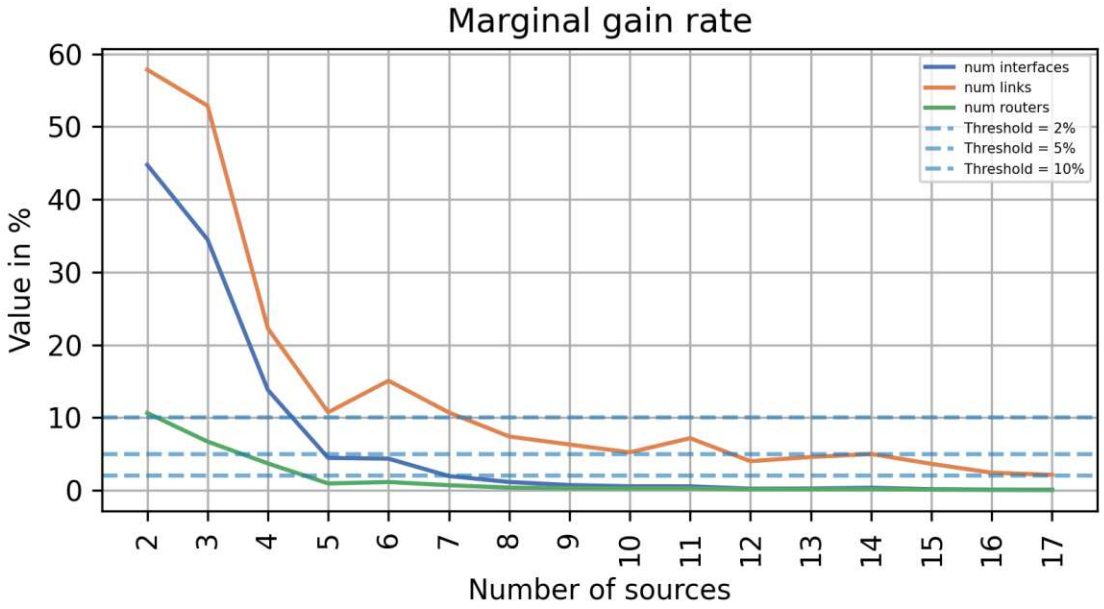

Figure 6.13: Marginal gain rate - plot 3_marginalGainRate

interest.

CHAPTER 7

Conclusion

Due to the complexity of the Internet and the lack of a central authority to manage the topology, discovering the topology is a challenging but crucial task. The Internet is dynamic, decentralized, redundant, and different policies are applied in different parts of the network. Topology discovery tools must face issues such as local outages, load balancing, and inconsistent implementation of networking protocols such as ICMP or BGP.

Knowledge of the Internet's structure allows the relevant parties involved to optimize paths, reducing packet transmission times as well as the occurrences of congestion. Discovery tools scanning the Internet, help in uncovering security vulnerabilities and bad configurations of hosts as well as getting insights into topics such as censorship and the network behavior of different organizations. Internet scans are performed by different organizations using different tools, varying scan periodicities, and a varying number of VPs.

Although it makes sense to perform different scans in order to handle different uses, it is essential to reduce network traffic as much as possible while maintaining good measurement quality. Possible options for achieving this optimization can, for example, be to reduce the scan periodicity, decrease the PPS rate, and reduce the number of VPs.

7.1 Summary

This thesis was motivated by the lack of existing analysis on the effect of the number of VPs on the discovery of Internet topologies. This work investigated the relevance of multiple VPs for Internet topology discovery with a particular focus on the effect on the coverage and structure of the topology.

7.1.1 Methodology and data

Existing data have been combined and processed to perform quantitative and qualitative analyzes. Different graphs have been built and compared using graph comparison methods that evaluate structural changes. The data used are publicly provided by CAIDA and consist of two data sets that contain information on the link and the node in aggregate form. A third part of the used data are warts files containing raw traceroute data created by the scanning tool Scamper. Only data from 8th to 23rd of February 2022 have been analyzed.

7.1.2 Key findings on vantage point discovery

This work verified the hypothesis of a positive correlation between the number of VPs and graph coverage. However, the marginal utility of additional VPs decreases rapidly, especially in the case of interface and router topologies. The relationship is therefore not linear, and many other factors affect the coverage. The type of topology analyzed has an impact on the challenges that discovery tools face, the observable patterns, and the limitations. More detailed analysis and specific recommendations regarding the optimal number of VPs can be found in the chapters 6.2 and 6.3.

