Internet Topology: From the Discovery Process to the Real Picture

Rami Cohen

Technion - Computer Science Department - Ph.D. Thesis PHD-2007-18 - 2007

Internet Topology: From the Discovery Process to the Real Picture

Research Thesis

Submitted in partial fulfillment of the requirements

for the degree pf Doctor of Philosophy

Rami Cohen

Submitted to the Senate of
the Technion - Israel Institute of Technology
Kislev 5768 Haifa December 2007

The research was done under the supervision of Prof. Danny Raz in the department of Computer Science.

I would like to thank Danny for his helpful and devoted guidance during all research stages.

The generous financial help of the Technion is gratefully acknowledged.

Contents

List of Figures

Abstract

The topological structure of the Internet infrastructure is an important and interesting subject that attracted significant research attention in the last few years. Apart from the pure intellectual challenge of understanding a very big, complex, and ever evolving system, knowing the structure of the Internet topology is very important for developing and studying new protocols and algorithms. Starting with the fundamental work of Falostous et al., a considerable amount of research was done recently in this field, improving our knowledge and understanding of the Internet structure. The work in this area composed of collecting information regarding the current (and possibly past) state of the Internet, inferring from the collected data the actual topological and the hierarchy structure, and analyzing this structure in order to understand the inherently important characteristic and evolution of the system.

In this research we focus on the Autonomous System (AS) level. First, we question the basic problem: how big is the Internet. In the AS level this means: how many peering relations exist between ASes. Finding this number is hard since there is no direct way to retrieve information from all nodes regarding their direct neighbors, and all our knowledge is based on sampling processes. Thus, it is very difficult to characterize the Internet since it may well be the case that this characterization is a result of the sampling process, and it does not hold for the "real" Internet. In the first part of this thesis we present strong evidence to the fact that a considerable amount (at least 35%) of the links in the AS level are still to be unveiled. Our findings indicate that almost all these missing links are of type *peer-peer*. We also examine the vertex degree distribution of the AS connectivity map, and show that while the *customer-provider* subgraph follow the power law, the *peer-peer* subgraph behave differently.

Next, we study another interesting problem, the Type of Relationship (*ToR*) problem. In this problem, given an AS connectivity graph and a set of paths, the goal is to assign relationship of *customer-provider* or *peer-peer* to AS links such that pre-defined policy restrictions are not violated. It turns out that the original *ToR* problem does not reflect the full nature of the commercial relationship between connected ASes. The main problem with the current definition is that hierarchical cycles may be formed, while this is impossible in the real internet. To address this point, we define a new problem called Acyclic Type of Relationship, *AToR*. We present an efficient algorithm determining whether an acyclic solution without invalid paths exists, and we show that the general decision version of the problem is NP-hard. We also present $\frac{2}{3}$-approximation algorithm for a maximum version of the problem, and consider practical aspects of inferring the actual type of relationships between ASes. This includes heuristics to infer also *peer-peer* and *sibling-sibling* relationships. We

support our approach by experimental and simulation results showing that our algorithms classify the type of relationship between ASes much better than all previous algorithms.

Finally, in the last part of the thesis, we study the path inflation phenomenon in which, due to BGP routing policy, routing between ASes is not necessarily done along shortest paths. To overcome this problem and to route packets over shortest paths after all, we consider an intermediate routing scheme. In this routing scheme routing between clients can be done via a set of relay servers located in intermediate nodes. We study this problem as an optimization problem where the objective target is to enable routing through shortest paths with a minimum number of relay servers. We show that this problem is NP-hard, present a $O(\log(n))$ approximation algorithm for it, and show that this is the best approximation one can get. We examine the practical aspects of the scheme by evaluating the gain one can get over real up-to-date data reflecting the current BGP routing policy in the Internet. We show that a relative small number of relay servers are sufficient to enable routing over the shortest paths between almost all considered nodes, which reduces the average path length of these inflated paths by 40%.

Chapter 1

Introduction

The topological structure of the Internet infrastructure is an important and interesting subject that attracted significant research attention in the last few years. Apart from the pure intellectual challenge of understanding a very big, complex, and ever evolving system, built by people and used by many more, knowing the structure of the Internet topology is very important for developing and studying new protocols and algorithms.

The work in this area composed of collecting information regarding the current (and possibly past) state of the Internet [4, 1, 45], inferring from the collected data the actual topological and the hierarchical structure [22, 14], and analyzing this structure in order to understand the inherently important characteristics and evolution of the system [24, 46, 26, 11]. One problem that arises in this context is that it is impossible to measure the full structure of the Internet directly and to obtain the full connectivity graph. This is not only due to the fact that the Internet is too big, but largely because there is no direct way to retrieve information from a node regarding its direct neighbors. Thus, it is very difficult to characterize the Internet since it may well be the case that this characterization is a result of the sampling process, and it does not hold for the "real" Internet. In other words, it is extremely difficult to know if the picture we have is a good approximation of the "real" Internet connectivity, or if it is biased to a large extend by the measurements, and thus does not reflect the "true" picture.

The current Internet consist of more than 20,000 ASes (Autonomous Systems) where each AS is a collection of routers under a single administrative authority. Routing inside an AS is done using a common Interior Gateway Protocol (IGP) such as OSPF [40], while routing between the different ASes is done using the Border Gateway Protocol (BGP) [43]. One of the well appreciated advantages of BGP is its ability to use policy based routing where each AS defines its own local policy. In practice, the policy of an AS reflects its commercial relationship with other ASes. Thus, the AS connectivity map has a hierarchical structure in which connected ASes have *customer-provider* relationship if a small AS is connected to a larger AS, and they have *peer-peer* relationship if they have comparable size (other types of relationship such as *sibling-sibling* also exist, but they apply to less than 2% of the connections [24]) [31, 7].

Several running projects and databases are used, among other things, to collect information regarding the AS connectivity graph. The Route-Views project [4] is a BGP based database that collects a snapshot of the Internet AS level topology on a daily basis, compos-

ing BGP routing tables from over 40 different sources. The Internet Routing Registry [1] is a union of world-wide routing policy databases that contains the local connectivity for the registered ASes. In other words, for each registered AS, the database contains all its adjacent ASes. The DIMES project [45] samples the Internet using distributed agents located in thousands hosts around the world, performing periodic *traceroute* to a set of IP addresses. While the union of the information from all these projects give more complete view on the AS connectivity graph, still many links remain hidden [14].

In parallel to the practical effort to build a more complete database, some theoretical works have been done to model the hierarchy structure and the connectivity map of the ASes. This effort started in the fundamental paper of Faloutsos et al., that studied the topology structure of the Internet [22]. Faloutsos et al. showed that despite of the apparent randomness of the Internet, simple power-law holds for the Internet both in the AS level and in the router level. This novel observation was adopted by many works that proposed algorithms that generate such a power-law graphs in order to describe the AS connectivity map (e.g. see [8]). On the other hand, other works have suggested different models to describe the connectivity map at the AS level. For example, models based on [49] consider the geographical location of ASes such that the probability that two ASes are connected is relative to their geographical distance. In addition, some models combine these approaches in different ways (e.g. see [13]).

The information that is gathered by projects like IRR, Route-Views and DIMES is also incomplete in the sense that the type of relationship between connected ASes is not part of the collected data. This incompleteness is due to the fact that many ASes are not willing to expose their commercial relationship, or due to the fact that this information is gathered from *traceroute* queries and BGP resources that consist of a set of path vectors. Thus, in order to get a more complete view, one should infer these relationships from the collected information. This is usually done using guidelines and assumptions regarding the policy used and knowledge regarding the gathered information. For instance, the Internet Routing Registry [1] is a union of world-wide routing policy databases that use the Routing Policy Specification Language (RPSL) [6, 39]. These databases contain, among other things, the local connectivity and the local import/export policy of the registered ASes. In [46] the authors analyzed the RPSL policies of ASes in the IRR and inferred the type of relationship between registered ASes. In Chapter 2 of this thesis we consider a variant of this method and we study some of its practical problems. Nevertheless, using the IRR database to infer the hierarchical structure of the AS connectivity map has several drawbacks. First, in some cases, entries in the IRR may be invalid and contain out-of-date data [14]. Second, this database is not complete enough. In particular, only about third of the ASes, most of them are located in Europe, are fully covered by this database.

While the IRR database contains the local policy of registered AS, it is not part of the information gathered by other projects. In these cases, other techniques should be used in order to infer the type of relationships. While databases that consist of BGP routing table such as the Route-Views database, and databases that consist of *traceroute* queries such as DIMES use different methods to collect the data, both databases consist of a set of routing paths (between ASes) that reflect the routing policy of these ASes. Thus, in order to infer the type of relationship from these routing paths, one should understand how the policy in the AS level affects these routing paths.

According to the guidelines presented in [7] and in [31] an AS usually does not export its provider or peer routes to other providers or peers. This policy indicates that BGP paths are valley-free, and step-free, i.e., after traversing a *provider-customer* or a *peer-peer* link, the path cannot traverse a *customer-provider* or *peer-peer* link [24].

Gao [24] was the first to infer the AS relationships from BGP routing tables, based on this valley-free nature of the routing paths. She developed a heuristic algorithm assuming that typically a provider has a larger size than its customer, and the size of an AS is usually proportional to its degree in the AS level connectivity graph. Thus, for each routing path, the AS with the highest degree is set as a *top provider* with respect to the path, inducing *customer-provider* relationship to preceding and subsequent links in the path. The experimental results of [24] indicate that 90% of the links in the Route-Views database are of type *customer-provider*, 8% are of type *peer-peer*, and 1.5% are of type *sibling-sibling*.

Subsequently, the Type of Relationship *ToR* problem was formally defined in [48] as a maximization problem. The technique proposed in [48] to solve the *ToR* problem combines data from multiple vantage points, where each BGP routing table gives partial view of the Internet from one AS. This technique does not rely on the degree of the ASes. The authors ranked the ASes based on their position in the graph, induced by a single BGP routing table. Then they infer the relationship by comparing the ranks of ASes as it derived from multiple sources. In [11] the authors showed that the decision version of the *ToR* problem is NP-complete in the general case. Moreover, they presented a linear time algorithm that determines if there is a fully valid solution (i.e without any invalid path). This algorithm maps a *ToR* instance into a *2SAT* formula by converting every two consecutive edges in any of the paths into a clause with two literals. Finding a Truth assignment to this formula induces a valley-free solution, while if the formula cannot be satisfied then the *ToR* instance must contain at least one valley. They also proved that the maximum version of the problem (i.e., maximizing the number of valid paths) cannot be approximated within $1/n^{1-\epsilon}$ (for any $\epsilon > 0$) for general instances with n paths unless NP=co-RP. This is done using approximation-preserving polynomial reduction from the Maximum Independent Set problem [28].

The type of relationship problem is an example to the effort done so far to study and understand the topological structure of the Internet. Nevertheless, the real picture is far from being well understood. Our research goal is this thesis is studying the topology structure of the Internet and its discovery mechanism. In particular, we focus on the AS connectivity and the relationship between ASes. Thus, the AS level connectivity map is modeled by a graph $G = \{V, E\}$. Each node in the graph represents an autonomous system, and an edge represents a peering relation between two ASes. In Chapter 2 we suggest a novel usage of the measurements themselves in order to infer information regarding the whole system. In other words, rather than looking at the overall graph that is generated from the union of the data obtained by performing many measurements, we consider the actual different measurements and the amount of new data obtained in each of them with respect to the previous collected data. This technique allows us to reach conclusions regarding the structure of the system we were measuring. In our case we apply the methods to the Autonomous System (AS) level connectivity graph.

We start with a rigorous study of the different data collecting techniques that are used in order to collect AS connectivity information. We then use the characterization of the data collection to draw conclusions regarding the actual structure of the full AS map. We

consider a new measure for graphs, the number of (policy-based) shortest path spanning trees needed to cover the edges of a graph and show that the AS map is unique in the sense that a considerable amount of links are not revealed in this covering process. We also show that this unrevealed links are mostly of the type *peer-peer* while this process unveiled many of the *customer-provider* peers. We also analyze the routing policy of available database using concepts that has been presented in [46] and [24], and we settle a fundamental difference between the results that have been presented in these works. While the size of the full AS connectivity map is unknown, we show several methods to estimate this size. In particular, we use a data from different database to approximate the actual number of missing links and present a strong evidence to the fact that at least 35% of the links in the AS level are still to be unveiled. We also examine the vertex degree distribution of the AS connectivity map. we explain the difference between earlier results that show that the AS connectivity map follow the power law [22] and other results that question this observation [14], by showing that while the *customer-provider* subgraph follow the power law, the *peer-peer* subgraph behave differently.

As mentioned above, the *ToR* problem was formally defined in [48] as follows:

Definition 1.0.1 *Given an undirected graph $G = (V, E)$, and a set of paths P, label the edges in E as either $-1, 0$ or $+1$ to maximize the number of valid paths in P, where a valid path can be one of the following types for $M, N \geq 0$:*

1. $-1, \ldots$ *(N times),* $+1, \ldots$ *(M times).*
2. $-1, \ldots$ *(N times),* $0, +1, \ldots$ *(M times).*

Here -1 *indicates a* customer-provider *edge,* 0 *indicates a* peer-peer *edge, and* $+1$ *indicates a* provider-customer *edge.*

For example, consider the instance of the *ToR* problem depicted in Fig. 1.1. This instance consists of nine ASes and two paths. A possible solution to that instance, containing only valid paths, is depicted in Fig. 1.2[1]. In this solution both paths are of type 1, namely they consist of several *customer-provider* links followed by several *provider-customer* links.

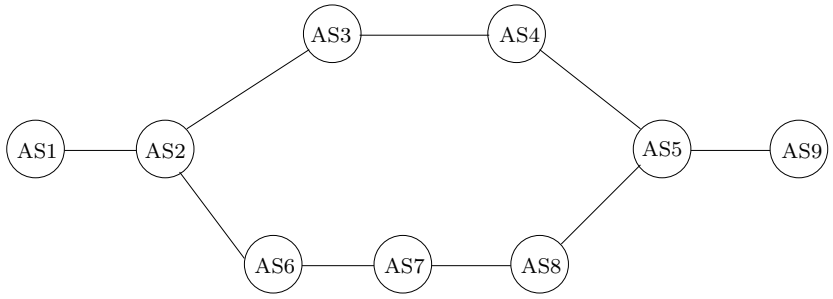


Figure 1.1: A *ToR* instace with two paths

[1] A directed edge in the graph going from node v to node u means that v is a customer of u.

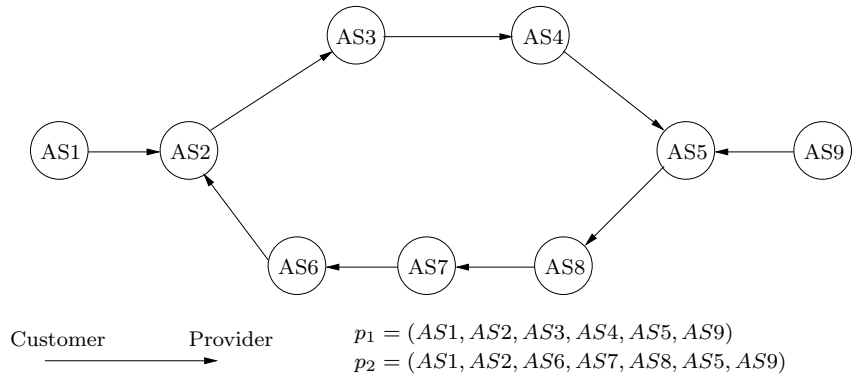


Figure 1.2: A *ToR* instace - A possible solution

The formal definition of the *ToR* problem indeed captures the fact that BGP paths are valley-free and step-free. However, it has an inherent drawback - it does not consider the hierarchical structure of the AS graph. In particular, in the real Internet the directed graph imposed by the assignment of the *customer-provider* relationship can **not** contain cycles (see [26, 25]). A national AS, for example, that provides services to a regional AS that provides services to a local AS cannot be also the customer of the local AS. Consider the solution to the *ToR* instance depicted in Fig. 1.2. In this solution both paths are valley-free, however it contains a directed cycle $(AS2, AS3, AS4, AS5, AS8, AS7, AS6, AS2)$, violating the hierarchical structure of the graph. One can achieve an acyclic solution to this specific instance, by changing the direction of the edge $(AS4, AS5)$. In contrast, every optimal solution to the *ToR* instance depicted in Fig. 1.3 contains a cycle[2].

In Chapter 3 we address this drawback by defining a new problem, the Acyclic Type of Relationship (*AToR*) problem, taking into account the acyclic structure of the AS connectivity graph. In this case, given a set of routing paths, the objective function is to maximize (or minimize) the number of valid (invalid) paths, keeping the directed graph acyclic. This new problem captures the type of relationship between connected ASes more accurately. Note, that while these two problems look similar, their analysis is quite different. In particular, in the *ToR* problem one should only satisfy local conditions in which every two consecutive edges in all the paths should be valley-free. On the other hand, in the new *AToR* problem, in addition to these local conditions, one should satisfy a more global condition ensuring that the assignment is acyclic. For that reason, techniques and algorithms that have been used with respect to the *ToR* problem, cannot be adopted and used to analyze and solve the *AToR* problem. A very similar problem, also termed AToR, was independently defined and studied by Kosub et al. [36]. This work, done in parallel to ours, studies only pure theoretical aspects of this problem.

After defining the *AToR* problem, we present an efficient algorithm determining whether an acyclic solution without invalid paths exists. Then we discuss the general case and consider a variant of this problem in which the objective function is to minimize the total number of valleys. This variant captures the fact that in some cases the export policy

[2]To dismiss the cycle $(AS1, AS2, ..., AS8, AS1)$, at least one vertex must have two outgoing edges, namely both edges attached to that vertex must be directed out. Since every two consecutive edges, in this instance, belong to a path, this two consecutive edges induce a valley (in this particular path).

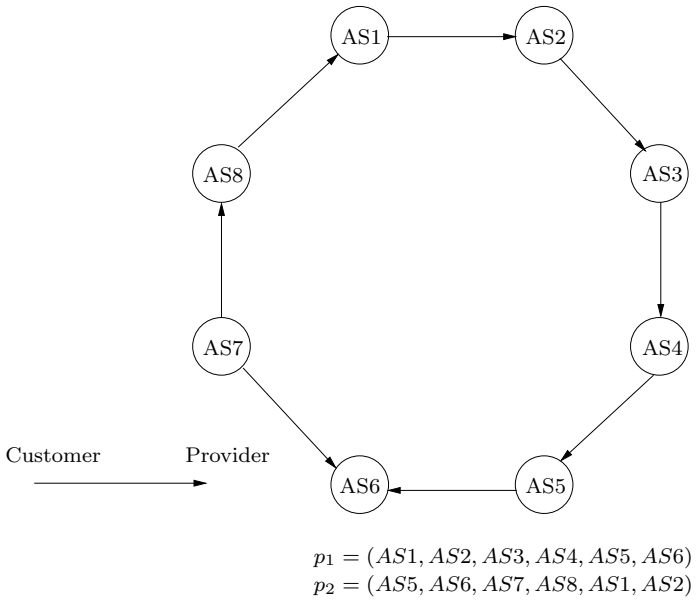


Figure 1.3: A *ToR* instace - every solution contains cycle

executed by an AS does not follow the export policy presented above. Thus, due to the locality of the policy, paths that traverse such ASes may contain valleys. We show that similar to the original problem (in which the objective function is to maximize the number of valid paths), the decision version of this variant of the problem is NP-hard, and we present $\frac{2}{3}$-approximation algorithm for the maximum version of the problem. We consider practical aspects of inferring the actual type of relationships between ASes. This includes heuristics to infer also *peer-peer* and *sibling-sibling* relationships, and then we examine our algorithms over real up-to-date data gathered from the Route-Views database, and perform simulations over several random graph. We also compare our algorithms to other approaches presented in [11, 48, 24].