7.1.3 Vantage point characteristics and performance

No VP consistently achieves the best or worst result throughout the entire period. The coverage patterns differ substantially between VPs. Different PPS rates have been observed at different VPs. Active measurement techniques have proven to be more effective than passive methods. Some achieve stable but moderate coverage, while others have greater variability.

7.1.4 Influencing factors and practical considerations

The geographical location has an impact on the coverage especially in the case of areas with political influences such as censorship. In addition, network characteristics and practices, such as the use of MPLS tunnels, Traffic Engineering and anonymous routers, further affect the coverage and structure of the topology obtained. The past scanning behavior and the PPS rate of a VP also influence the results, along with the use of the network of other parties. When making the strategic decision to use a specific VP, these factors should be taken into account. It should also be kept in mind that increasing the number of VPs comes at a cost. Among others, it implies an increase in maintenance, monitoring, infrastructure, and communication costs.

7.2 Future work

Although significant efforts have been made to minimize bias and limitations of this work, some constraints were inevitable. Those limitations, which arise from shortcomings of the tools and analytical choices, provide promising opportunities for future research.

7.2.1 Vantage point representation and distribution

The VPs used are by no means representative of the global topology of the Internet. The complete set of characteristics defining a representative VP is not known due to the complex nature of topology discovery and the lack of the ability to perform isolated, repeatable but realistic experiments or simulations. Most of the VPs are in western countries, while entire regions, such as Africa, are not represented by any VP. Future research could address this bias and explore the diversity of VPs as well as the influence of different characteristics such as the network type, the geographical location, and the past scanning history. Combining already available geographic information of network nodes could provide further insight in this field.

7.2.2 Temporal and coverage analysis

The data analyzed in this work are three years old. In regions with frequent topological changes, these data might not adequately represent the current regional topology. Furthermore, the data used covers a short time span of 16 days. Different patterns, such as seasonal effects, might be observable in some parts of the network when analyzing a different time period. Future research could benefit from the analysis and comparison of results from different time periods.

7.2.3 Graph analysis and methodology

The Neo4j graph created in this study can be used as a starting point for additional analysis. The exploration of further, more sophisticated graph comparison methods is an opportunity for future research fields. Those comparison methods could, for example, be spectral graph analysis or make use of machine learning to discover complex patterns in parts of the topology. Another opportunity is to establish metrics specific to the comparison of Internet topologies taking into account characteristics such as the node type, a node's connectivity, centrality, or the type of the topology itself. Examining the network at more granular levels, the community structure, for instance, might allow one to observe patterns that might not be visible from a global view. Analyzing the effects of VPs with multiple IP addresses could provide information on the effects of address diversity on the coverage. Certain challenges, such as flow-based load balancing or blocking, might be mitigated to some extent.

7.2.4 Technical improvements

Improving tooling used for alias resolution, link and interface discovery is crucial to improve the reliability of measurements. The concrete challenges to address include improved approaches to the handling of anonymous routers, load balancing, and MPLS. Advancements in that field could lead to increased precision, as in the case of alias resolution.

7.2.5 Alternative scanning approaches

Due to the absence of knowledge of a complete topology graph, the total coverage cannot be reliably computed. Current research and also this work focus therefore on the marginal coverage instead. Future research might use statistical methods for correlation analysis with respect to the number and characteristics of VPs compared to the coverage achieved. An example of such a method is factor analysis in order to elaborate on the impact that different characteristics have on the topology. Statistical analysis of the marginal utility could help improve deployment strategy guidelines. Further analysis of alternative scanning approaches, such as the combination of active and passive measurements, using more VPs that only scan a certain neighborhood in the network could lead to improvements in scanning efficiency and obtained coverage.

Overview of Generative AI Tools Used

In general, artificial intelligence tools have been used in order to explore new alternative ideas for various problems throughout this work. It has also served as an additional source of input for evaluating chosen approaches.