The combination of the hierarchical structure and the policy based routing used in the AS graph, imposes that connectivity in the AS graph does not necessarily mean reachability. Thus, routing in the AS level is different than router level routing. In particular, routing is not necessarily done along shortest paths and a concatenation of routing paths does not necessarily produce new legal routing path. As a result, in many cases routing paths in the Internet are inflated, and the actual length of routing paths between clients is longer than the minimum hop distance between them [27, 47].

For instance, consider the AS topology graph depicts in Figure 1.4. While the length of the physical shortest path between AS6 and AS4 is two (using the path AS6, AS7, AS4), this is not a valid routing path since it traverses a valley. In this case, the length of the shortest **valid** routing path is five (using the path AS6, AS5, AS1, AS2, AS3, AS4). In practice, using a real data gathered from 41 BGP routing tables, Gao and Wand [27] showed that about 20% of AS routing paths are longer than the shortest AS physical paths[3]

While routing policy is a fundamental and important feature of BGP, the resulted path inflation may increase the delay between network clients. In this case, the service provided

[3]Our experiments exhibit a similar result over an up-to-date data from May 2007 (see Section 4.3).

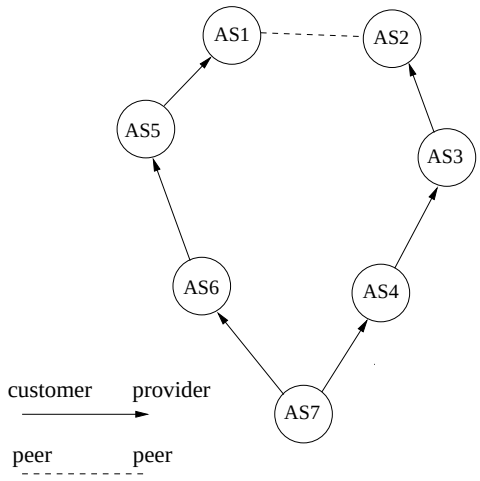


Figure 1.4: Comercial relationship policy

by network applications that are sensitive to delay may be harmed. Voice over IP (VoIP) for example, is a network technology that uses the Internet to carry voice signals. VoIP applications such as Skype (*http://www.skype.com*), Google Talk (*http://www.google.com/talk/*) and others, are becoming more and more popular offering IP telephone services for free. By its nature, the quality of VoIP calls is sensitive to network delay and a considerable amount of effort is done to reduce the delay between clients in order to achieve better quality. In particular, while a one way delay of 150 milliseconds is noticeable by most users but is most cases is acceptable, a one way delay over 400 milliseconds is unacceptable [32]. When reducing the delay is crucial, one should find ways to route packets over shorter paths and to overcome the path inflation phenomenon.

In Chapter 4 we consider an intermediate routing scheme, in which routing data between clients can be done via intermediate nodes. These intermediate nodes are used to overcome the path inflation problem in the Internet and eventually to reduce the delay between network clients. In particular, we enable clients to communicate via shortest paths despite the policy enforced by ASes. This method is based of the fact that due to the policy paradigm, a concatenation of valid routing paths does not necessarily induce a new valid path. In Figure 1.4, for example, (AS6, AS7) and (AS7, AS4, AS3) are two valid paths, but their concatenation (AS6, AS7, AS4, AS3) is **not** a valid path since it contains a valley[4]. Using the intermediate routing scheme one can deploy a relay server in a client located at AS7. Thus, AS6 can route a packet to AS7 using the valid path (AS6, AS7), and then AS7 can route the packet to AS3 using the valid path (AS7, AS4, AS3). In general, in the intermediate routing scheme, packets can be routed via one or more intermediate nodes and therefore the total routing length between clients can be reduced.

While intermediate routing may be supported in the IP layer using the IP source routing mechanism [5] or IP-IP encapsulation [42], these options are not practical since such packets are usually discarded by IP routers and IP filters (mostly for performance and security reasons). A more practical way to use intermediate routing is in the application layer. In

[4]On the other hand, one can see that according the policy guidelines presented above a sub-path of a valid path is valid as well.

this way, instead of sending a packet to the destination, the application sends the packet to an intermediate node. A special application, called *relay server* (or *intermediate server*), that was deployed in this node beforehand, forwards the packet by sending a new IP packet from the intermediate node to the destination or to another, closer, intermediate node. This process continues until the packet reaches its original destination. Clearly, if we have *relay servers* at each AS, one can easily obtain the shortest path between any pair of source and destination. Nevertheless, the deployment of *relay servers* is not cost-free and therefore a natural objective is to minimize the number of these servers and still to be able to route packets over shortest paths. The intermediate routing paradigm can be utilized, for example, by VoIP providers to improve the quality of their service; by Internet Content Provider (ICP) that have several contents servers, to reduce the delay between their clients and therefore to improve their customers' service; and by other applications.

We formally define the Intermediate Shortest Path Routing problem, and study the complexity of the problem. In particular, we show that the decision version is NP-hard and we present a $O(\log(n))$ approximation algorithm to the problem. This algorithm is the best one can get, and we prove that the problem cannot be approximated within a factor of $(1-\epsilon)\cdot\ln(n)$ for any $\epsilon > 0$ unless $NP \subseteq DTIME(n^{\log\log n})$. Then we examine our algorithm over a real up-to-date data that is gathered from the Route-Views database [4], considering specifically two practical scenarios.

This thesis sheds light on several open questions related to the topological and hierarchical structure of the Internet, mostly connected to the practical routing policy used by BGP. Our insights include estimation of the actual size of the Internet, the type of these missing links and their structure, and a better understanding on the acyclic hierarchical structure of the Internet. We show how this understanding can be used to develop algorithm and technique to utilize the Internet in more efficient way, and we hope that our results will be used to improve and develop new protocols and algorithms. We summarize our work and present some conclusions in Chapter 5.

Chapter 2

The Internet Dark Matter – on the Missing Links in the AS Connectivity Map[1]

In this chapter we suggest a novel usage of the AS level collected data in order to infer information regarding the AS connectivity graph. In addition to looking at the overall graph that is generated from the union of the data obtained by performing many measurements, we consider the actual different measurements and the amount of new data obtained in each of them with respect to the previous collected data. This allows us to reach conclusions regarding the structure of the system we are measuring, and ar to estimate its total size. We present strong evidence to the fact that a considerable amount (at least 35%) of the links in the AS level are still to be unveiled. Our findings indicate that almost all these missing links are of type *peer-peer*, and we provide novel insight regarding the structure of the AS connectivity map with respect to the peering type.

2.1 Understanding the AS Data Gathering Process

In order to formalize the methods used to gather information about the available peering relations in the Internet we use several simplifications and assumptions. This helps us create a rigorous view of the discovery process, and hopefully maintains the most important and relevant aspects of the discovery process while eliminating less important issues.

There are several methods to gather and sample information of the AS connectivity map. The first one is based on BGP routing tables, where each routing table contains the AS paths between a source to each of the relevant subnetworks in the Internet. Since most AS level routing do not distinguish between different networks within the same AS, we assume, for simplicity, that each routing table contains the set of AS path from the source to all other ASes. Thus, the collection of all the path vectors from a given AS to all other ASes is a tree.

As discussed in Chapter 1, BGP is a policy based routing protocol and in practice valid routing paths do not include *valleys* nor *steps*. Although ASes may use other policies and BGP routing table may reflect more than one route for a destination AS, we assume that

[1]The results described in this chapter appeared also in [16].

under the above policy, routing is done along shortest paths. This assumption makes the discussion regarding the retrieval process formal and rigorous. Thus, one can now model the process of retrieving peering information by creating a policy based shortest path tree, namely, a set of shortest paths from a given node to all the nodes in the graph that follows the policy guidelines presented above.

The question of discovering peering relations translates now to the amount of edge covered by a union of such trees rooted at a given set of nodes. In other words, the amount of peering relations covered by a collection of BGP path vectors from a set of ASes, corresponds to the amount of edges covered by a set of the corresponding trees. In this work we use the Route-Views project [4] as a source for our BGP database. This project collects a snapshot of the Internet AS level topology on a daily basis from over 40 ASes.

The second methods to gather information is the Internet Routing Registry [1]. This is a union of world-wide routing policy databases that use the Routing Policy Specification Language (RPSL) [6, 39]. These databases contain, among other things, the local connectivity information for the registered ASes. In terms of the AS connectivity graph, this corresponds to discovering all the edges connected to a given node and therefore these objects are referred to as stars. Obviously, if we had such a complete and an updated database we could easily derive the AS connectivity map. However, not all the ASes are willing to publish their peering relationship. Moreover, in some cases the entries in the database are out of date, thus they may fail to contain existing peers while in some other cases they may contain peering relationship that are no longer valid.

The most updated and complete IRR database is maintained by RIPE [3]. This is one of four Regional Internet Registries and during June 2005 it consisted of about 6800 ASes that have registered in Europe[2]. In this database almost all the registered ASes (over 98%) share their peering relationship. In contrast, only 400 ASes out of 11000 ASes that have registered in ARIN (American Registry for Internet Numbers) share this information.

As mentioned above, entries in the IRR may be invalid. Thus, one should use some filters mechanism that remove these entries. In this work we use a filter that is based on the sanity checks that have been presented in [14]. These filters are based on the fact that a valid peer should appear in the entries of both ASes while a peer that appear in only one entry may be out of date and should be removed from the IRR. In case that one of the ASes does not have an entry in the IRR it is not clear whether this peer should be filtered or not. Thus, we consider two filtered database, the first one contains this peer while in the second these peer are removed.

DIMES [45] is a new project that samples the Internet using *traceroute*. In particular, distributed agents located in thousands hosts around the world, perform periodic *traceroute* to a set of IP addresses. In contrast to the other methods (i.e. BGP routing tables and IRR) this technique obtains information regarding the Internet connectivity in the router level. Nevertheless, one can correlate between an IP address and its corresponding AS (i.e. the AS that allocated this IP address) and therefore connectivity in the router level induces connectivity in the AS level. As of June 2005, DIMES consists of 3781 distributed agents[3]

[2]Actually there are almost 10000 registered ASes but over 3000 out of them seem to be inactive due to the fact that they do not appear in the Route-Views database and their entries look invalid (see discussion regarding IRR filtering mechanism in the next paragraph).

[3]Note that several agents can be located in a single AS.

located in 77 countries around the world and it spans 39000 links from the AS map.

2.1.1 Covering graph by shortest path trees

In [14] the authors found that the available databases consisting of BGP routing data alone are not complete enough, and when adding the IRR database a significant number of links are revealed. They explain this result by the fact that there are several paths to each AS while only one path is published, and by the private nature of *peer-peer* links. In this section we reconstruct the gathering process that is used to build the Route-Views database, using information from June 2005. We show that most of the links in this database are already disclosed by less than 10 BGP routing tables (from different sources) while the amount of peers that are discovered by the rest of the sources is very small. Simulating this process over several graph models we find that policy has a significant role in the revealing process and that most of the hidden peers are of type *peer-peer*, while almost all the *customer-provider* peers are unveiled.

During June 2005 the Route-Views database consisted of about 20000 ASes and 43200 links and it was a superposition of 40 different BGP routing tables[4]. With respect to the discussion above, the process of composing these BGP routing tables into a complete database is similar to the process of spanning a graph by a set of policy-based shortest path trees. In order to examine the process in which these routing tables are composed, we took from the Route-Views database the 40 BGP routing tables (each representing a policy-based tree). Then, using 50 random permutations, we calculated the average number of new links found in each step. Figure 2.1 depicts the average number of new peers that are unveiled by the i'th BGP tree. Clearly, the first routing table discover over 20000 links[5] (the average size of a BGP routing table). However, the amount of new discovered links decreases very fast. In fact, starting from the tenth routing table, each new table unveiled less than 300 new peers (compare to more than 1000 peers in the first 5 trees), a very small number comparing to the size of the graph. In other word, a small number of BGP trees revealed a significant amount of peers while the rest of the routing tables reveal very little. Although many links remain hidden, incrementing the number of BGP routing tables will not help much in increasing the number of unveiled links.

As mentioned before, by disregarding the policy, a BGP routing table can be approximately simulated by a shortest path tree. Thus, we can simulate this covering process by generating graphs and cover their edges by a set of shortest path trees. In Figure 2.2 we show this covering process for several random graphs[6]. One can see that similar to the Route-Views covering process, the contribution of new trees by means of new edges decreases very fast. Obviously, the degression is more moderate in graphs that have more links due to the

[4]Actually, there are 45 routing tables but 5 of them contains less than 1500 links compare to more than 20000 links in the others. Since their total contribution is less than 200 links, we ignore these small tables to avoid unnecessary deviations.

[5]In practice, a complete BGP routing table may contain several paths to each AS. Thus, each table contains more links compared to a tree.

[6]In this experience, and throughout this paper we consider the following random graphs. $G_N(p)$ graphs in which an edge between two nodes exists with probability p, a Barabasi-Albert graph [10, 8] and Waxman graph [49], all have about 18000 nodes. For each model we generated two graphs with 40000 (termed a a small graph), and one with 80000 edges (termed a large graph).

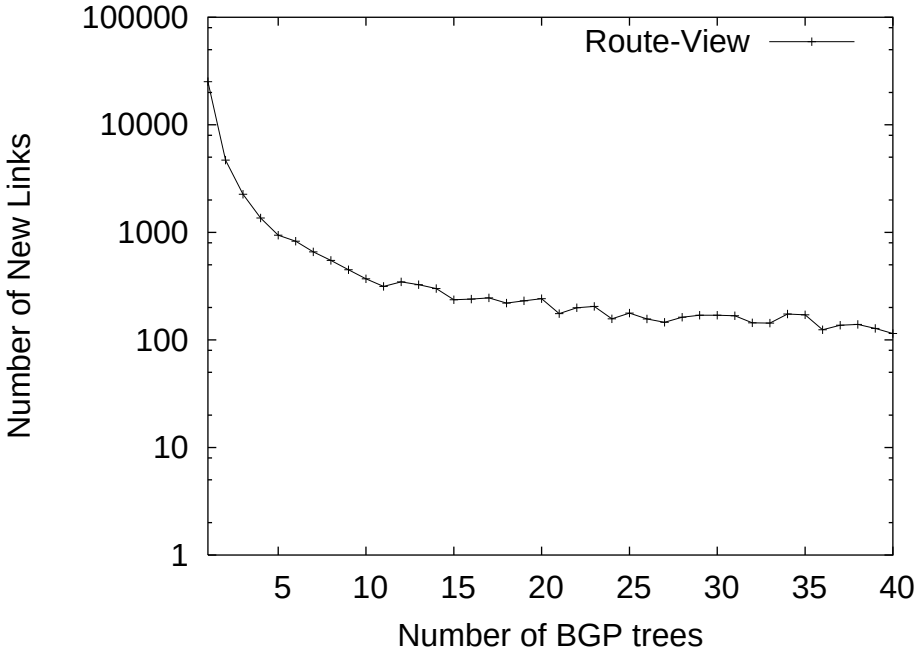


Figure 2.1: Route View Covering Process

fact that each tree contains exactly $N-1$ edges, thus exposing large graph requires more trees.

There are two main differences between the covering process of the AS database and our simulated graphs. The first one refers to the quantity of the cover. From the fact that the IRR database contains thousands of links that do not appear in the route-Views database we know that the route-Views database covers at most 60% of the links in the AS connectivity map. On the other hand, in our simulation the set of 40 trees cover almost all the edges in the graph (see Figure 2.3). The second difference can be observed by considering figures 2.1 and 2.2. While the degression of the number of new link becomes moderated in the AS graph (see Figure 2.1), it remains linear (in the log scale) in our simulation (see Figure 2.2).

In order to understand these differences we want to study the impact of the policy routing on the covering process. According to our policy paradigm, as described above, two ASes from different hierarchy level are connected by *customer-provider* link while two ASes from the same hierarchy level are connected by *peer-peep* link. In order to simulate this structure we need to classify the nodes in each one of our graphs into several hierarchy levels and give orientation to the links according to this hierarchy.

Using the guidelines from [29, 24, 48] we divide the set of nodes V of each graph into four hierarchy groups according to the vertex degree where the number of ASes in each group increases exponentially. Thus, the set of nodes of each graph is divided in the following way: 15 ASes (that have the highest degree) are in level 1, 150 ASes are in level 2, 1500 ASes are in level 3, and about 16500 ASes (that have the lowest degree) are in level 4. According to this hierarchy, ASes from the same groups are connected by a *peer-peer* relationship while ASes from different groups are connected by a *customer-provider* relationship. Note that while

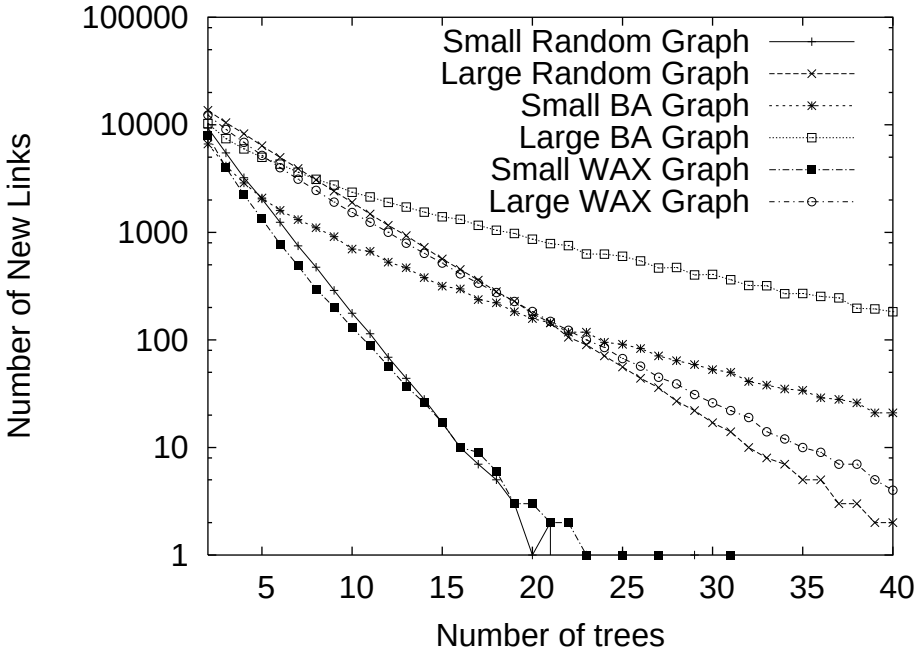


Figure 2.2: Graph Simulation Covering Process

this approach (degree based hierarchy) may not be the best to approximate the hierarchy structure of the AS connectivity map, it provides "good enough" method to simulate the policy-based discovery process.

In general, when routing policy is considered, connectivity does not necessarily mean reachability, namely if two ASes are physically connected via one or more physical paths it does not necessarily means that there is a valid routing path with respect to the adopted policy. In heavy-tailed models such as Barabasi-Albert, there is a strict correlation between the hierarchy structure of the graph and the policy. Thus, In these kind of models more edges are of type *customer-provider* and the reachability under the policy routing constraints is stronger compare to other model. In particular, in the Barabasi-Albert graphs that have been used in our simulation, more than 54% of the edges are of type *customer-provider* while in the random graphs and the Waxman graphs only 30% and 35% of the edges are of type *customer-provider* respectively. Moreover, In the Barabasi-Albert graphs a single tree unveiled couple of thousands vertices, in the Waxman graphs a single tree unveiled couple of hundreds vertices, while in the Random graphs only a few dozens of vertices are unveiled by a single tree. One can observe that there is a correlation between the number of *customer-provider* links in a graph, its structure, and its reachability.

The intuition behind this property is that a valid routing path cannot contain more than one *peer-peer* link [24, 48]. Moreover, *customer-provider* links must precede *customer-provider* links. Thus, in a heavy-tailed graph, where many vertices are connected to a heavy core by *customer-provider* links, almost all the ASes are connected. In other models, every vertex can reach only its local environment since the existence of many *peer-peer* links forbid

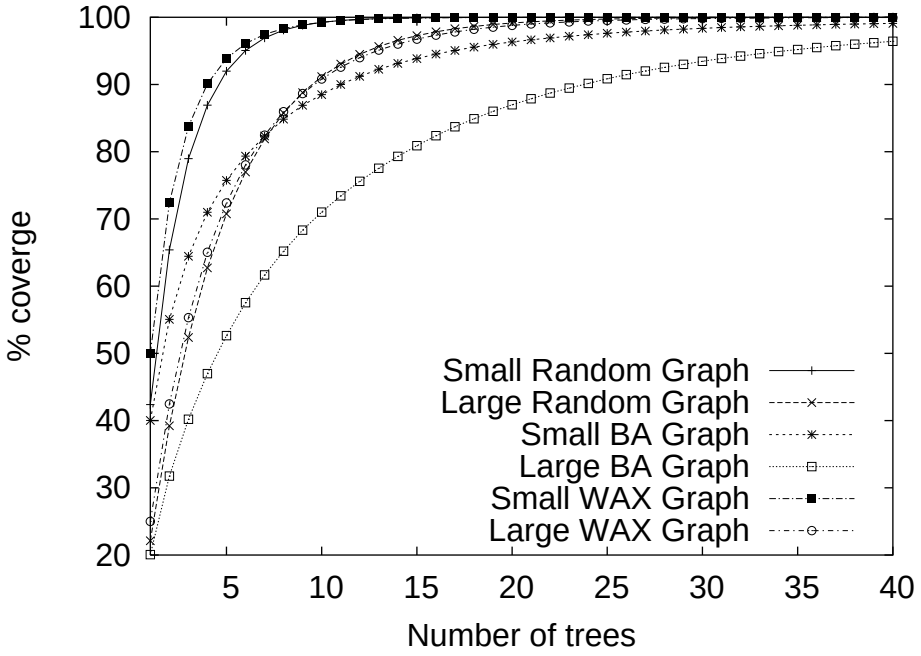


Figure 2.3: Percentage of Covering

many (long) paths. Using the fact that the AS graph is strongly connected[7] similarly to the Barabasi-Albert model, we simulate the policy-based covering process over this model alone. In Figure 2.4, one can see that similar to previous simulation, a small number of trees reveals most of the link and similar to the AS graph, the degression of number of new link become moderated. In addition, significant amount of peers (more than 45%) remain hidden (see Figure 2.5).

Another interesting point is the difference between the covering process of *peer-peer* links and *customer-provider* links. In [24, 48] the authors presented some heuristics to infer the type of relationship of the peering in the AS graph. In both works they found that in the Route-Views database, less than 8% of the links are of type *peer-peer*. Using the algorithm presented in [24] we have found that 39700 out of 43200 links in the current Route-Views database are of type *customer-provider* while 3650 are of type *peer-peer*. In contrast, in [46] the authors show that in the IRR database more than 56% of the links are of type *peer-peer*. They have tried to explain this fundamental difference by the fact that one of the algorithms may mislead or by the fact that entries in the IRR are incorrect. We suggest a different explanation that is based on the fundamental difference between BGP based database and IRR database. As explained before, due to the locality of *peer-peer* links, it is hard to unveiled this kind of links by BGP routing tables, thus this overwhelming majority of *customer-provider* links in the Route-Views database is not surprising and one should not infer that this is the ratio between the number of *peer-peer* and *customer-provider* links in the full AS graph. In contrast, the IRR database is not affected by the policy and therefore it unveils more *peer-peer* links and reflects the ratio between *peer-peer* and *customer-provider* links

[7] In the Route-Views database every single tree unveiled almost all the ASes.

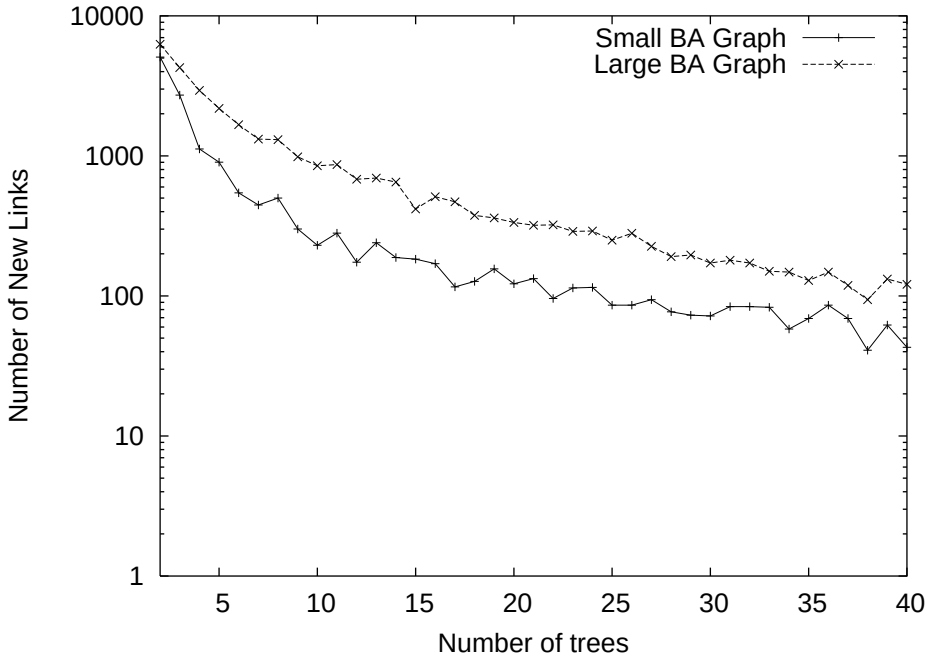


Figure 2.4: Policy Based Covering Process

more accurately. Our simulations support this explanation and show that in the covering process (that simulates the BGP database) less than 3% of the links that have been unveiled are of type *peer-peer* compare to almost 45% in the full graph. Namely, a graph may contain many *peer-peer* links that will be hidden in a subgraph composed of BGP routing tables. In addition, in a subgraph that consist of a set of stars (simulating the IRR database) there are 44% *peer-peer* links, similar to the original full graph.

Recall that when we ignored policy, the covering process unveiled almost all the edges in the graph (including the *peer-peer* links). We also saw that *peer-peer* links are almost not revealed when policy is used. An interesting question is, if we consider the *customer-provider* subgraph alone (i.e the subgraph that consist of *customer-provider* links), what is the quality of the cover? namely, Does the covering process cover most of these links, or due to policy consideration many links remain hidden?

Figure 2.6 depicts the coverage of both the *peer-peer* and the *customer-provider* subgraphs. In the large graph (i.e. the graph that has about 80000 links) 48000 links are of type *customer-provider* and more than 90% of these links were unveiled in the covering process (recall that 97% of all the links were unveiled when the policy is ignored). In the small graph (i.e. the graph that has about 40000 links) in which 25758 link are of type *customer-provider*, 84% of these links were covered. This coverage is smaller than the one in the large graph since there are less links in this graph and reachability is more difficult. Nevertheless, in both graphs the coverage of the *customer-provider* subgraph is very large which implies that the AS connectivity graph has a similar behavior. Thus, while the covering process of the Route-Views database is not complete and a significant amount of links are remain hidden, a large portion (80%-90%) of the *customer-provider* peers are unveiled

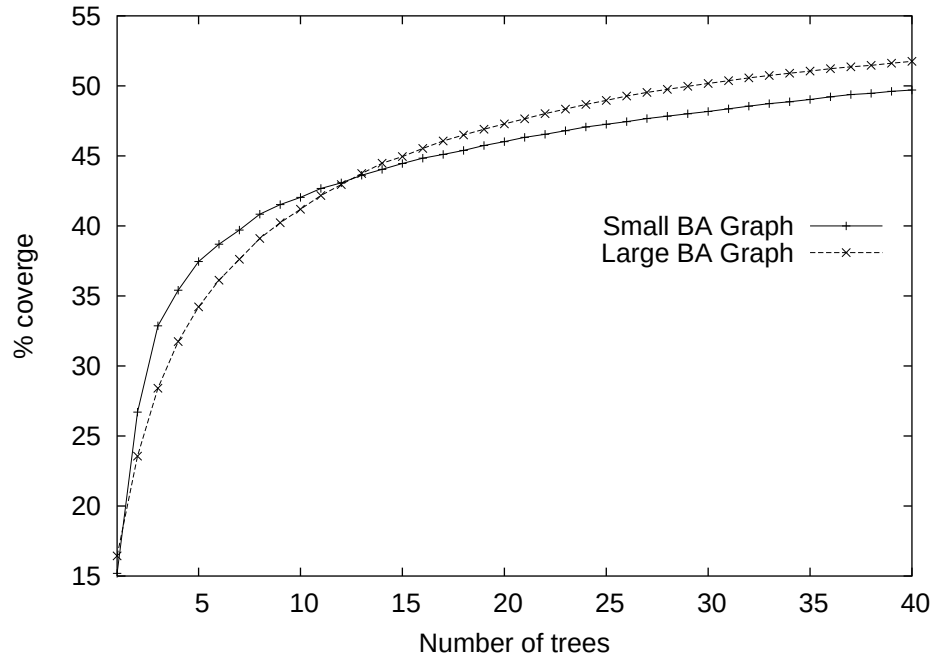


Figure 2.5: Policy Based Percentage of Covering

by this database. Namely, the Route-Views database span almost all the *customer-provider* subgraph while the *peer-peer* subgraph remains hidden.

2.1.2 IRR Policy Analysis

In this section we use the concept that was presented in [46] and analyze the policies in the IRR database using a variant of this method[8]. Using this analysis we infer the business relationship of the registered ASes. In addition, we consider two practical scenarios in which links may be classified in more than one way.

Routing Policy Specification Language (RPSL) allows network operator to specify routing policies at various levels in the Internet hierarchy. In particular, RPSL is used in IRR to describe BGP routing policy at the Autonomous System level. RPSL is an object oriented description language that contains many classes and attributes. The most important class from our point of view is the *aut-num* class that contains *import* and *export* attributes that describe the routing policy of an AS. Parsing the RIPE IRR we have obtained the export policy of about 8,200 registered ASes. To infer the business relationship between connected ASes (i.e. the type of peers) we use the guidelines presented in [7] and [31] in which an AS can export its routes and its customer routes to its providers and peers, but usually does not export its provider or peers routes. In contrast, AS can export its routes and its customer routes, as well as its provider or peer routes to its customers and sibling. Consider

[8]In particular, we do not rely on the import policy of an AS, since in many cases this policy does not reflet the actual export policy of the neighbor AS. Thus, in contrast to [46] we infer the business relationship of a link only if the export policy of both edges are available.

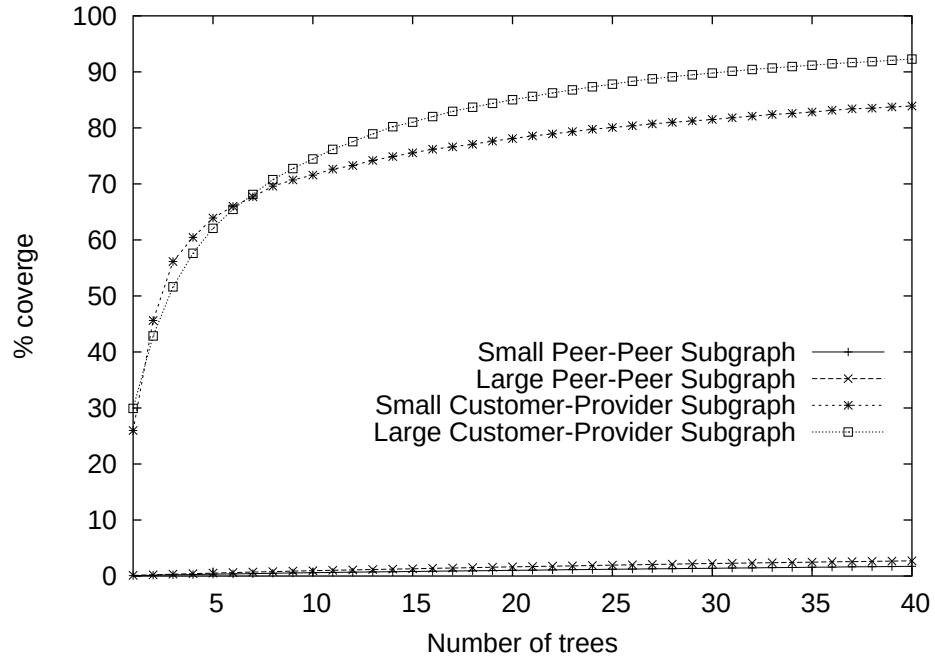
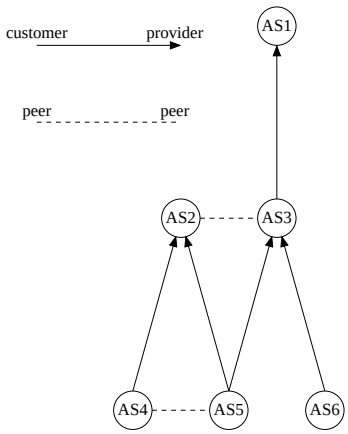


Figure 2.6: Subgraph Covering Process

a simple topology of six ASes described in Figure 2.7. The export policy of each AS is also described in RPSL format using *aut-num* class and *export* attribute[9]. For example, AS5 does not export AS4 and AS2 to AS3. Thus, according to the guidelines described above, the peer *AS5-AS3* cannot be a *customer-provider* link (where AS5 is the provider), neither *sibling-sibling* link. In order to determine the type of this link we should explore the export policy of the other edge (i.e. AS3). Since AS3 export all its neighbor to AS5 we now can infer that *AS5-AS3* is a *customer-provider* (where AS3 is the provider). One can observe that in order to determine the type of a single peer, the export policy of both edges are required. Moreover, the import policy of an AS does not reflect the export policy of the other edge. Thus, only peers in which both edges describe their export routing policy in the IRR can be analyzed. As described above our first IRR filter mechanism that is used to remove invalid peers meet this requirement and therefore it is used in our analysis. After analyzing the IRR database using this method, we have found that 26700 out of 36237 links in the IRR database are of type *peer-peer*, 8990 links are of type *customer-provider* and small amount of 490 links are of type *sibling-sibling*.

Analyzing peering type of relationship as described above may lead to several problems in which the type of links may be interpreted in different ways. The first problem refers to the case in which an AS has no more than one provider (e.g. if an AS is a stub or if an AS is in top hierarchy). In this case (and according to our export guidelines) an AS exports all its routes to its neighbors, regardless the type of relationship. Thus, the export policy

[9]In practice, the *aut-num* class contains more attribute such as *import* to describe the import policy and other administrative attributes for maintenance. In addition, the *export* may be more complex and contain regular expressions, routes, and aggregation of routes and ASes (using *as-set* and *route-set* RPSL classes).



aut-num: AS3
export: to AS1 announce AS3 AS5 AS6
export: to AS2 announce AS3 AS5 AS6
export: to AS5 announce AS1 AS2 AS3 AS6
export: to AS6 announce AS1 AS2 AS3 AS5

aut-num: AS5
export: to AS2 announce AS5
export: to AS3 announce AS5
export: to AS4 announce AS5

aut-num: AS4
export: to AS2 announce AS4
export: to AS5 announce AS4

aut-num: AS6
export: to AS3 announce AS6

aut-num: AS1
export: to AS3 announce AS6

aut-num: AS2
export: to AS3 announce AS2 AS4 AS5
export: to AS4 announce AS2 AS3 AS5
export: to AS5 announce AS2 AS3 AS4

Figure 2.7: Routing Policy Example

in these cases is not enough to determine the type of relationship of a link. For instance, using the same export policy described in Figure 2.7, the link *AS1-AS3* can be identified as a *peer-peer* instead of a *customer-provider* and the link *AS3-AS6* can be identified as a *sibling-sibling* instead of a *customer-provider*. In the RIPE database only 950 links (out of 36237) are in this category and may be interpreted in more than one way. In particular, 300 *sibling-sibling* links can be classified as *customer-provider* links and 650 *customer-provider* links can be classified as *peer-peer*.

While this problem refers to the way in which the export policy is interpreted, the second problem questions the classic paradigm that consider three, well defined, types of relationship (i.e. *customer-provider*, *peer-peer*, *sibling-sibling*). In contrast to the theoretical guidelines in which an AS either exports all its providers (and peers) or none, in practice there may be case in which an AS export only a subset of its providers. For instance, consider that AS5 exports AS3 to AS4 but does not export AS2. According to our analysis, in this case the link *AS4-AS5* is interpreted as a *peer-peer* link (since AS4 does not export AS2 and AS5 does not export AS2). Nevertheless, this analysis may be wrong since although AS5 does not export all its providers it exports part of them. Thus, these link cannot be interpreted by the classic

paradigm presented above. In particular, one may infer that the type of the link *AS4-AS5* should be *customer-provider* (where AS5 is the provider) and not *peer-peer*. This kind of ambiguity is very common and 4800 peers in the IRR meet this category. These links may be interpreted as *customer-provider* or *sibling-sibling* instead of *peer-peer* or *customer-provider* respectively, since at least one edge exports subset of its providers.

One way to deal with this problem is to determine the type of each link by a threshold as follow. We say that a *AS-i* is *x-provider* of *AS-j* if exactly $100x\%$ of *AS-i* providers are exported to *AS-j*. According to this definition if *AS-j* is *0-provider* of *AS-i* and *AS-i* is *0-provider* of *AS-j*, thus the link *AS-j, AS-i* is of type *peer-peer*. If *AS-j* is *1-provider* of *AS-i* and *AS-i* is *1-provider* of *AS-j*, thus the link *AS-j, AS-i* is of type *sibling-sibling*. And finally, If *AS-j* is *1-provider* of *AS-i* and *AS-i* is *0-provider* of *AS-j*, thus the link *AS-j, AS-i* is of type *customer-provider* (where *AS-j* is the provider). For the rest of the cases (i.e. where $0 < x < 1$) we define a threshold t. If $x > t$ then x is considered to be 1. If $x \leq t$ then x is considered to be 0. Figure 2.8 depicts the number of *peer-peer* and *customer-provider* links in the IRR for different threshold values. Clearly, for small threshold values the amount of *customer-provider* links increases at the expense of *peer-peer* links (in absolute number, the total amount of *sibling-sibling* links almost does not change and thus it is not considered in our discussion). Another interesting inference from Figure 2.8 is that almost all the links that moved from *peer-peer* to *customer-provider* type have a big threshold value (i.e. $t = 0.9$). Namely in almost all the cases, a specific AS exports all its providers (to another AS) except one.