The following tools have been used for this work:

- ChatGPT (GTP-4o): ChatGPT has been used to find additional literature in the area of graph analysis and eigenvalue computation. The gathered literature has then been evaluated for its relevance to the thesis's topic and its quality. The tool also provided input for brainstorming related to the chapters 5.3, 6, and 7.
- Writefull for Overleaf: Writefull has been used to improve the general wording throughout the thesis, leading to more clarity and a more academic writing style. The proposed text changes, such as the use of alternative words, a different sentence structure, or punctuation, have been reviewed and incorporated to some extent.
- Claude (Sonnet 3.7-4): Claude has been used for code optimization regarding readability, the removal of errors, and the usage of alternative algorithmic approaches. The suggested changes have been rigorously reviewed and tested before completely incorporating them.

List of Figures

List of Tables

Glossary

Autonomous System An AS is a collection of routers administered by a single technical entity. One or multiple IGPs are used to define how packets should be routed through the network of the AS. To systems external to the AS, the AS appears to have a single coherent routing plan [25].. 2

Cypher Cpyher is a language used for performing queries and updates of graph databases, especially Neo4j databases. By making use of the indexes defined on a graph database, Cypher provides a performant access to the desired data. This query language allows one to query for complex relationships [65]. Extension Neo4j procedures and functions offer the possibility of calculating graph metrics like the node degrees or defining a certain neighborhood space to name some [66].. 55

Neo4j Neo4j belongs to the broad group of NoSQL databasese. More specifically, it is a graph database. As such, Neo4j is able to store a graph representation. While this is also possible with classical relational databases, models graph databases have the following key differences: the nodes and the relationships between them are the basic concept. The structure of a Neo4j graph database is more flexible than in the case of relational databases. Multiple properties can be defined on relationships in the form of key-value pairs. Neo4j guarantees the common reliability principles: atomicity, consistency, isolation, and durability, and supports transactional operations [67]. Graph databases are used in a broad set of use cases: fraud detection, network management, social networks, software system analysis, recommendation engines, and more [65].. 45, 46, 48

Round Trip Time The RTT is an important function in Internet measurements. It is defined as the time span that a packet needs from the point in time where it is sent to the receipt of the packet acknowledgment. The RTT is mainly relevant for reliable transport protocols such as TCP (see Chapter 3.1.6). Those protocols resend a packet when the acknowledgment has not arrived until a certain threshold. This threshold is based on an estimated RTT value [68].. 34

Traffic Engineering Traffic Engineering deals with the optimization of routes in ASs. Several different objectives can be the goals. Some of them are: congestion

prevention, optimize resource usage, packet loss, enforce Service-level agreement (SLA)s. In order to do this, the maintainer of the AS has to monitor the network, adjust routing options, and resource allocations to align the system state with the goals. The shortest path algorithms used by the IGP are rather simple and often lead to congestion in certain cases. Among others, MPLS can be used to configure sophisticated traffic engineering solutions [69]. 25, 38, 70

Acronyms

API Application Programming Interface. 41

AS Autonomous System. 2, 22, 25, 26, 37, 38, 41, 64, 79, 80, *Glossary:* Autonomous System

ASCII American Standard Code for Information Interchange. 16

ASD Adjacency Spectral Distance. 29, 63

BGP Border Gateway Protocol. 13, 22, 25, 37, 40, 44, 69

CAIDA Cooperative Association for Internet Data Analysis. 13, 33, 43, 44, 52, 54, 70

CIDR Classless Inter-domain Routing. 18, 19

CLI Command line interface. 46

CRISP-DM Cross Industry Standard Process for Data Mining. ix, xi, xiv, 5, 10, 11, 52, 53

CRS Compressed Row Storage. 50

CSV Comma-Separated Values. 45, 46

DNS Domain Name System. 15–17, 26

FEC Forwarding Equivalence Class. 20, 23

GARR Italian research and education network. 42

HTTP Hypertext Transfer Protocol. 18

IANA Internet Assigned Numbers Authority. 19

ICANN Internet Corporation for Assigned Names and Numbers. 16

ICMP Internet Control Message Protocol. 17, 20, 26, 33–37, 42–44, 69

ID Identifier. 38–40

IGP Interior Gateway Protocol. 22, 79, 80

IP Internet Protocol. xiii, 1, 13, 17–26, 33–35, 37–40, 42–45, 48, 55, 56, 59, 60, 62, 71, 77