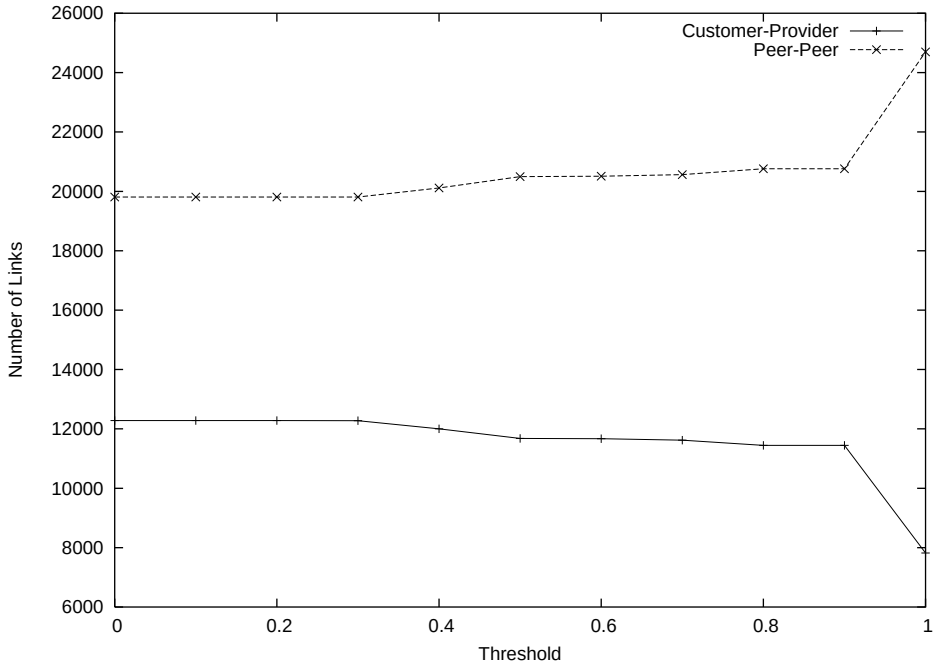


Figure 2.8: $x -$ *customer-provider* Threshold

Another way to deal with this problem is to consider new types of relationship. Recall, that the types of relationship intend to describe the business relationship between ASes. For

instance, in a *customer-provider* link the providers gives to the customer full access to its routes. Nevertheless, in some cases a provider gives only partial access to its provider (i.e. some of the routes are blocked), thus the provider is not a *full-provider* but a *semi-provider*.

2.2 Approximating the AS Connectivity Graph Size

Using data, collected during June 2005 from Route-Views, IRR, and DIMES projects we unveiled 83782 AS peers[10]. Nevertheless, as discussed earlier, this data is not complete enough and despite the increasing effort to reveal the full map, some peers may remain hidden. In this section, we address the following question: What is the overall size of the AS connectivity graph. We want to be able to answer this question without assuming anything about the full (partly unrevealed) graph. It is important to note that having a good approximation of the size of the AS connectivity map is not just a theoretical question. The overall number of active ASes is known (about 21000 during June 2005), and thus the overall number of edges translates directly to the average node degree - which is an important parameter regardless of the model we use. We show in this section that the AS connectivity map contains at least 128000 links and most likely the size of the graph is higher.

First, we try to estimate the size of the connectivity graph using the degree of the stars in the IRR database assuming that the database is representative. This database consist of 6583 stars and 72474 links, using the first filtering or 7129 stars and 82339 links, using the second filtering. Thus, the average degree is $\frac{72474}{6583} = 11$ and $\frac{82339}{7129} = 11.54$ respectively. Using the fact that the AS graph consists of 21000 active ASes this implies that the AS graph consists of $21000 \cdot \frac{11}{2} = 115000$ or $21000 \cdot \frac{11.54}{2} = 121000$ peers. Clearly, the estimation using this method depends on the filtering used. An aggressive filtering, may remove legal peers and therefore it induces a smaller IRR graph compare to the real one. Thus, the estimation in this case deviates down. On the other hand, using a moderate filtering the induced graph may contain peers that are no longer valid. In this case the estimation deviates up. To emphasize this issue let us estimate the size of AS connectivity graph using the unfiltered IRR data. In this case the database contains 9247 stars and 138343 reflecting an average degree of 14.9 and a size of 156000 edges for the full graph.

With respect to the assumption that the database is representative, one may refer to the fact that almost all the ASes in the IRR are located in Europe and it is possible that the average degree of ASes in Europe differs from the average degree of an ASes in other region. In order to avoid such assumption, we aim at estimating the number of AS peering relations using the data added by the IRR database to the Route-Views database.

In Section 2.1.2 we analyzed the type of relationship in the IRR database and showed that 8990 out of 36237 peers (i.e. 25%) in the filtered IRR[11] are of type *customer-provider*. If we consider the last discussion in Section 2.1.2 regarding the way in which export policy is analyzed, there may be a doubt regarding 4200 *peer-peer* links. In this case, when we consider only the undoubt links, $\frac{8990}{8990+22056} = 29\%$ of the link are of type *customer-provider*. Assuming that the sampling space is representative, 25-29% of the peers in the full AS connectivity

[10] This is after filtering the IRR data as discussed in Section 3.5.

[11] Recall that this analysis requires that both edges will be in the database. Thus, we have used only the second filter mechanism.

map are of type *customer-provider* while 71-75% are of type *peer-peer* (or *sibling-sibling*). Since the Route-views database contains 39700 *customer-provider* links it indicates that the full AS connectivity map contains at least $\frac{100}{29} * 39700 = 138000$ links. Using a more conservative analysis, considering all the doubt links (i.e. link that their type is unknown with respect to the last discussion in Section 2.1.2) as a *customer-provider* links, 37% of the links in the IRR are of type *customer-provider*. Therefore, the same technique indicates that the AS connectivity map contains at least 107300 links. Nevertheless, in this case $0.36 * 36237 = 13045$ of the peers in the IRR are of type *customer-provider* while the IRR and the Route-Views data have only 6730 common *customer-provider* links. Thus, the size of the *customer-provider* subgraph is at least $39700 + 13045 - 6730 = 46015$ links. In this case the lower bound is 128000 links[12].

Obviously, these estimations are very conservative and intent to give a lower bound on the size of the AS map. For instance, in the last estimation we assumed that the Route-Views data does not discover about 6500 *customer-provider* links in Europe alone. This indicates that (under the assumption that 36% of the links are of type *customer-provider*) a large portion of 20000 *customer-provider* links remain hidden and therefore the estimation should be 165800 links.

Next estimation is based on the intersection of the new sample space (the stars coming from the IRR) with the existing coverage created by trees from the BGP routing information. If the new data contains a set of independent edges, we could measure the portion of the full graph covered by the BGP data, because it gives us the probability that an edge is covered by the BGP data. Recall that the Route-Views database contains 43200 links and the unfiltered IRR contains 102106 links. The union of these two graph contains 128697 links. This means that 16618 out of the 102106 edges where already covered by the Route-Views database. Therefore, the probability of an edge to be discovered by the Route-Views database is $\frac{16618}{102106} = 0.16$ and the total number of edges can be approximated by $1/0.16 \times 43200 = 265400$. Obviously, removing invalid peers from the IRR database, increase the probability of an edge to be discovered by the Route-Views database. Thus, similar to pervious method (using the average degree of a node), the estimation is significantly affected by the filter mechanism. Using the filtered IRR, this probability is increased to $\frac{12107}{46611} = 0.26$ (where 12107 is the number of peers exists in the IRR and have already covered by the Route-Views database, and 46611 is the total number of peers in the filtered IRR database) using the first filter mechanism and $\frac{8719}{36237} = 0.24$ using the second filter mechanism. Thus, the total number of edges can be approximated by $1/0.26 \times 43200 = 166000$ and $1/0.24 \times 43200 = 180000$. Note that the accuracy of this method depends of the independency of the database. Thus, if there is a correlation between the data the estimation will deviate and indicate a smaller value (and vice versa).

In order to examine the method we simulated the process over several graphs. The first graph is a Barabasi-Albert graph and the second one is a superposition of a Barabasi-Albert graph (that contains 40000 edges) and a Waxman graph (that contain 100000 edges). Both graphs contain 18000 nodes. From each graph we have constructed two subgraphs. The

[12] In [46] the authors indicate that 42% of the links in the IRR are of type *customer-provider*, thus estimating the lower bound using their results may be different. However, since information regarding the common *customer-provider* links is also required, we cannot present this estimation.

Graph	Size Estimation	Graph Size
BA	161700	161955
BA+Wax	141085	141135

Table 2.1: Graph Size Estimation

Estimation Method	400 Biggest Stars	1000 Small Stars
Average Degree (First Method)	718000	58900
Data Intersection (Second Method)	195000	121800

Table 2.2: Robustness of Estimation

first one consisted of 5000 random stars (that simulates the IRR database) and the second consisted of 40 policy based shortest path trees (that simulates the Route-Views database). Table 2.1 summarizes the average results of the estimation process over 20 independent iteration, with respect to the actual size[13].

In contrast to the vertex degree method, where the estimation is significantly affected by the average degree of the sampling space, this method seems to be much more robust with respect to the degree of the nodes in the sampling space. However, our sampling space is indeed dependable and the estimation is still affected by the average degree. Every AS has at least one peer in the Route-views (Recall that the Route-Views consists of a set of trees that span the graph), thus the probability of an edge in an AS with small degree to be covered by the Route-Views is bigger than an AS with bigger degree. To demonstrate it, we divided the IRR into two subgraphs. The first subgraph contains the 400 biggest stars (i.e. the stars with the highest degree) and the second contains 400 small stars[14]. Table 2.2 summarizes the estimation using these two subgraphs (instead of using the full IRR).

Next, we use the same technique to estimate the size of the AS map using data from the DIMES project. Thus, we measure the intersection between DIMES and IRR data and between DIMES and Route-views data. During June 2005, DIMES unveiled 38928 links. 7000 links from them have already been revealed by the IRR data, while 23850 from them have been revealed by the Route-Views data. Namely, the probability of an edge in the DIMES data to be discovered by the IRR data or by the Route-Views database is $\frac{23850}{43200} = 0.55$ and $\frac{7000}{36237} = 0.19$ respectively. Thus, the total number of edges can be approximated by $1/0.55 \times 38928 = 201606$ using IRR and $1/0.19 \times 38928 = 70522$ using Route-Views. Clearly, the second estimation (using DIMES and Route-Views data) seems to be very low. However, since DIMES is based on *traceroute* queries, it obtains only links that traverse valid routing AS paths. Thus, it has strong overlapping with BGP based database such as Route-Views and the probability of an edge covered by DIMES to be unveiled by Route-Views is bigger compare to two independent subgraphs.

Currently DIMES consists of almost 4000 distributed agent performing *traceroute* to a set of random IP addresses. While IRR contains peering information of ASes located in Europe, less than 25% of DIMES agents are located in Europe. Moreover, the majority of

[13]Recall that in the simulations, unlike in the AS case, we know the actual size of the full graphs.

[14]The IRR contains many ASes with one peer (i.e. their degree is one). These links are found by the Route-Views as well. Thus is order to avoid this side effect we did not took the ASes with the smallest degree but ASes that have at least 5 peering relationship.

IP addresses are located outside Europe. Thus, most of the *traceroute* performed by DIMES agents are probably targeted to destination that are not covered by IRR data and their source is outside of IRR scope as well. Therefore, the correlation between IRR and DIMES data is weaker than to independent random subgraph and the estimation based on these two data set deviates up.

Using estimations presented so far, one can bound the size of the AS connectivity map between 128000 and 200000 links. Both methods used in this section are sensitive to many parameters (e.g. the average degree, the accuracy of filtering and the type of relationship analysis, independency of the database, etc.), thus trying to approximate the accurate size of the AS connectivity map is very difficult. Nevertheless, while the lower bound seems to be too conservative, the upper bound is too loose. Therefore, the actual size of the AS connectivity map is somewhere between these boundaries.

2.3 Vertex Degree Distribution

In their paper, Faloutsos et al. [22] showed that despite the apparent randomness of the Internet, a simple power-laws holds for the Internet in the AS level. This novel observation was adopted by many researches and it is one of the basic building blocks for modelling the AS connectivity map. The authors use the NLANR - National Laboratory for Applied Network Research data [2] consisting of several BGP routing tables. This kind of database (i.e. a database that consisting on routing tables alone) is incomplete and may cause a significant inaccuracies. In other words, the graph that is derived from this kind of database is only a subgraph of the full AS graph, thus properties that hold in the subgraph may not be valid for the full graph. In [14] the authors questioned this observation and showed using a more complete database (yet not fully complete) that the vertex degree distribution deviates from the straight line (reflecting a power-law distribution). One can see this deviation in Figure 2.9 that depicts the complementary distribution function of the AS degree as it is derived from the route-Views database alone, and from the route-Views plus IRR data. The data for this graph has been collected during June 2005. In this section we study the vertex degree distribution of the AS graph. We show that despite the fact that the databases are not complete, their vertex degree distribution may reflect the distribution of the full graph. In particular, we observe that the *customer-provider* subgraph follows the power-law while the *peer-peer* subgraph may behave differently. Although the most complete database contains both IRR, Route-Views and DIMES data, in order to understand the vertex degree distribution of the AS graph, we use the IRR database alone since it is not affected by the policy and therefore it is much more representative in the sense that the ratio between *peer-peer* and *customer-provider* link is more accurate.

As discussed in Section 3.5, BGP based database contains mostly *customer-provider* links and therefore it may be considered as a subgraph of the *customer-provider* graph. Moreover, using more representative database (i.e the IRR database) we showed that the ratio between *peer-peer* and *customer-provider* links is completely different and there are many more *peer-peer* links in the full AS graph. Therefore, although the vertex degree distribution of databases such as Route-Views and NLANR follows the power-law, it reflects the distribution of the *customer-provider* subgraph alone and does not give information

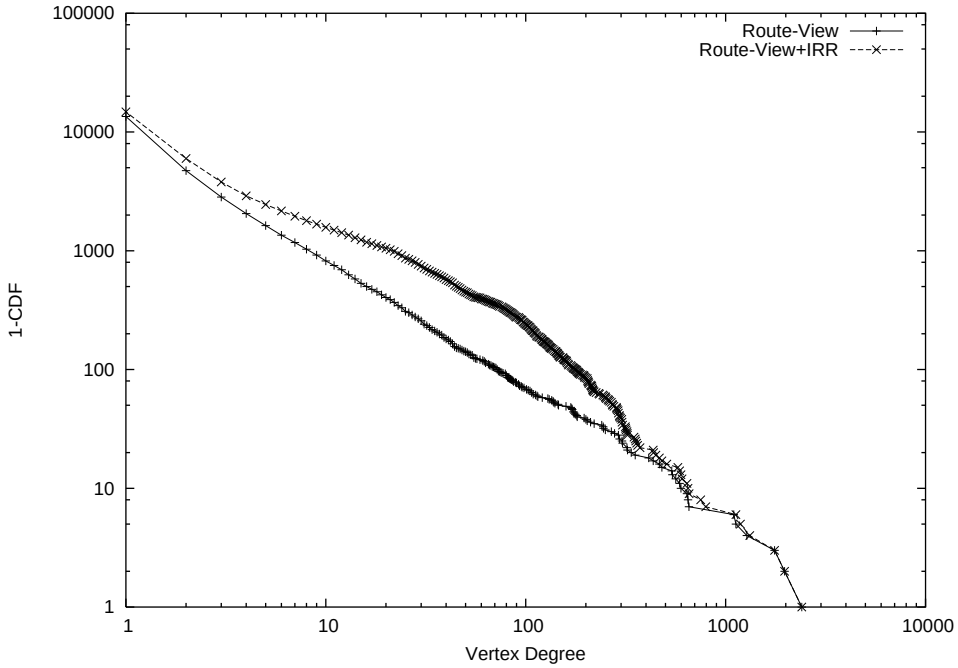


Figure 2.9: AS Vertex Degree Distribution

regarding the distribution of the *peer-peer* subgraph.

To support this finding, we use an independent *customer-provider* subgraph that is based on IRR data. We use the analysis described in Section 2.1.2 to divide the IRR data into *peer-peer* and *customer-provider* subgraphs. With respect to the last discussion in 2.1.2, we consider only the links with undoubt type. Thus, we have 8990 *customer-provider* and 22050 *peer-peer* links. Figure 2.10 depicts the vertex degree distribution of the *customer-provider* subgraphs as they derived from the Route-Views and the IRR database. The size of the *customer-provider* subgraph derived from the IRR data is much smaller than the Route-Views subgraph (in particular the first contains 8990 links while the second contain 43200 links), thus it is located below Route-Views subgraph. Clearly, both graphs follow the power-law.

In contrast to the *customer-provider* subgraph we suggest that the *peer-peer* subgraph does not follow the power law. Figure 2.11 depicts the vertex degree distribution of the *peer-peer* subgraph derived from the IRR. The distribution is completely different. In particular, it is much more similar to the distribution of Waxman graphs (see Figure 2.12).

Naturally, the distribution of the full graph is a superposition of both distributions (i.e. *customer-provider* and *peer-peer* subgraph). Figure 2.13 depicts the vertex degree distribution of the IRR. Recall that about 75% of the links in the IRR database are of type *peer-peer* and only 25% are of type *customer-provider*. Thus, the distribution of the full graph is mostly affected from the *peer-peer* subgraph.

So far we drew conclusions regarding the vertex degree distribution using only partial data. In particular, we used subgraph consisting of a collection of policy-based shortest path tree (i.e. the Route-Views database) and another subgraph consisting of a set of random

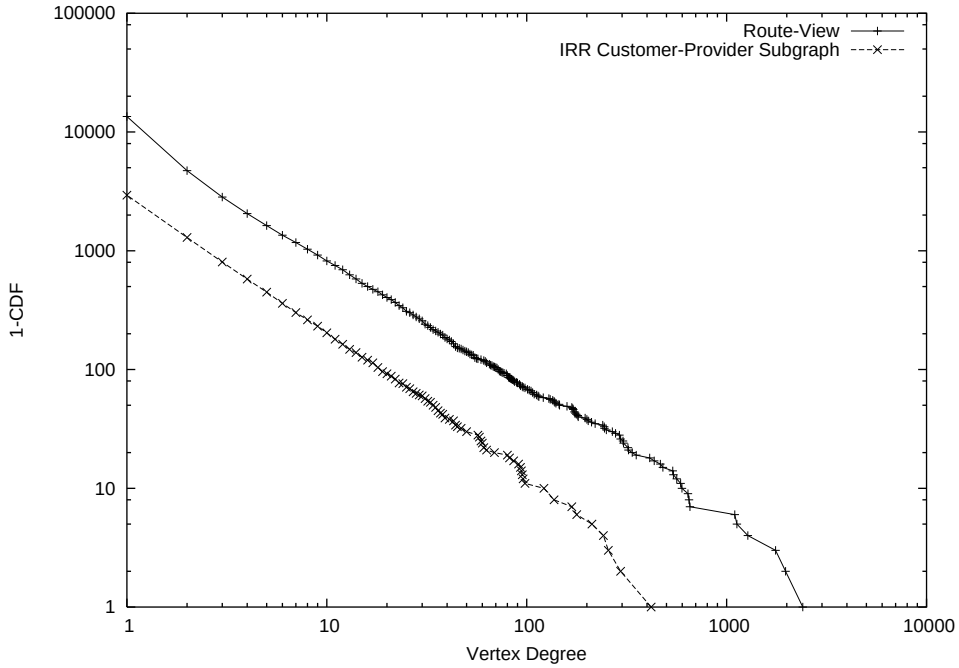


Figure 2.10: Vertex Degree Distribution of *customer-provider* Subgraphs

"stars" (i.e. the IRR database). As pointed up in [37], the vertex degree distribution of a graph may be differ from the distribution of a subgraph that is derived from the original graph. In particular, given a random graph, the vertex degree distribution of a subgraph formed by a collection of shortest paths trees from a set of sources to a set of destination, may be heavy-tailed. Thus, one may question our inferences by suggesting that these subgraphs do not represent the vertex degree distribution of the full graph.

However, in [37] the set of destination vertices is very small compare to the number on vertices in the graphs (only 1%) while when we consider BGP routing table, the destination set contains almost all the vertices in the graph. The significance of this difference is depicted in Figure 2.14. One can observe that when the size of the set of destination is small, the vertex degree distribution of the derived subgraph follow the power-law, but increasing the size of the set bring the distribution of the derived subgraph closer to the distribution of the full random graph.

While the Route-Views database represent a policy-based shortest path trees subgraph, the IRR database may represent a random subgraph (generated by a set of random stars) and does not follow the last discussion. Our simulation results indicate that in this case the vertex degree distribution of the subgraph is similar to the full graph. Figures 2.15 and 2.16 depicts the vertex degree distribution of small and big subgraphs generated by a set 1500 and 5000 random stars respectively. Both subgraphs preserve the vertex degree distribution of the full graph.

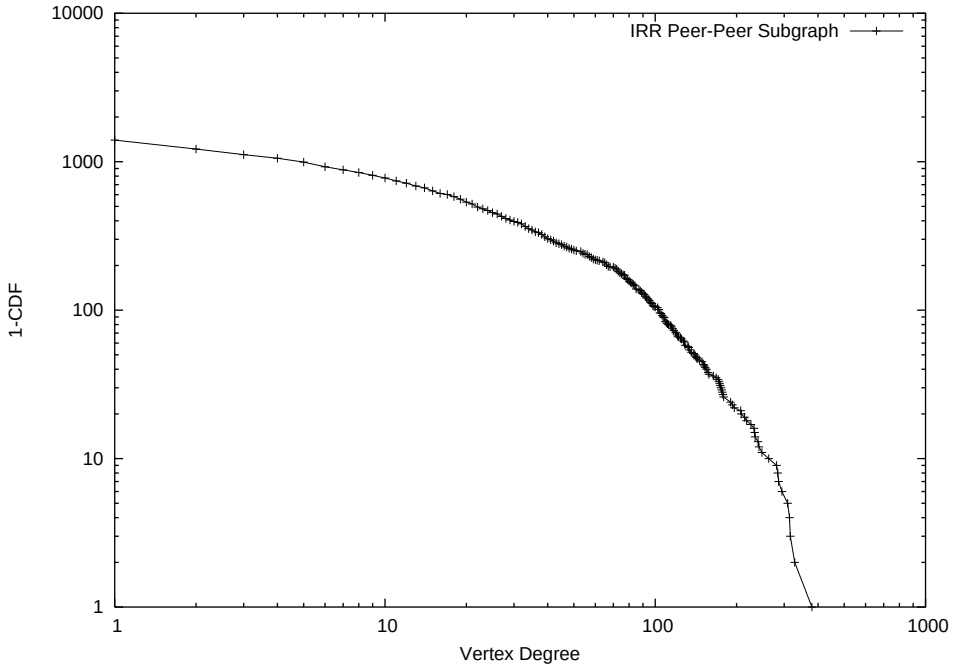


Figure 2.11: Vertex Degree Distribution of *peer-peer* Subgraph

2.4 Summary

In this chapter we showed that despite the increasing effort to unveil the AS connectivity map at least 35% of the links are still missing from all known databases. Less conservative estimations indicate that more than 50% of the link remain hidden. By understanding the gathering process of databases such Route-Views and IRR we showed that almost all missing links are of type *peer-peer* while a considerable amount of *customer-provider* links are revealed. Thus, trying to disclose the full AS connectivity graph by an increasing set of BGP routing table or by a set of agents performing periodic *traceroute* (both discover mostly *customer-provider* links) may be insufficient in order to fully unveil the *peer-peer* subgraph. A better understanding and modelling the structure of these unveiled *peer-peer* links and their location in the hierarchical structure is a subject to future work. Note that unlike the Route-View and IRR databases, at this time the DIMES project is relatively new and a more thorough study of its information gathering is in place.

We also studied the vertex degree distribution of the AS connectivity graph and showed that the distribution of the *peer-peer* subgraph is considerately different from the one of the *customer-provider* subgraph. These inferences, may lead to new models describing the AS connectivity map that consist of two separate models. One describing the *peer-peer* subgraph and another describing the *customer-provider* subgraph. In particular, these models should take into account our finding regarding the vertex degree distribution of each subgraph. Namely, the vertex degree distribution of the *customer-provider* subgraph follows the power-law and the the vertex degree distribution of the *peer-peer* subgraph is similar to the distribution of a Waxman graph.

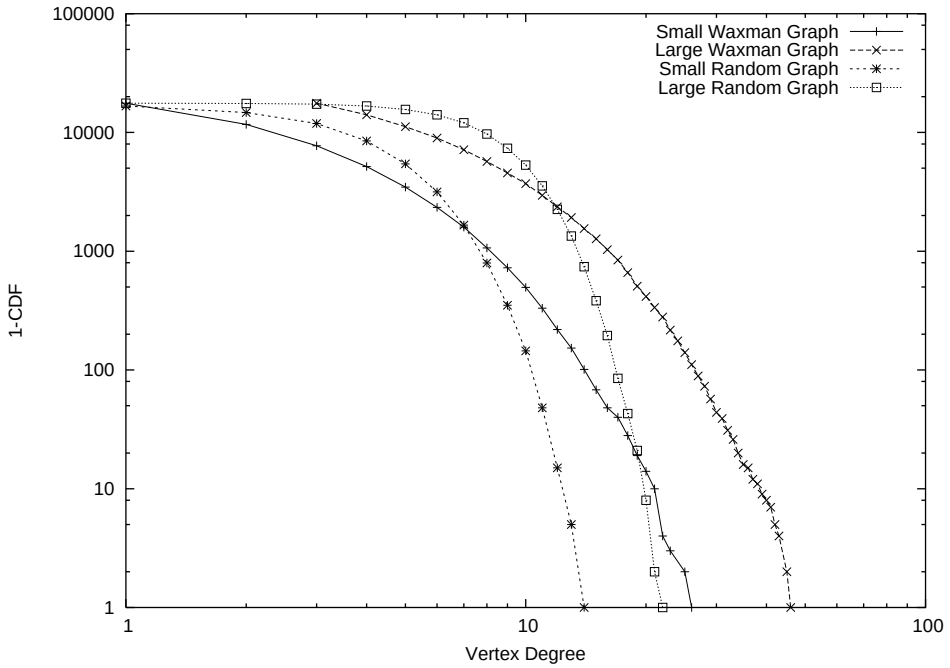


Figure 2.12: Vertex Degree Distribution of Several Graphs

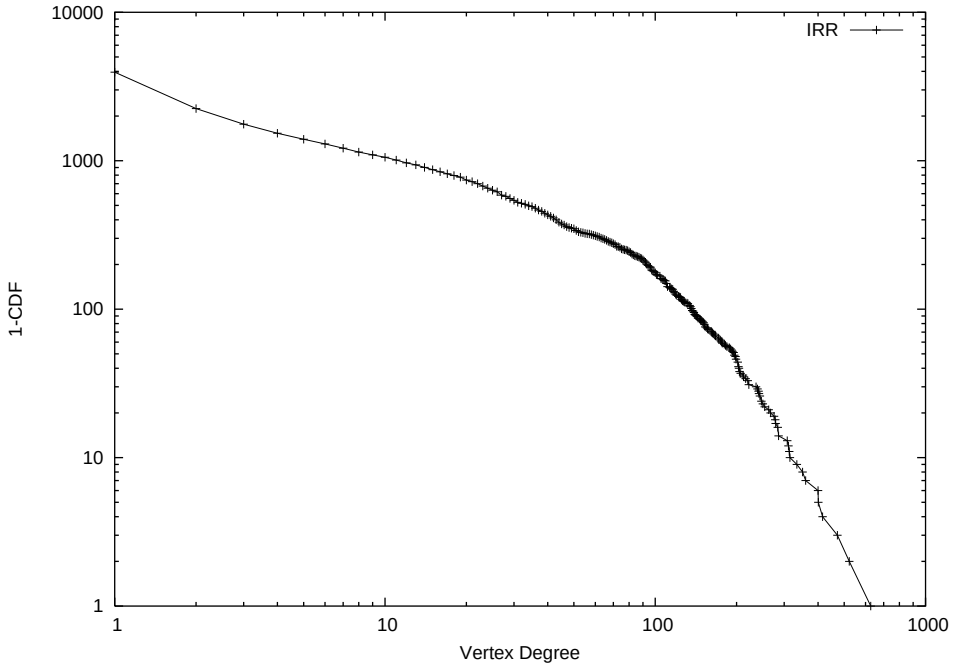


Figure 2.13: IRR Vertex Degree Distribution

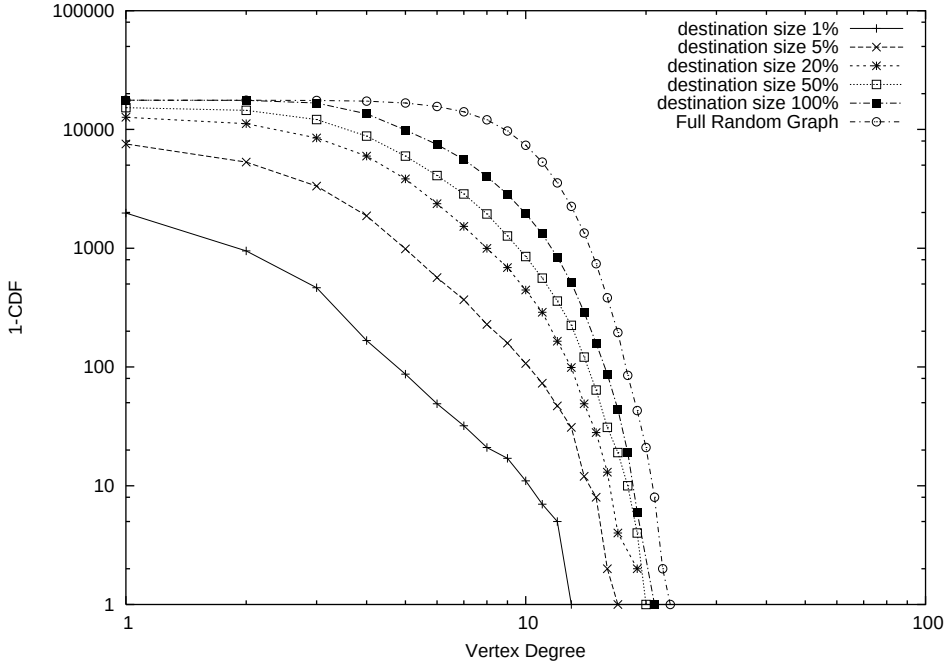


Figure 2.14: Vertex Degree Distribution of a Shortest Path Subgraph

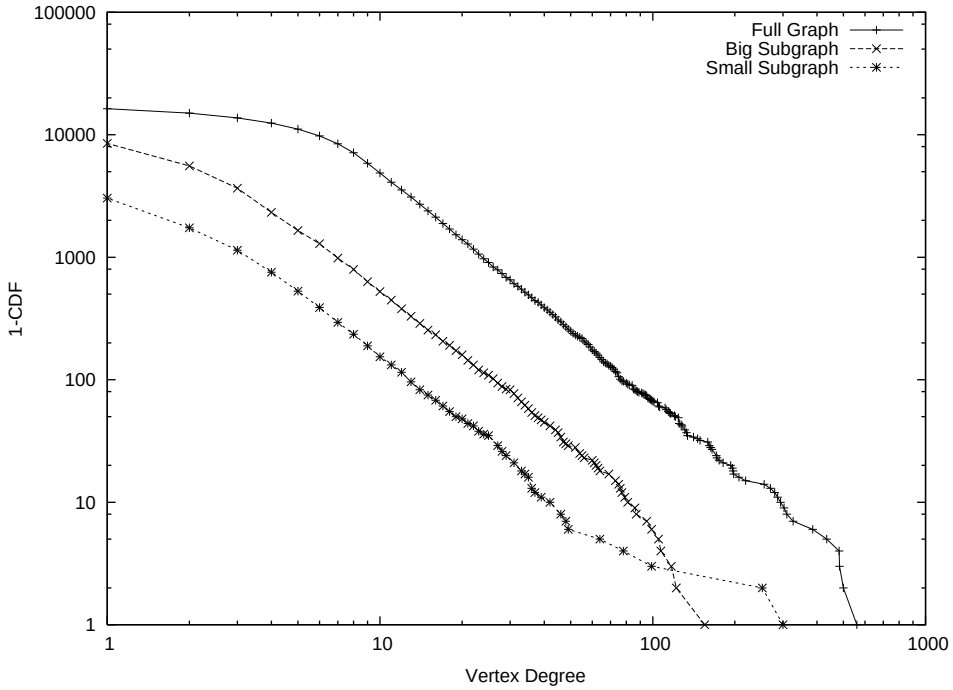


Figure 2.15: Vertex Degree Distribution of a Barabasi-Albert Subgraphs

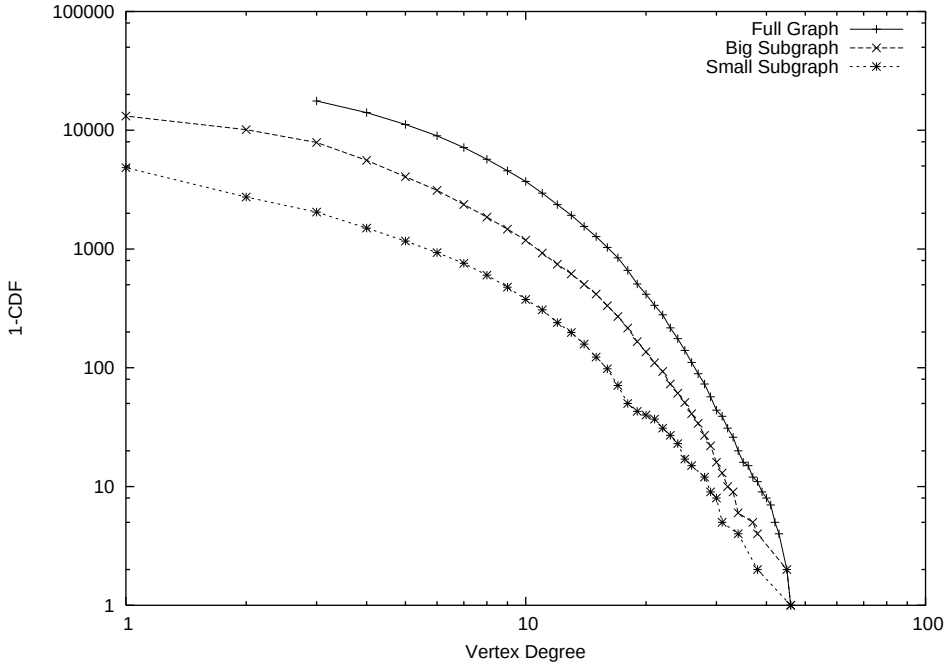


Figure 2.16: Vertex Degree Distribution of a Waxman Subgraphs

Chapter 3

Acyclic Type of Relationships[1]

In this chapter we study the Acyclic Type of Relationship ($ATo R$) problem, taking into account the acyclic structure of the AS connectivity graph. Given a set of routing paths, the objective function is to maximize (or minimize) the number of valid (invalid) paths, keeping the directed graph acyclic. This new problem captures the type of relationship between connected ASes more accurately compare to the Type of Relationship (ToR) problem studied in [11, 48, 24].

We present an efficient algorithm that allows determining if there is a hierarchical assignment without invalid paths, show that the related general optimization problem is NP-complete, and present a $\frac{2}{3}$ approximation algorithm where the objective function is to minimize the total number of local policy mismatches. We support our approach by extensive experiments and simulation results showing that our algorithms classify the type of relationship between ASes much better than all previous algorithms.

3.1 Model and Problem Definition

As discussed in Chapter 1, the AS connectivity map is modeled by a graph $G = (V, E)$. Assigning an orientation to a particular edge in the graph indicates the business relationship between its corresponding ASes. Thus, an edge (v, u) is directed from v to u if v is a customer of u (and respectively u is a provider of v). On the other hand, an undirected edge indicates that the corresponding ASes are connected by *peer-peer* relationship. As explained in Chapter 1, a directed cycle that contains at least one directed edge (i.e., a *customer-provider* edge) violates the hierarchical structure of the graph. On the other hand, ASes from the same hierarchy level can be connected by *peer-peer* links. Thus, cycles that consist of undirected edges alone (i.e., all the links composing these cycles are of type *peer-peer*), implying that the participants ASes are in the same hierarchy level, are permitted. With respect to this observation, we use the term *hierarchical cycle* to describe a directed cycle that contains at least one directed edge. We use the term *valid path* to indicate that the path does not contain valleys nor steps. Thus, $p = \{v_1, v_2, ..., v_n\}$ is a valid path if for all $1 < i < n$, the edge (v_{i-1}, v_i) is directed from v_{i-1}, to v_i or the edge (v_{i+1}, v_i) is directed from v_{i+1}, to v_i. A path is *invalid* if it is not *valid*. Considering the hierarchy structure of the AS

[1]The results described in this chapter appeared also in [17].

graph and using these policy guidelines, we define the *AToR* (Acyclic Type of Relationship) problem as follows:

Definition 3.1.1 *Given an undirected graph G and a set of paths P, assign orientation to some of the edges of G such that the directed graph does not contain* hierarchical cycles *(i.e., directed cycles that contains at least one directed edge), and the number of valid paths is maximized.*

While the *AToR* problem considers links of types *customer-provider* and *peer-peer*, other types of relationships

One may observe that *sibling-sibling* edges are not part of the problem definition. Using the guidelines described in Chapter 1 a *sibling-sibling* link does not have any export limitation, thus, considering this type of links in the definition would lead to a trivial unrealistic solution in which all the links in the graph are of type *sibling-sibling*. Thus, we first find a solution that does not contain *sibling-sibling* links and then we refine our algorithm to include such edges (see Section 3.4).

In some cases an instance to the *AToR* problem includes only the set of paths P while the graph G is omitted. In such cases, one may consider a graph $G' = (V', E')$ that is imposed by the set of paths, namely $V' = \{v|v \in P\}$, and $E' = \{e|e \in P\}$. The decision version of the problem, *k-AToR*, is defined as follows:

Definition 3.1.2 *Given an undirected graph G, a set of paths P, and an integer k, test if it is possible to give orientation to some of the edges of G such that the directed graph does not contain* hierarchical cycle*, and the number of invalid paths is at most k.*

In sections 3.2 and 3.3 we present theoretical analysis both for the *AToR* and the *k-AToR* problems. In these analyses we consider solutions that contain only directed edges (i.e., edges of type *customer-provider*). One can argue that this requirement is stricter than necessary and therefore does not reflect the real practical problem. Yet, as we prove in the next lemma, every solution that contains *peer-peer* links can be converted to a solution that contains only *customer-provider* links by giving an orientation to the *peer-peer* links. Clearly, in a solution that contains only directed edge, every directed cycle is a *hierarchical cycle* and acyclic solution is a solution that does not contain *hierarchical cycles*.