ISP Internet Service Provider. 24–26, 37, 38, 40, 56

KNC Known Node-Correspondence. 30

LIR Local Internet Registry. 19

LSD Laplacian Spectral Distance. 29

LSE Label stack entry. 23

LSP Label Switching Path. 24

LSR Label Switching Router. 23

LSRR Loose Source and Record Route. 18, 20

MB Megabyte. 42

MBT Monotonic Bounds Test. 39

MDA Multi-path Detection Algorithm. 41, 43

mDNS Multicast Domain Name System. 16, 17

MPLS Multiprotocol Label Switching. 2, 13, 23, 25, 38, 70, 72, 80

MSB Most significant bit. 18, 20

NCAR National Center for Atmospheric Research. 64

NLSD Normalized Laplacian Spectral Distance. 29

NRLI Network Reachability Information. 22

OS Operating System. 34, 35

PMTU Path Maximum Transmission unit. 20

PoP Point-of-Presence. 2, 24, 26, 42

PPS packets per second. 54, 57, 67, 69, 70

PRF Pseudorandom function. 21, 22

RAM Random-Access Memory. 47, 53

RIB Routing Information Base. 22, 37

RIR Regional Internet Registry. 19

RTT Round Trip Time. 34, 35, 41, 79, *Glossary:* Round Trip Time

SLA Service-level agreement. 80

SLR Systematic Literature Review. 8–10, 52

SSH Secure Shell. 47

SSRR Strict Source and Record Route. 18

TCB Transmission Control Block. 21

TCP Transmission Control Protocol. 17, 21, 22, 24, 34, 35, 42, 43, 79

TLD Top-Level Domain. 16

TTL Time-to-Live. 17, 20, 23, 34–36, 42

UDP Uniform Data Protocol. 17, 21, 24, 34, 35, 37, 41–43

VP Vantage Point. 1–3, 11, 13, 25, 40, 41, 55, 64, 65, 67, 69–72

Bibliography

[1] B. Donnet and T. Friedman, "Internet topology discovery: a survey," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 4, pp. 56–69, 2007.

[2] P. Barford, A. Bestavros, J. Byers, and M. Crovella, "On the marginal utility of network topology measurements," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pp. 5–17, 2001.

[3] J. Klick, S. Lau, M. Wählisch, and V. Roth, "Towards better internet citizenship: Reducing the footprint of internet-wide scans by topology aware prefix selection," in *Proceedings of the 2016 Internet Measurement Conference*, pp. 421–427, 2016.

[4] J. L. Mackie, *The Cement of the Universe: A Study of Causation*. Oxford, England: Oxford, Clarendon Press, 1974.

[5] T. D. Cook, D. T. Campbell, and W. Shadish, *Experimental and quasi-experimental designs for generalized causal inference*. Houghton Mifflin Boston, MA, 2002.

[6] Y. Levy and T. J. Ellis, "A guide for novice researchers on experimental and quasi-experimental studies in information systems research," *Interdisciplinary Journal of information, knowledge, and management*, vol. 6, p. 151, 2011.

[7] D. I. Sjøberg, J. E. Hannay, O. Hansen, V. B. Kampenes, A. Karahasanovic, N.-K. Liborg, and A. C. Rekdal, "A survey of controlled experiments in software engineering," *IEEE transactions on software engineering*, vol. 31, no. 9, pp. 733–753, 2005.

[8] B. Curtis, "Measurement and experimentation in software engineering," *Proceedings of the IEEE*, vol. 68, no. 9, pp. 1144–1157, 1980.

[9] G. H. Travassos, P. S. M. dos Santos, P. G. Mian, P. G. M. Neto, and J. Biolchini, "An environment to support large scale experimentation in software engineering," in *13th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2008)*, pp. 193–202, IEEE, 2008.

[10] E. S. Engineering. `http://lens-ese.cos.ufrj.br/ese`. Last accessed on December 10, 2024.

[11] S. Keele *et al.*, "Guidelines for performing systematic literature reviews in software engineering," 2007.

[12] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, "Lessons from applying the systematic literature review process within the software engineering domain," *Journal of systems and software*, vol. 80, no. 4, pp. 571–583, 2007.

[13] R. Wirth and J. Hipp, "Crisp-dm: Towards a standard process model for data mining," in *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, vol. 1, pp. 29–39, Manchester, 2000.