Lemma 3.1.1 *Given an instance (G, P) to the* AToR *problem and a solution S that assigns an orientation to some of the edges of G, such that the directed graph does not contain* hierarchical cycles *and the number of valid paths is k, there is a solution S' that assigns an orientation to* ***all*** *the edges of G, such that the directed graph does not contain directed cycles and the number of valid paths is at least k.*

Proof Denote by E_1 the set of edges that have an orientation with respect to S, and denote by $G_1 = (V, E_1)$ the graph imposed by this set of edges. Clearly, G_1 does not contain directed cycles. We build S' gradually by assigning orientation to the set of undirected edges $E \setminus E_1$ (i.e., converting the set of *peer-peer* links into *customer-provider* links). We show that each such step does not reduce the number of valid paths and preserves the acyclic property of the graph.

Consider a *peer-peer* link $e = (v, u)$, namely $e \in E \setminus E_1$. The graph G_1 does not contain directed cycles therefore, if there is a directed path $p = (v, ..., u)$ then there is no directed path $p' = (u, ..., v)$ (otherwise, their concatenation is a directed cycle in contradiction to the assumption)[2]. Thus, if there is a directed path from v to u we assign e from v to u. Otherwise, we assign e from u to v. In both cases, after adding e to E_1, the graph G_1 is still acyclic.

Next, we show that every valid path remains valid by assigning direction to a link e. Clearly, paths that do not traverse e are not affected so we need to show that every valid path that traverses e remains valid. Consider a valid path that traverses e. According to the discussion above, this path consists of N *customer-provider* links followed by e which is a *peer-peer* link followed by M *provider-customer* links (where $N, M \geq 0$). If e is assigned to be a *customer-provider* links then the new path consists of $N + 1$ *customer-provider* links followed M *provider-customer*. On the other hand if e is assigned to be a *provider-customer* links then the new path consists of N *customer-provider* links followed $M + 1$ *provider-customer*. In both cases the new path is valid. ∎

3.2 The *0-AToR* Problem

In this section we present an efficient algorithm for the *0-AToR* problem. In other words, given an instance (G, P) to the *AToR* problem, the algorithm determines if there is a solution without any invalid paths. In such case, the algorithm finds such a solution. For simplicity, we present and analyze the algorithm over a special case of the *0-AToR*, called the *0-AToR2* problem, in which the length of all paths is exactly two links. Although this version seems to be simpler than the *0-AToR* problem, it does not. In particular, in the following lemma we show that the complexity of the *0-AToR2* problem is identical to the complexity of the *0-AToR* problem[3].

Definition 3.2.1 *Given an instance (G, P) to the* 0-AToR *problem, we say that there is a* Satisfying *assignment to (G, P) if there is an orientation to the edges such that the directed graph is acyclic and all the paths are valley-free.*

Lemma 3.2.1 *Given an instance (G, P) to the* 0-AToR *problem, there is an instance (G, P_2) to the* 0-AToR2 *problem such that there is a* Satisfying *assignment to (G, P) if and only if there is a* Satisfying *assignment to (G, P_2).*

Proof P_2 is generated in the following way: for each $p = \{AS_1, AS_2, ...AS_n\} \in P$ produce $n - 2$ paths $p_1, ...p_{n-2}$ such that $p_i = \{AS_i, AS_{i+1}, AS_{i+2}\}$. For each assignment, if the path p_i is invalid than the path p is invalid as well (it contains valley in AS_i, AS_{i+1}, AS_{i+2}). On the other hand, if the path p is invalid, at least one of the paths $p_1, ...p_{n-2}$ is invalid (if p contains valley at $(AS_i, AS_{i+1}, AS_{i+2})$ then the path p_i contains a valley as well). Clearly, an acyclic assignment to (G, P), induces an acyclic assignment to (G, P_2) and vice versa, since both instances consist of the same graph. ∎

[2]Note that p and p' are not necessarily in P.

[3]Recall that in this section we consider solutions that assign an orientation to all the edges. In this case, a valid path is a valley-free path (i.e., a path that does not contain valleys).

As discussed above, the directed graph imposed by a solution to an instance of the *0-AToR2* problem does not contain directed cycles. Thus, one can present this solution as an ordering π of the vertices of a graph such that for each directed edge $(v_i, v_j) \in E$, going from v_i to v_j, $\pi(v_i) < \pi(v_j)$. Clearly, the directed graph induced by such ordering is acyclic. Moreover, this solution does not contain valleys, namely for each path $p = (v_i, v_j, v_k)$ $\pi(v_j) > \pi(v_i)$ or $\pi(v_j) > \pi(v_k)$. The following algorithm determines if such an ordering exists.

Algorithm *ATOR*$(G = (V, E), P = (p_1, ..., p_n))$

1. $P_2 = \phi$
2. For all $p_i = \{AS_1, AS_2, ...AS_m\} \in P$.
3. for$(i = 1$ to $m - 2)$
4. $P_2 = P_2 \cup \{AS_i, AS_{i+1}, AS_{i+2}\}$
5. i=1
6. For all $v \in V, \pi(v) = -1$.
7. while $P_2 \neq \phi$ do
8. Find $v \in V$ such that $\forall p = (v_i, v_j, v_k) \in P_2$, $v \neq v_j$
9. If such v does not exist return *NO SOLUTION*.
10. set $\pi(v) = i$.
11. i=i+1.
12. P_2‘ $= \{p | v \in p\}$.
13. $P_2 = P_2 \setminus P_2$‘
14. while $i \leq |V|$
15. if $\pi(v) = -1$, set $\pi(v) = i$.
16. $i = i + 1$.
17. return π.

In the first stage (steps 1 to 4) the algorithm generates a set of paths P_2 according to the construction described in Lemma 3.2.1, such that the length of each path in P_2 is exactly two and (G, P) has a *Satisfying* assignment if and only if (G, P_2) has a *Satisfying* assignment. Steps 5 and 6 are for initialization. Then, in every iteration the algorithm finds a vertex that does not appear in the middle of any path. Giving this vertex the current lowest value in the ordering insures that the associated paths are valid. On the other hand, if such vertex does not exist, it means that at least one path contains this vertex in the middle. Thus, giving this vertex the current lowest value in the ordering makes this path invalid.

In each iteration at least one path is removed (steps 12 and 13), and thus at most $|P_2|$ iterations are performed. In each iteration, the algorithm goes through the vertices and pick one vertex (Step 8). Moreover, $|P_2| = |P| \cdot (N - 2)$ where N is the average length of a path in P. Therefore, the time complexity of the algorithm is $O(|P| \cdot N \cdot |V|)$.

If the algorithm finds a solution (i.e., it does not return *NO SOLUTION*) the peering relationships are induced as follow: For each edge (v, u) in the graph, v is a customer of

u (and respectively u is a provider of v) if and only if $\pi(v) < \pi(u)$. Next we show the correctness of the algorithm.

If an instance (G, P) of the *0-AToR* problem has a *Satisfying* assignment, clearly a sub-instance (i.e., an instance that consists of a subset of P) has a *Satisfying* assignment as well. Thus,

Observation 3.2.1 *Given an instances (G, P) and (G, P') to the* 0-AToR *problem, such that $P' \subseteq P$. If (G, P') does not have a* Satisfying *assignment then (G, P) does not have a* Satisfying *assignment.*

Given an instance $(G = (V, E), P)$ of the *0-AToR* problem, denote by $G_p = (V_p, E_p)$ the graph imposed by P (i.e., $V_p = \{v | v \in P\}$ and $E_p = \{e | e \in P\}$). Clearly, the ordering of any vertex v such that $v \in V \setminus V_p$ does not affect the validity of any path (since this vertex does not appear in any path), thus:

Observation 3.2.2 *Given an instance (G, P) to the* 0-AToR *problem. (G, P) has a* Satisfying *assignment if and only if (G_p, P) has a* Satisfying *assignment.*

Theorem 3.2.1 *Given an instance (G, P) to the* 0-AToR *problem, if Algorithm* ATOR *returns ordering π of the vertices of a graph, then this ordering induces a* Satisfying *assignment.*

Proof Clearly, the directed graph induced by the ordering is acyclic. We show that this directed graph is valley-free with respect to the set of paths P_2. Without loss of generality, assume that $p = (v_x, v_y, v_z)$ is removed from P_2 in the i'th iteration. According to steps 8 and 10 of the algorithm $\pi(v_x) = i$ or $\pi(v_z) = i$. Moreover, $\pi(v_y)$ was not set yet, thus $\pi(v_y) > i$ and therefore p is valley-free.

Now, recall that P_2 is constructed according to Lemma 3.2.1, thus the directed graph imposed by the ordering π, is valley-free with respect to the set of paths P as well. ∎

Theorem 3.2.2 *Given an instance (G, P) of the* 0-AToR2 *problem, if Algorithm* ATOR *returns* NO SOLUTION*, then there is no* Satisfying *assignment.*

Proof Without loss of generality, assume that the algorithm returns *NO SOLUTION* in the i'th iteration and the set of paths that was removed in the first $i-1$ iterations is P_2''. Denote by P_2' the remaining paths, namely $P_2' = P_2 \setminus P_2''$. We show that the instance (G', P_2'), does not have a *Satisfying* assignment, where $G' = (V', E')$ is a subgraph of G in which $V' = \{v | v \in P_2'\}$ and $E' = \{(v, u) | (v, u) \in E \bigcap v, u \in V'\}$. Assume that there is an ordering π that induces a *Satisfying* assignment. Denote by v the node with $\pi(v) = 1$. Recall that $v \in V'$, therefore $v \in P_2'$. According to steps 8 and 9 in the algorithm, $\forall v \in P_2', \exists p = (v_x, v_y, v_z) \in P_2'$ such that $v \equiv v_y$. Thus, since $\pi(v_x) > 1$, $\pi(v_z) > 1$, and $\pi(v_y) = 1$, p contains a valley, so (G', P_2') does not have a *Satisfying* assignment. According to Observation 3.2.2, (G, P_2') does not have a *Satisfying* assignment and according to Observation 3.2.1 (G, P_2) does not have a *Satisfying* assignment. Again, recall that P_2 is constructed according to Lemma 3.2.1, thus the instance (G, P) does not have a *Satisfying* assignment as well. ∎

Discussion: One may notice that the *AToR2* problem resembles to the well known *BETWEENNESS* problem, studied in the field of Computational Biology [15]. The input to the

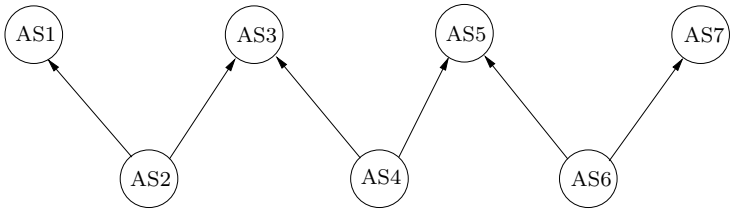
BETWEENNESS problem consists of a set of points $S = \{x_1, x_2, ..., x_n\}$ and a set of constraints, where each constraint is a triplet $\{x_i, x_j, x_k\} \in S \times S \times S$. A solution to the problem is a total order on its points such that every constraint $\{x_i, x_j, x_k\}$ satisfies $x_i < x_j < x_k$ or $x_k < x_j < x_i$, namely x_j is *between* x_i and x_k (see [15]). With respect to our problem, the set of triplet corresponds to the set of paths with length two. While in the *BETWEENNESS* problem x_j must be between x_i and x_k, in the *AToR* problem x_j can be between x_i and x_k or bigger than x_i and x_k (i.e., x_j must not be lower than x_i and x_k). Despite the appeared similarity between the problems their complexity is completely different. Thus, while determining if there is a *satisfying* assignment to the *AToR2* problem (and finding such an assignment) can be done in polynomial time (using the algorithm described above), determining if there is a *satisfying* assignment to the *BETWEENNESS* problem is NP-hard [41]. The intuition behind this difference is the fact that in the *AToR* problem, determining the ordering of one edge of a path satisfies this path (i.e., the path is valid). On the other hand determining the ordering of single point in a triplet, does not necessarily satisfy this triplet, in the *BETWEENNESS* problem.

3.3 The *k-AToR* Problem

While determining if there is a solution without invalid paths (i.e., finding a solution to the *0-AToR* problem) can be done in polynomial time, the general case is much more complex. In particular, the reduction presented in [11] for the *ToR* problem holds for the *AToR* problem. Thus, the general decision version (called the *k-AToR* problem) is NP-hard and the maximum version of the problem (i.e., maximizing the number of valid paths) cannot be approximated within $1/n^{1-\epsilon}$ (for any $\epsilon > 0$) for general instances with n paths unless NP=co-RP.

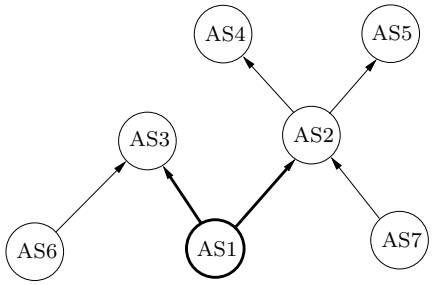
In this section we consider a variant of the problem in which the objective function is to minimize the total number of valleys. This problem is different from the original problem since one invalid path may contain more than one valley (see Fig. 3.1), and on the other hand, if several paths traverse a common AS, one valley (in the common AS) may cause several invalid paths (see Fig. 3.2). The objective function of this variant of the problem is motivated by the fact that in some cases ASes (mistakenly or purposely) do not follow the export policy discussed in Chapter 1. In particular, in such cases, an AS may export its provider routes to other providers. Thus, due to the locality of the policy, paths that traverse such ASes may contain valleys. Another reason that can explain valleys is the existence of links of type *sibling-sibling*. As discussed in Chapter 1 and in Section 4.1 these links does not have any export limitation but they are not part of the problem definition. In Section 3.4 present a heuristic that settle valleys by links of type *sibling-sibling*. We show that the decision version of this variant of the problem is NP-hard using a reduction from the Feedback Vertex Set (FVS) problem [34, 28], and we present a $\frac{2}{3}$-approximation algorithm to the maximum version of the problem.

Consider an instance of the *k-AToR* problem in which all the paths contain exactly two edges (we refer this problem as *k-AToR2* problem). In this case an invalid path contains exactly one valley and therefore if there is no duplicated paths, every valley corresponds to one invalid path. Thus, given a graph G and a set of paths P, we build an instance (G, P_2) to the *k-AToR2* such that for any assignment, the number of valleys with respect to the



p={AS1, AS2, AS3, AS4, AS5, AS6, AS7}

Figure 3.1: One invalid path - Multiple valleys



p_1 ={AS7, AS2, AS1, AS3, AS6}
p_2 ={AS5, AS2, AS1, AS3}
p_3 ={AS4, AS2, AS1, AS3}

Figure 3.2: One valley - Multiple invalid paths

set of paths P is equal to the number of invalid paths with respect to the set of paths P_2. Thus, for each $p = \{AS_1, AS_2, ...AS_n\} \in P$ we produce $n-2$ paths $p_1, ...p_{n-2}$ such that $p_i = \{AS_i, AS_{i+1}, AS_{i+2}\}$, then we remove all the duplicate paths. In this case, given an assignment of the edges in G, every valley in the edges (v_1, v_2) and (v_2, v_3) with respect to P, corresponds to invalid path (v_1, v_2, v_3) with respect to P_2. Based on this observation, henceforth we consider only the *k-AToR2* problem.

Theorem 3.3.1 *The* k-AToR2 *problem is NP-hard.*

Proof We prove the Theorem using a reduction from the Feedback Vertex Set (FVS) problem [28]. In the FVS, given a directed graph $G = (V, A)$ and a positive integer $k \leq |V|$, one should determine if there is a subset $V' \subseteq V$ such that $|V'| \leq k$ and the deletion of V' results in a directed acyclic graph.

Given an instance to the FVS problem, a directed graph $G = (V, A)$ and a positive number k, we construct an instance to the *k-AToR2* problem that consist of $|V| + 2|A|$ paths. For each $v \in V$ we match a path $p_v = \{v_1, v_2, v_3\}$. For each directed edge $(v, u) \in A$ we match two paths: $p^1_{v,u} = \{v_2, v_3, u_1\}$ and $p^2_{v,u} = \{v_3, u_1, u_2\}$. For instance, the graph depicted in the upper part of Fig. 3.3 is transformed to the the *k-AToR* instance depicted in the bottom part of Fig. 3.3.

Now we show that the reduction holds, namely there is a subset $V' \in V$ such that $|V'| \leq k$ and the graph $G' = (V \setminus V', A)$ is acyclic if and only if there is a assignment to the corresponding *k-AToR2* instance that contains less than k invalid paths.

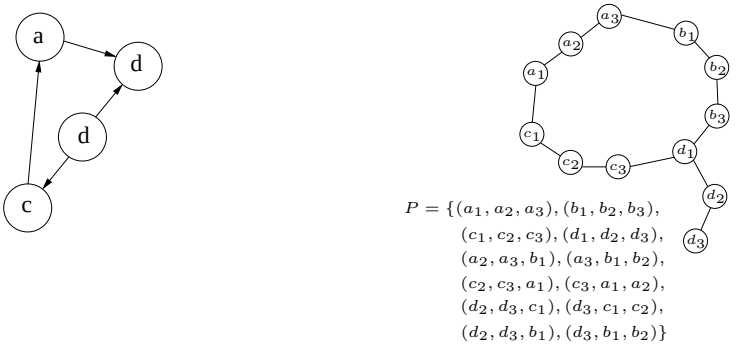


Figure 3.3: Example: FVS to *k-AToR* reduction

In the reduction presented above, a directed cycle $v^1, v^2, ..., v^m, v^1$ with respect to the FVS instance induces the following set of paths with respect to the corresponding *k-AToR* instance:

$$(v_1^1, v_2^1, v_3^1), (v_2^1, v_3^1, v_1^2), (v_3^1, v_1^2, v_2^2), (v_1^2, v_2^2, v_3^2), ...$$
$$..., (v_1^m, v_2^m, v_3^m), (v_2^m, v_3^m, v_1^1), (v_3^m, v_1^1, v_2^1)$$

One can see that in this set of paths every vertex is traversed by at least one path. Thus, according to the algorithm presented in Section 3.2, this set of path must contain at least one invalid path. According to this observation, given an optimal solution V_{opt} to a FVS instance, any assignment to the corresponding *k-AToR* instance contains at least $|V_{opt}|$ invalid paths. Thus, if an instance to the FVS problem does not have a solution (i.e., for all $V' \subseteq V$ such that $|V'| \leq k$, the graph $G = (V \setminus V', A)$ contains directed cycles), then the corresponding *k-AToR* instance does not have an assignment with at most k invalid paths.

To show the other direction, we use the following observation

Observation 3.3.1 *An instance (G, P) to the* AToR *problem has a solution that contains k invalid paths, where $P' \subseteq P$ is the set of invalid paths, if and only if $(G, P \setminus P')$ has a feasible solution with respect to the* 0-AToR *problem. In other words, removing the set of invalid paths imposed by a solution to the* AToR *problem induces a feasible solution to the* 0-AToR *problem.*

Given, a solution V' to a FVS instance $G = (V, A)$ (i.e., $V' \subseteq V$, and $|V'| \leq k$ such that $G' = (V \setminus V', A)$ is acyclic), denote by P' the set of paths induced by the set of vertices V' in the corresponding *k-AToR* instance, namely, $P' = \{(v_1, v_2, v_3) | v \in V'\}$. We show that the instance $(G_{FVS}, P \setminus P')$ has a feasible assignment to the *0-AToR* problem using the ordering approach presented in Section 3.2. Thus, According to Observation 3.3.1, there is a assignment that contains at most k invalid paths with respect to (G_{FVS}, P).

Without loss of generality, assume that $V' = \{v^1, v^2, ..., v^k\}$. Since, $\forall i \leq k$, $(v_1^i, v_2^i, v_3^i) \notin P \setminus P'$, the vertices v_2^i are not traversed by any path (v_2^i may be an end point of some paths). Thus, according to the algorithm presented in Section 3.2 we can set $\pi(v_2^i) = i$ for

all $i \leq k$. According to steps 12 and 13 of the algorithm, all the paths, attached to v_2^i are removed, thus, the vertices v_1^i and v_3^i (for all $i \leq k$) are not traversed by any path, and we can set $\pi(v_1^i) = k + i$ and $\pi(v_1^i) = (2 \cdot k) + i$. The set of the remaining paths, after all the paths attached to v_1^i and v_3^i are removed, are corresponded to the acyclic directed graph $G' = (V \setminus V', A)$ with respect to the reduction presented above. At this stage, we set an order on the vertices of the *k-AToR2* instance according to the acyclic structure of G'. In particular we repeat the following procedure until all the vertices are set: First, we find a sink v in G. Since v is a sink, the corresponding vertex v_1 is not traversed by any path, thus, we set $\pi(v_1)$ to be the current lower value in the ordering. By removing the path attached to v_1 (i.e., (v_1, v_2, v_3)), we can set $\pi(v_2)$ to be the current lower value in the ordering (since (v_1, v_2, v_3) is the only path traversed v_2). Then, we remove all the paths attached to v_2 (these paths are corresponded to the set of outgoing edge from v) and we can set $\pi(v_3)$ and remove all the paths attached to v_3. ■

One can see that the reduction presented above is an approximation preserving reduction. Accordingly, the minimization version of *k-AToR2* problem is APX-hard [33]. Moreover, the best known approximation result for the FVS problem is $O(\log |V| \log \log |V|)$ [21]. Thus, any algorithm with better approximation ratio with respect to the *k-AToR2* problem, will directly improve the best known approximation result of the FVS.

Next, we present a $\frac{2}{3}$-approximation algorithm to the maximization *k-AToR2* problem. For each vertex $v \in V$, denote by E_v the set of the paths such that v is an end point of each path in the set. Similarly denote by M_v the set of the paths such that v is in the middle of each path in the set. Formally, $E_v = \{p | p \in P, p = \{v, u, w\} \cup p = \{u, w, v\}\}$, $M_v = \{p | p \in P, p = \{u, v, w\}\}$.

Algorithm *k-ATOR2* $(G = (V, E), P = (p_1, ..., p_n))$

1. $P_2 = \phi$
2. For all $p_i = \{AS_1, AS_2, ...AS_m\} \in P$.
3. for$(i = 1$ to $m - 2)$
4. $P_2 = P_2 \cup \{AS_i, AS_{i+1}, AS_{i+2}\}$
5. i=1
6. For all $v \in V, \pi(v) = -1$.
7. while $P_2 \neq \phi$ do
8. Let v be the vertex in V such that $\forall u \in V, |M_v|/|E_v| \leq |M_u|/|E_u|$
9. set $\pi(v) = i$.
10. i=i+1.
11. P_2‘ $= \{p | v \in p\}$.
12. $P_2 = P_2 \setminus P_2$‘
13. while $i \leq |V|$
14. if $\pi(v) = -1$, set $\pi(v) = i$.
15. $i = i + 1$.
16. return π.

The algorithm is similar to the algorithm presented in Section 3.2. In particular, the algorithm presented in Section 3.2 is a special case of this algorithm in which the vertex, selected in Step 8 must follow $|M_v|/|E_v| = 0$ (i.e., $|M_v| = 0$)[4]. Next we prove that this algorithm is a $\frac{2}{3}$-approximation algorithm.

Lemma 3.3.1 *The vertex v that is selected in Step 8 fulfils $|M_v|/|E_v| \leq \frac{1}{2}$.*

Proof In every path $p = \{v_1, v_2, v_3\}$ two vertices (v_1 and v_3) are end points and one vertex (v_2) is in the middle. Thus, $|P| = \sum_{v \in V} |M_v| = \frac{1}{2} \sum_{v \in V} |E_v|$, and at least one vertex $v \in V$ satisfies $|M_v|/|E_v| \leq \frac{1}{2}$ and therefore the vertex selected in Step 8 fulfil $|M_v|/|E_v| \leq \frac{1}{2}$. ∎

Theorem 3.3.2 *Given an instance (G, P_2) of the* k-AToR2 *problem, the Algorithm* k-ATOR2 *returns an ordering π on the vertices of the graph such that the directed graph induced by this ordering is acyclic and the total number of valley-free paths P_{valid} is at least $\frac{2}{3} \cdot |S^{opt}|$, where S^{opt} is the optimal solution.*

Proof The algorithm returns an ordering over the vertices, thus the directed graph induced by this ordering is acyclic. Assume that $p = (v_x, v_y, v_z)$ is removed from P_2 in the i'th iteration. If $p \in E_v$ then according to Step 9 in the algorithm $\pi(v_x) = i$ or $\pi(v_z) = i$. Moreover, $\pi(v_y)$ was not set yet, thus $\pi(v_y) > i$ and therefore p is valley-free. On the other hand, if $p \in M_v$ then $\pi(v_y) = i$ while $\pi(v_x)$ and $\pi(v_z)$ were not set yet. In this case $\pi(v_x), \pi(v_z) > i$ and p is an invalid path. According to this observation, in each iteration, $|E_v|$ paths turn to be valid paths while $|M_v|$ paths turn to be invalid paths. According to Lemma 3.3.1 $|M_v|/|E_v| \leq \frac{1}{2}$ therefore the total number of valid paths is at least double the number of invalid paths. In other words, $P_{valid} \geq 2(|P_2| - P_{valid})$ and hence $P_{valid} \geq \frac{2}{3} \cdot |P_2| \geq \frac{2}{3} \cdot |S_{opt}|$. ∎

One may observe that the proof is independent of the value of $|S^{opt}|$. Thus, the theorem proves a stronger result. In particular, the algorithm guarantees that at least $\frac{2}{3}$ of the paths are valley-free regardless on the optimal solution.

While we consider an objective function that minimizes the total number of valleys, other objective targets may be considered as well. In particular, one may consider an objective target that minimizes the total number of directed cycles. This objective target may be interesting from a theoretical point of view, but it seems that it does not have a practical interest. While invalid paths and valleys may appear in the internet for several reasons (e.g. export policy misconfiguration) the hierarchical acyclic structure of the internet is a result of the business relationship between ASes. Large ASes always provide services to small AS and this ordering imposes an acyclic structure.

3.4 Practical Consideration

In this section we show how the theoretical algorithms presented in sections 3.2 and 3.3 can be used to solve the practical problem of inferring the correct type of relationships from the collected data. In practice we may have to choose between more than one solution and we

[4]Note that E_v and M_v are defined with respect to the set of paths P_2 in the algorithm.

also have to consider other type of relationships, namely *peer-peer* and *sibling-sibling* that were not considered in these algorithms.

Consider the algorithms presented in sections 3.2 and 3.3. While these algorithms find a specific ordering, the space of possible solutions may contain more than one ordering. In particular, in Step 8 of both algorithms, more than one vertex may satisfy the required condition (i.e. minimizing $|M_v|/|E_v|$) and therefore different orders found by the algorithm may induce different peering relationships. In particular, the ordering found by the algorithm may induce peering relationship that are different from the real relationships. To obtain a more accurate solution, one should use heuristics and guidelines that consider the topological structure of the graph and the gathering process of the data[5].

Considering the fact that the overwhelming majority of ASes are small[6], most of the routing paths obtained by BGP routing tables are between small ASes (i.e. the source and the destination of the path are small ASes). Thus, most of the routing paths consist of several *customer-provider* links follows by several *provider-customer* links. For instance, consider the routing path depicts in Figure 3.4. One possible solution may determine that $AS(i)$ is a customer of $AS(i{+}1)$ (i.e. $AS7$ is the top provider). In this case, the vertices are picked by Step 8 of the algorithm in the following order: $AS1, AS2, AS3, AS4, AS5, AS6, AS7$. In a more realistic solution $AS4$ is defined to be the top provider. In this case, the vertices are picked by Step 8 of the algorithm in the following order: $AS1, AS7, AS2, AS6, AS3, AS5, AS4$. Our heuristic is based on this observation and we choose the vertices assigned to a single path in rotation, i.e. after picking a vertex from one end of a path[7] we prefer that the next vertex picked from this path will be from the second end of the path.

Figure 3.4: Space of Solutions

3.4.1 Finding *Peer-Peer* Relationships

The algorithms described in previous sections give orientation to all edges in the graph, and therefore determines a *customer-provider* relationship between adjacent ASes. In practice some of the edges may be of type *peer-peer*. While Lemma 3.1.1 proved that every *peer-peer* link can be converted to a *customer-provider* link without harming the solution, the opposite does not hold. Namely, not every *customer-provider* link can be converted to a *peer-peer* link.

For example, consider a solution to the *0-AToR* instance, depicted in Figure 3.5. By converting the link $(AS1, AS2)$ into *peer-peer*, the path $p_1 = (AS1, AS2, AS3)$ becomes invalid since *customer-provider* link cannot follow *peer-peer* link. On the other hand, converting the link $(AS3, AS4)$ into *peer-peer* will not affect the validity of any path, but it will violate the hierarchical structure of the graph (if AS3 and AS4 are connected by *peer-peer*, it means

[5]Note that enumerating all possible solutions is computationally infeasible.

[6]The term small AS refers to the hierarchy level of the AS

[7]In this case, we refer to the entire long path before it was sliced into short paths according Lemma 3.2.1.

that both ASes are in the same level in the hierarchical structure. Thus, AS4 cannot be a customer of a customer of AS3).

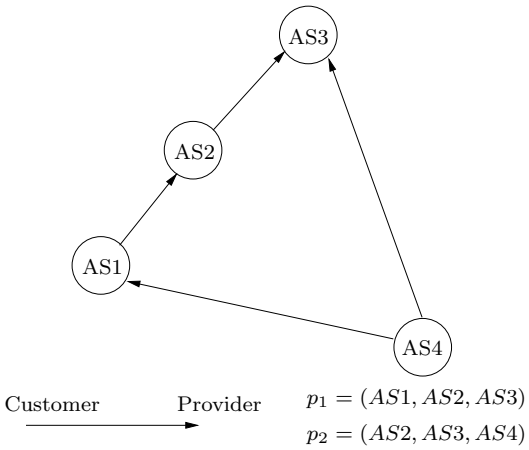


Figure 3.5: $(AS1, AS2)$ cannot be converted to *peer-peer* neither $(AS3, AS4)$

With respect to this observation, we need to assign *peer-peer* relationship to some of the links reserving the quality of the solution, namely without increasing the number of invalid paths and without violate the hierarchical, acyclic structure of the graph. We handle this task in two stages, during the execution of the algorithm, and after the assignment of the *customer-provider* links.

In the first step, we refine the algorithms by assigning *peer-peer* relationship to some of the links. In Algorithm *k-AToR* we omit Step 15, while in Algorithm *AToR* we omit Step 16 (i.e. $i = i + 1$). Thus, all the remaining vertices have the same value in the topological sort, which is greater than the values of all other vertices. The peering relationship is induced as follow: Given an edge (v, u), the edge is of type *peer-peer* if and only if $\pi(v) = \pi(u)$, and v is a customer of u if and only if $\pi(v) < \pi(u)$. Thus, all the edges connecting the remaining vertices are of type *peer-peer* while other edges are of type *customer-provider*. In this case, the graph does not contain *hierarchical cycle*, but cycles that consist of *peer-peer* links alone are possible. Recall that two ASes connected by *peer-peer* link, means that they are in the same hierarchy level. Thus, these kind of cycles are permitted and they do not violate the hierarchical structure of the AS graph (see Figure 3.6 for example). In addition, for each valid path $p = (v_i, v_j, v_k)$, $\pi(v_i) < \pi(v_j)$ or $\pi(v_k) < \pi(v_j)$ thus the following five options are possible:

$\pi(v_i) < \pi(v_j) = \pi(v_k)$, $\pi(v_i) < \pi(v_j) < \pi(v_k)$,
$\pi(v_i) < \pi(v_j) > \pi(v_k)$, $\pi(v_i) = \pi(v_j) > \pi(v_k)$,
$\pi(v_i) > \pi(v_j) > \pi(v_k)$

One can see that in all cases, the peering relationship that are derived from each assignment induces a valid path.

In the second step, we convert *customer-provider* links into *peer-peer* similar to the way described in [20]. First, we find a set of *peer-peer* candidates. A *customer-provider* link is added to this set if it can be converted to *peer-peer* link without violate the hierarchical structure of the graph and without increasing the number of invalid paths. While converting any single link from the candidate set is permitted, converting simultaneously more than one

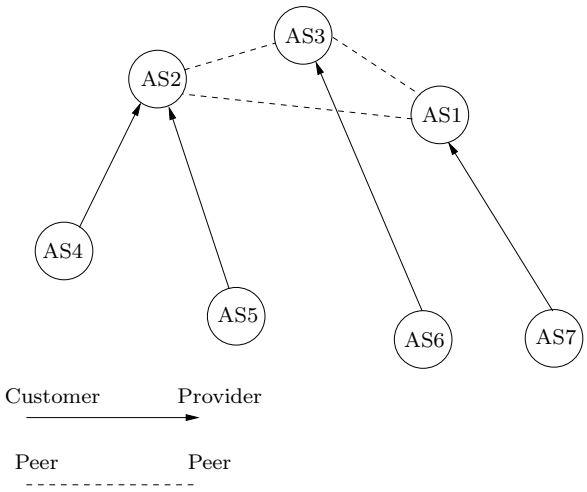


Figure 3.6: $AS1$, $AS2$, and $AS3$ are in the same hierarchy level

link may violate the hierarchical structure or increase the number of invalid paths. Thus, we convert the links in the set one after the other, preferring links that connect nodes that their vertex degrees are on the same order, testing each time if the graph remains acyclic and if the number of invalid path is not increased.

3.4.2 Finding *Sibling-Sibling* Relationships

A solution of the Algorithm *k-AToR* presented in Section 3.3 may contain invalid paths. In particular, when $|M_v|/|E_v| > 0$ with respect to Step 8 in the algorithm, it means that at least one valley traverses vertex v. While these kind of anomalies can be explained by an unexpected policy, it may reflect a *sibling-sibling* relationship between connected ASes. Thus, invalid paths may be settled by assigning *sibling-sibling* relationship to one of the edge on each valley. Similar to the discussion regarding *peer-peer* links in Section 3.4.1, converting a *customer-provider* link into *sibling-sibling* link may violate the hierarchical structure of the graph. We convert *customer-provider* link into *sibling-sibling* in the following way. First, we find a set of *sibling-sibling* candidates. This set consists of all the edges that are traversed by at least one valley. Then we convert the links in the set one after another, preferring links that connect nodes that their vertex degrees are on the same order, until the set is empty or until all the paths become valid, testing every step if the graph remains acyclic.

3.5 Experiments and Simulation Results

In this section we examine practical versions of the algorithms presented above over real data gathered from the Route-Views project. In particular we consider the approximation algorithm presented in Section 3.3 along with the heuristics discussed in Section 3.4. We also simulate the algorithms over several random graphs, and compare the obtained results to other approaches presented in [24, 48, 11].

We use data from April 2006 that consists of 21505 ASes, 45783 links and over 1,500,000

paths. First, we execute Algorithm *AToR* over the entire database. The algorithm returns *NO SOLUTION*, namely every assignment contains cycles or invalid path. Moreover, we found out using the algorithm presented in [11] (and similar to the results presented in [11] over older data), that the simpler *0-ToR* problem (i.e. regardless the existence of cycles), does not have a valley-free solution as well[8].

Next, we execute the different algorithms over the entire Route-Views database. To verify and evaluate the results obtained by the algorithms, one needs to compare these results against actual data, namely one needs to obtain information regarding the real type of relationship of links in the AS connectivity graph, and compare the relationships inferred by the algorithms against the relationships of these links. In contrast to previous work that validate the results against the type of relationships of few ASes (usually contacting the network administrators of these ASes and asking for internal information regarding the actual type of relationships of each AS [24, 20]), we use a general and much more extensive method consisting of a data, collected from the IRR database and and we use the methods presented in 2 to infer the peering relationships between these ASes. While the IRR database contains 36,000 links and the Route-Views database contains 49,500 links, the intersection of these databases consists of 10,000 links (i.e. 10,000 links appears in both databases). Thus, we compare the solution, obtained by the algorithms to the orientation induced by the IRR database with respect to these intersected links.

Result analysis and measurement: Given an edge (u, v), the export policy of u to v and the export policy of v to u determines the type of relationship of this edge [7, 31]. In particular, an AS may export its routes and its customer routes or it may export its routes and its customer routes, as well as all its provider or peer routes. We denote the first case as *Type I* and the second case as *Type II*. Table 3.1 show how the export policy of an edge (u, v) is derived from this export policy. For example, if u exports to v its routes and its customer routes (i.e. *Type I*) and u exports to v its routes and its customer routes, as well as all its provider or peer routes (i.e. *Type II*, it means that u is a customer of v.

Type of relationship $u - v$	Export policy of u to v	Export policy of v to u
customer-provider	Type I	Type II
provider-customer	Type II	Type I
peer-peer	Type I	Type I
sibling-sibling	Type II	Type II

Table 3.1: Export Policy and Type of Relationship

If the orientation of an edge, assigned by a specific algorithm is different from the actual orientation of the edge, it means that there is a mismatch between the export policy inferred by the algorithm to the actual export policy of one or both nodes. We say that the orientation of an edge is *fully mismatched* if the export policy of both nodes is incorrect. We say that the orientation of an edge is *partly mismatched* if the export policy of one node is incorrect.

[8]We refer to the *0-ToR* problem as the decision version of the *ToR* problem that determines if there is a solution without any invalid path.

Table 3.2 presents the number of partial and full mismatch edges of each graph and each algorithm (*CR* - our algorithm, presented in Section 3.3; *BPP* - the algorithm presented in [11]; *Gao* - the algorithm presented in [24]; and *SARK*- the algorithm presented in [48]). The table also depicts the number of policy mismatch, i.e. the total number of cases in which the export policy derived by the algorithm is different from the actual policy[9]. One can see that the results, obtained by our algorithm are better compared to the results obtained by other algorithms. Moreover, while the solution obtained by *CR* is acyclic, the solutions obtained by *BPP*, *SARK*, *Gao*, contain directed cycles. An interesting result is that the number of partial mismatch (in all algorithms) is extremely high compared to the number of full mismatch which may emphasize the difficulty to identify *peer-peer* relationship (a partial mismatch is usually caused by confusing between *peer-peer* and *customer-provider* relationships).