[14] A. for Computing Machinery. `https://dl.acm.org/`. Last accessed on December 10, 2024.

[15] IEEE. `https://ieeexplore.ieee.org/`. Last accessed on December 10, 2024.

[16] IEEE. `https://ieeexplore.ieee.org/Xplorehelp/searching-ieee-xplore/command-search#summary-of-data-fields`. Last accessed on July 10, 2023.

[17] T. C. UCSD. `https://www.caida.org/`. Last accessed on December 10, 2024.

[18] P. V. Mockapetris, "Rfc1035: Domain names-implementation and specification," 1987.

[19] P. Hoffman and K. Fujiwara, "Rfc 9499: Dns terminology," 2024.

[20] "Internet Protocol." RFC 791, Sept. 1981.

[21] V. Fuller and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan." RFC 4632, Aug. 2006.

[22] B. Hinden and D. S. E. Deering, "Internet Protocol, Version 6 (IPv6) Specification." RFC 2460, Dec. 1998.

[23] D. S. E. Deering and B. Hinden, "IP Version 6 Addressing Architecture." RFC 4291, Feb. 2006.

[24] D. S. E. Deering, J. McCann, and J. Mogul, "Path MTU Discovery for IP version 6." RFC 1981, Aug. 1996.

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

[26] "Internet Control Message Protocol." RFC 792, Sept. 1981.

[27] "User Datagram Protocol." RFC 768, Aug. 1980.

[28] L. Eggert, G. Fairhurst, and G. Shepherd, "UDP Usage Guidelines." RFC 8085, Mar. 2017.

[29] W. Eddy, "Transmission Control Protocol (TCP)." RFC 9293, Aug. 2022.

[30] A. Viswanathan, E. C. Rosen, and R. Callon, "Multiprotocol Label Switching Architecture." RFC 3031, Jan. 2001.

[31] J.-R. Luttringer, Y. Vanaubel, P. Mérindol, J.-J. Pansiot, and B. Donnet, "Let there be light: Revealing hidden mpls tunnels with tnt," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 1239–1253, 2020.

[32] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with paris traceroute," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pp. 153–158, 2006.

[33] B. Augustin, T. Friedman, and R. Teixeira, "Measuring load-balanced paths in the internet," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 149–160, 2007.

[34] G. Tilch, T. Ermakova, and B. Fabian, "A multilayer graph model of the internet topology," *Int. J. Netw. Virtual Organ.*, vol. 22, p. 219–245, jan 2020.

[35] E. Marechal, P. Mérindol, and B. Donnet, "Isp probing reduction with anaximander," in *International Conference on Passive and Active Network Measurement*, pp. 441–469, Springer, 2022.

[36] M. Kwon and S. Fahmy, "Topology-aware overlay networks for group communication," in *Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '02, (New York, NY, USA), p. 127–136, Association for Computing Machinery, 2002.

[37] P. Wills and F. G. Meyer, "Metrics for graph comparison: a practitioner's guide," *Plos one*, vol. 15, no. 2, p. e0228728, 2020.

[38] M. E. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, vol. 74, no. 3, p. 036104, 2006.

[39] S. P. Borgatti, "Centrality and network flow," *Social networks*, vol. 27, no. 1, pp. 55–71, 2005.

[40] M. Tantardini, F. Ieva, L. Tajoli, and C. Piccardi, "Comparing methods for comparing networks," *Scientific reports*, vol. 9, no. 1, p. 17557, 2019.

[41] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal*, vol. 23, no. 2, pp. 298–305, 1973.

[42] T. C. UCSD. `https://www.caida.org/catalog/software/skitter/`. Last accessed on December 10, 2024.

[43] M. Kerrisk. https://man7.org/linux/man-pages/man8/traceroute.8.html. Last accessed on December 10, 2024.

[44] V. Paxson, "End-to-end routing behavior in the internet," *IEEE/ACM transactions on Networking*, vol. 5, no. 5, pp. 601–615, 1997.

[45] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," in *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '02, (New York, NY, USA), p. 133–145, Association for Computing Machinery, 2002.

[46] A. Marder, M. Luckie, A. Dhamdhere, B. Huffaker, K. Claffy, and J. M. Smith, "Pushing the boundaries with bdrmapit: Mapping router ownership at internet scale," in *Proceedings of the Internet Measurement Conference 2018*, pp. 56–69, 2018.

[47] K. Keys, Y. Hyun, M. Luckie, and K. Claffy, "Internet-scale ipv4 alias resolution with midar," *IEEE/ACM Transactions on Networking*, vol. 21, no. 2, pp. 383–399, 2012.

[48] A. Faggiani, E. Gregori, L. Lenzini, S. Mainardi, and A. Vecchio, "On the feasibility of measuring the internet through smartphone-based crowdsourcing," in *2012 10th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, pp. 318–323, 2012.

[49] Y. Shavitt and U. Weinsberg, "Quantifying the importance of vantage points distribution in internet topology measurements," in *IEEE INFOCOM 2009*, pp. 792–800, IEEE, 2009.

[50] Y. Shavitt and E. Shir, "Dimes: Let the internet measure itself," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 71–74, 2005.

[51] M. Luckie, "Scamper: a scalable and extensible packet prober for active measurement of the internet," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 239–245, 2010.

[52] T. C. UCSD, "IPv4 Prefix-Probing Traceroute Dataset." https://publicdata.caida.org/datasets/topology/ark/ipv4/prefix-probing/2022/02/, Apr. 2024.

[53] T. C. UCSD, "Ipv4 prefix-probing traceroute dataset." https://www.caida.org/catalog/datasets/ipv4_prefix_probing_dataset/. Last accessed on December 30, 2024.

[54] T. C. UCSD, "Macroscopic Internet Topology Data Kit (ITDK)." https://publicdata.caida.org/datasets/topology/ark/ipv4/itdk/2022-02/midar-iff.nodes.bz2, Apr. 2024.

[55] T. C. UCSD, "Macroscopic Internet Topology Data Kit (ITDK)." https://publicdata.caida.org/datasets/topology/ark/ipv4/itdk/2022-02/midar-iff.links.bz2, Apr. 2024.

[56] T. C. UCSD, "Macroscopic internet topology data kit (itdk)." https://www.caida.org/catalog/datasets/internet-topology-data-kit/release-2022-02/. Last accessed on December 30, 2024.

[57] B. Jonglez, "scamper-pywarts." https://github.com/drakkar-lig/scamper-pywarts, Aug. 2016.

[58] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst, *Templates for the solution of linear systems: building blocks for iterative methods*. SIAM, 1994.

[59] D. C. Lay, *Linear algebra and its applications*. Pearson Education India, 2003.

[60] A. Hagberg, P. J. Swart, and D. A. Schult, "Exploring network structure, dynamics, and function using networkx," tech. rep., Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 2008.

[61] G. Csardi and T. Nepusz, "The igraph software," *Complex syst*, vol. 1695, pp. 1–9, 2006.

[62] I. Amazon Web Services. https://aws.amazon.com/de/neptune/. Last accessed on August 12, 2025.

[63] J. Webber, "A programmatic introduction to neo4j," in *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity*, pp. 217–218, 2012.

[64] G. Wan, L. Izhikevich, D. Adrian, K. Yoshioka, R. Holz, C. Rossow, and Z. Durumeric, "On the origin of scanning: The impact of location on internet-wide scans," in *Proceedings of the ACM Internet Measurement Conference*, pp. 662–679, 2020.

[65] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor, "Cypher: An evolving query language for property graphs," in *Proceedings of the 2018 international conference on management of data*, pp. 1433–1445, 2018.

[66] I. Neo4j. https://neo4j.com/labs/apoc/4.1/overview/. Last accessed on December 27, 2024.

[67] F. M. S. López and E. G. S. De La Cruz, "Literature review about neo4j graph database as a feasible alternative for replacing rdbms," *Industrial Data*, vol. 18, no. 2, pp. 135–139, 2015.

[68] P. Karn and C. Partridge, "Improving round-trip time estimates in reliable transport protocols," *ACM SIGCOMM Computer Communication Review*, vol. 17, no. 5, pp. 2–7, 1987.

[69] J. McManus, J. Malcolm, M. D. O'Dell, D. O. Awduche, and J. Agogbua, "Requirements for Traffic Engineering Over MPLS." RFC 2702, Sept. 1999.