Algorithm	Partial Mismatch	Full Mismatch	Policy Mismatch
CR	1463	93	1649
BPP	1526	104	1734
SARK	2329	192	2713
Gao	1610	109	1828

Table 3.2: Experiments results Route-Views vs. IRR

While the experiment presented above is performed over real data, its validation using the IRR may lead to biased validation results. In particular, as mentioned in Chapter 1, in some cases entries in the IRR may be invalid and contain data that is out-of-date or not updated. Second, as pointed out in 2, in some cases the interpretation of the export policy may be ambiguous. For that reason we perform simulations, examining the different algorithms over several random graphs. This process consists of the following steps. First, we generated a random graph and assign a type of relationship to its edges. Then, we simulate the gathering process of BGP routing tables, by modeling each BGP routing table as a policy-based shortest path tree (for more details see Section). Finally, we execute the different algorithms and measure their inaccuracy, i.e. in how many edges, each algorithm failed to assign the correct type of relationship.

We consider several random graphs; each of them consists of 22,000 vertices (similar to the number of ASes observed by the Route-Views database). As was suggested in 2 about third of the links in the AS graph are of type *customer-provider* while almost all the rest of the links are of type *peer-peer*. In the first graph we divide the set of nodes of the graph into five hierarchical groups, where the number of ASes in each group increases exponentially. Thus, the set of nodes of each graph is divided in the following way: 10 ASes are in level 1, 140 ASes are in level 2, 1350 ASes are in level 3, 3500 ASes are in level 4, and about 17000 ASes are in level 5. According to this hierarchy, ASes from the same groups are connected by a *peer-peer* relationship while ASes from different groups are connected by a *customer-provider* relationship. In order to guarantee the reachability of the nodes in the graph with respect to the policy enforced[10], we build the graph such that each AS has, on the average,

[9]This is exactly (Partial Mismatch $+ 2 \cdot$ Full Mismatch).

[10]Recall that when routing policy is considered, connectivity does not necessarily mean reachability.

two providers. We call this graph *layered graph*. The second graph follows the observation that the vertex degree distribution of the *customer-provider* subgraph follows the power-law while the *peer-peer* vertex degree distribution does not (see Chapter 2). Thus, in this graph the *customer-provider* subgraph is modeled by a Barabasi-Albert graph [10, 8] while the *peer-peer* subgraph is modeled by a random graph. We call this graph *Barabasi-Albert graph*. The ten top providers in both graph are fully connected by a set of 45 *peer-peer* links.

Algorithm	Partial Mismatch	Full Mismatch	Policy Mismatch
CR	221	0	221
BPP	1808	44	1896
SARK	5590	35	5660
Gao	417	1	419

Table 3.3: Simulation results of the Layered graph

Algorithm	Partial Mismatch	Full Mismatch	Policy Mismatch
CR	304	7	318
BPP	730	22	774
SARK	9980	806	11592
Gao	833	26	885

Table 3.4: Simulation results of the Barabasi-Albert graph

Tables 3.3 and 3.4 presents the number of partial and full mismatch edges of each graph and each algorithm. It also presents the number of cases in which the export policy derived by the algorithm is different from the actual policy. Similar to the experiments over the Route-Views database, in both graphs the results obtain by our algorithm are much better compared to other algorithms. These results stand out mainly when the number of full mismatch is considered. Likewise, the solutions obtained by our algorithm is the only one that preserve the hierarchical acyclic structure of the graph.

3.6 Summary

In this chapter we studied the Type of Relationship problem. We observed that the conventional definition of the problem does not consider the hierarchical acyclic structure of the AS connectivity map. Base on this observation we defined a new problem, the Acyclic Type of Relationship problem, that takes into account this hierarchical structure. We proved that determining if there is a solution without invalid paths can be solved in a polynomial time, and presented an efficient algorithm for this case. We also presented a $\frac{2}{3}$ approximation algorithm for a variant of the problem, considering the total number of valleys. Our experiments and simulation results show that our algorithms classify the type of relationship between ASes much better than all previous approaches.

Chapter 4

Decreasing Path Length in BGP Using Intermediate Routing

In this chapter we consider an intermediate routing scheme, in which routing data between clients can be done via intermediate nodes. This is done in order to overcome the path inflation problem in the Internet and eventually to reduce the delay between network clients. In particular, we enable clients to communicate via shortest paths despite the policy enforced by ASes. we study this problem as an optimization problem in which the objective target is to enable routing through shortest paths with a minimum number of relay servers. We show that this problem is NP-hard, present a $O(\log(n))$ approximation algorithm for it, and show that this is the best approximation one can get. We examine the practical aspects of the scheme by evaluating the gain one can get over real up-to-date data reflecting the current BGP routing policy in the Internet. We show that a relative small number of relay servers are sufficient to enable routing over the shortest paths between almost all considered nodes, which reduces the average path length of these inflated paths by 40%.

4.1 Model and Problem Definition

Given a set of valid routing paths P and an AS connectivity graph G, denote by $d(v,u)$ the length of the shortest physical path[1] between v and u, and by $d_v(v,u)$ the length of the **valid** path between v and u[2] (if there is no valid path between v and u then $d_v(v,u) = \infty$). An intermediate path with respect to a subset of the vertices U, is a concatenation of valid paths where the end points of each paths, excluding the first and the last nodes, are in U. We denote by $d^U(v,u)$ the length of the shortest intermediate path between v and u, where U is the set of intermediate vertices. The Intermediate Shortest Path Routing (*ISPR*) problem is defined as follows:

Definition 4.1.1 *Given a graph $G = (V,E)$, a set of pairs $Q = \{(s_1,t_1),(s_2,t_2),...,(s_n,t_n)\}$ such that $Q \subseteq V \times V$, and a set of valid paths P, find a subset of vertices $U \subseteq V$ such that*

[1] The term *physical path* refers to any valid or invalid path.

[2] For the sake of simplicity, if a pair of ASes have more than one valid path between them, we consider the shortest valid path. Nevertheless, these parallel paths can be integrated in the model by representing the pair of ASes by several pairs of nodes, such that every pair of node has a unique valid path.

$\forall 1 \leq i \leq n, d^U(s_i, t_i) = d(s_i, t_i)$. *In other words* $\forall 1 \leq i \leq n, \exists \{v_i^1, v_i^2, ...v_i^{m_i}\} \subseteq U$ *such that* $d(s_i, t_i) = d_v(s_i, v_i^1) + d_v(v_i^1, v_i^2) + ... + d_v(v_i^{m_i-1}, v_i^{m_i}) + d_v(v_i^{m_i}, t_i)$. *In this case the set of intermediate vertices* U *is called a feasible solution.*

Intuitively speaking, the relay servers located in the set of intermediate vertices U can be used to route packets from the sources to the destinations along the shortest physical paths by concatenating several valid paths. Note, that in this work we do not deal with the the mechanism and the implementation aspects required by the clients and the intermediate servers in order to utilize the system. This, may be subject for further research. Henceforward, we use the reasonable assumption (that is supported by a real data) that single hop paths are always valid. Otherwise, a shortest path may not be achievable and in this case a feasible solution may not exist. This assumption can be relaxed by modifying the definition of the problem such that we seek to find the shortest available intermediate path of each pair. Note, that this modification does not affect the theoretical analyses of the problem presented in this paper.

Assuming that single hop paths are always valid, $U = V$ is a trivial feasible solution to the *ISPR* problem. Our objective, as discussed in Chapter 1, is to minimize the cardinality of the solution (i.e., to minimize $|U|$). Thus, the *MIN-ISPR* problem is defined as follows:

Definition 4.1.2 *Given an instance of the* ISPR *problem, denote by* F *the set of feasible solutions. Find a solution* U_{opt} *such that* $U_{opt} = \arg\min_{U \in F} |U|$ *(*U_{opt} *is called optimal solution).*

For example, consider the instance of the *ISPR* depicted in Figure 4.1, while the length of the shortest physical path between AS6 and AS4 is two and the length of the shortest physical path between AS10 and AS3 is five (i.e., $d(AS6, AS4) = 2$ and $d(AS10, AS3) = 5$), the lengths of the shortest valid paths are five and infinity respectively (i.e., $d_v(AS6, AS4) = 5$ and $d_v(AS10, AS3) = \infty$). By deploying two intermediate servers in AS7 and in AS8 one can route packets from AS6 to AS4 using the valid paths (AS6, AS7) and (AS7, AS4) and form AS10 to AS3 using the valid paths (AS10, AS8), (AS8, AS6, AS7) and (AS7, AS4, AS3). The total lengths of the paths in this case are two and five respectively. One can see that every other solution requires at least two intermediate nodes (e.g., AS7 and AS5 is another possible solution). Thus, $U = \{AS8, AS7\}$ is an optimal solution and $|U| = 2$.

The decision version of the problem, *k-ISPR*, is defined as follows:

Definition 4.1.3 *Given an instance of the* ISPR *problem and a positive number* k*, test if there is a feasible solution* U *such that* $|U| \leq k$.

While in Chapter 1 we have described several aspects of BGP policy guidelines and explained how they cause a path inflation, the actual policy is not part of the problem definition. In particular, an instance to the ISPR problem consists of a set of valid paths rather than a specific policy. This approach has the following reasons: First, this kind of definition is more general, since given any policy it is easy to find the set of routing paths derived by this specific policy (for example, by emulating the BGP protocol). Thus, this definition holds for any kind of BGP policy. Second, usually ASes do not unveil their local policy scheme and therefore the routing paths derived by this policy cannot be obtained. On the other hand,

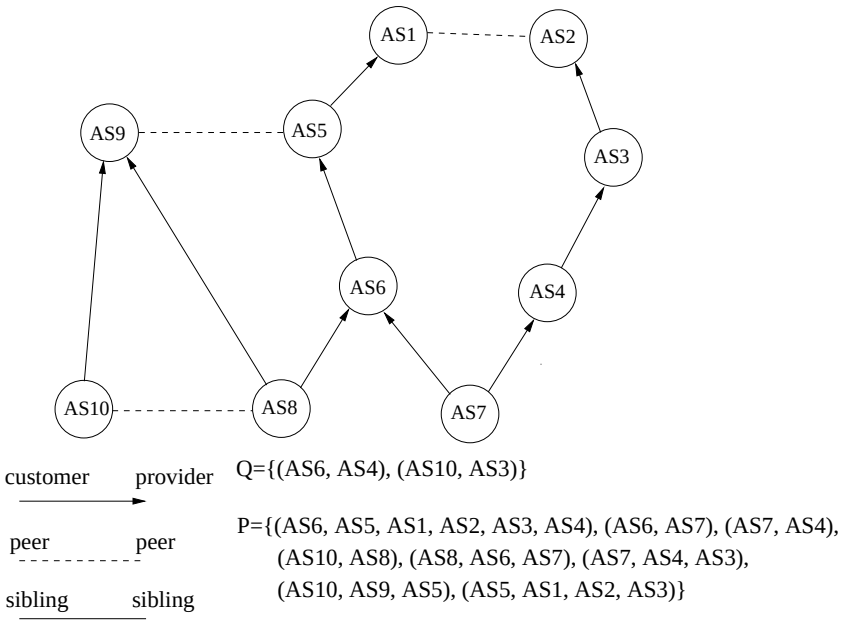


Figure 4.1: Intermediate routing example

there are several ways to obtain valid BGP routing paths (e.g., database containing BGP routing tables, traceroute iterations, etc.). Thus, we can examine the algorithm over real data where the actual BGP policy is unknown. Moreover, the distance metric is not part of the problem definition as well; one may consider any metric such as hop count, delay, etc., according to his (or her) requirements and objectives. Thus, the problem definition is not limited to the AS connectivity map, and it can be used to address different types of network architectures, in which the default routing strategy does not meet the routing demands that are required by a specific application.

4.2 On the Complexity of the *ISPR* Problem

In this section we study the complexity of the *ISPR* problem. In particular, we show that the *k-ISPR* problems NP-hard, we present a $O(\log(n))$ approximation algorithm where n is the number of vertices, and show that this is the best algorithm one can get.

Theorem 4.2.1

I) *The* k-ISPR *problem is NP-hard.*

II) *The* MIN-ISPR *problem cannot be approximate within a factor of* $(1-\epsilon)\cdot\ln(n)$ *for any* $\epsilon > 0$ *unless* $NP \subseteq DTIME(n^{\log\log n})$.

Proof We prove the theorem using a reduction from the Set Cover (SC) problem [28]. First we show that the reduction holds and then we show that it is an approximation preserving reduction. Thus, the *ISPR* problem is hard as the minimization version of Set Cover, and the later, cannot be approximate within a factor of $(1-\epsilon)\cdot\ln(n)$ for any $\epsilon > 0$ [23]. In the

SC problem, given a finite set S, a positive number k, and a set of subsets C of S, one should test if there is a set of subsets, $C' \subseteq C$ such that every element in S appears at least in one set of C' and $|C'| \leq k$ (without loss of generality, we consider an instances to the Set Cover problem, such that for each element in S there is at least one subset in C that contains this element).

Given an instance of the SC problem, i.e., a finite set S, a positive number k, and a set of subsets C of S, we construct an instance to the *k-ISPR* problem as follows. For each element s_i in the set S we match a vertex v_{s_i}, and for each subset $C_j \in C$ we match a vertex v_{C_j}. We also add another vertex v_t, so we have, $V = \{v_{s_i} | s_i \in S\} \cup \{v_{C_i} | C_i \in C\} \cup \{v_t\}$. For each subset $C_i \in C$ we add an edge between v_{C_i} and v_{s_j} if and only if $s_j \in C_i$, and an edge between v_{C_i} and v_t. Namely, $E = \{(v_{C_i}, v_{s_j}) | s_j \in C_i, C_i \in C\} \cup \{(v_{C_i}, v_t) | C_i \in C\}$. The set of paths contains all the possible path of length one, i.e. $P = \{(v, u) | (v, u) \in E\}$. The set of pairs with respect to the *k-ISPR* problem is $Q = \{(v_{s_i}, v_t) | \forall s_i \in S\}$. For example, Figure 4.2 depicts an instance of the *k-ISPR* problem corresponding to the following SC instance: $S = \{1, 2, 3, 4, 5, 6, 7, 8\}$, and $C = \{(1, 2, 3, 4), (2, 5, 8), (4, 6, 7)\}$.

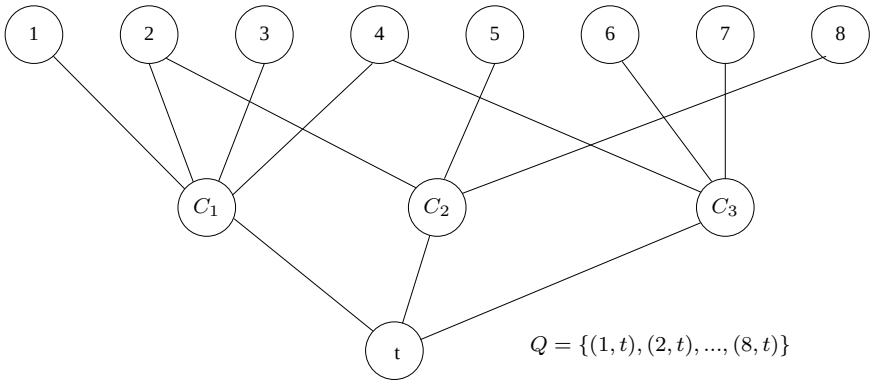


Figure 4.2: Example: Set Cover - ISPR Reduction

According to the construction described above, vertices of type v_{s_i} and the vertex v_t are connected only to vertices of type v_{C_j}. Thus, for each vertex v_{s_i}, any simple physical path between v_{s_i} and v_t is of type (v_{s_i}, v_{C_j}, v_t). Moreover, these paths are not in P. Thus, for each pair $(v_{s_i}, v_t) \in Q, d(s_i, t) = 2$ and $d_v(s_i, t) = \infty$ (we use the fact that every element is covered by at least one subset, thus, there is at least one physical path between any vertex of type v_{s_i} and v_t). Moreover, the path (v_{s_i}, v_{C_j}, v_t) exists if and only if $s_i \in C_j$. Now we show that the reduction holds, namely there is a subset $C' \subseteq C$ such that $|C'| \leq k$ and C' covers S if and only if there is a solution to the corresponding *k-ISPR* instance.

Given a solution $C' = \{C_1, C_2, ..., C_{k'}\}, k' \leq k$ to a SC instance, we show that $U = \{v_{C_1}, v_{C_2}, ..., v_{C_j}\}$ is a solution to the corresponding *k-ISPR* instance. For each pair $(v_{s_i}, v_t) \in Q$, s_i is covered by C', namely $\exists C_j, C_j \in C', s_i \in C_j$. Since $v_{C_j} \in U$ then v_{s_i} can reach t using the following valid paths $(v_{s_i}, v_{C_j}), (v_{C_j}, v_t)$ and $d_v(v_{s_i}, v_{C_j}) + d_v(v_{C_j}, v_t) = 2$ (i.e., $d^U(v_{s_i}, v_t) = d(v_{s_i}, v_t) = 2$). Thus, U is a feasible solution to the *ISPR* instance and $|U| = |C'| \leq k$. On the other hand, let us assume that there is a solution U to the *k-ISPR* instance ($|U| \leq k$). Namely, for each pair $(v_{s_i}, v_t) \in Q$ there is an intermediate node $v_{C_j} \in U$ such that $s_i \in C_j$. Thus, the set $\{C_j | v_{C_j} \in U\}$ covers the set of elements S, and the size of this set is lower or equal to k.

It is left to show that the reduction is an approximation preserving reduction. Given an instance I_{SC} to the minimization version of the SC problem, denote by $OPT_{SC}(I_{SC})$ its optimal solution. Let I_{ISPR} to be the corresponding ISPR instance (i.e., I_{ISPR} is constructed using the recursion described above), and denote by $OPT_{ISPR}(I_{ISPR})$ its optimal solution. As showed above, any solution $\{C_1, C_2, ..., C_k\}$ to I_{SC} induces a solution $\{v_{C_1}, v_{C_2}, ..., v_{C_k}\}$ to I_{ISPR}. Thus, $|OPT_{SC}(I_{SC})| \geq |OPT_{SC}(I_{ISPR})|$. On the other hand, given a solution U to I_{ISPR}, as showed $C' = \{C_j | v_{C_j} \in U\}$ is a solution to I_{SC}, and $|U| \geq |C'|$. ∎

One can argue that while the reduction presented above holds for a theoretical policy scheme, the problem induced by a practical policy is much simpler. However, considering the practical valley-free policy scheme discussed in Chapter 1, the same reduction remains valid by assigning the *customer-provider* relationships such that vertices of the type v_{C_i} are connected only to providers. In this case, for all $v_{s_i} \in V$ there is no valid path between v_{s_i} and t. Thus, even when restricted to real practical policies, is hard to approximate.

Next, we present a $O(\log(n))$ approximation algorithm to the *MIN-ISPR* problem. This algorithm starts with an empty set of intermediate nodes and adds vertices to the set gradually in a greedy fashion. Given an instance to the *ISPR* problem, G, P, Q; a pair of vertices $(s,t) \in Q$; and a set of intermediate nodes U (note that U is not necessarily a feasible solution with respect to the *ISPR* instance), denote by $C^U(s,t)$ the minimum number of intermediate nodes one should add to U in order have the shortest intermediate path between s and t. Formally, $C^U(s,t) = min_{U' \subseteq V, d^{U \cup U'}(s,t) = d(s,t)} |U'|$. Note that if U is a feasible solution then $\forall (s,t) \in Q, C^U(s,t) = 0$. Clearly, adding vertices to U cannot increase the cost $C^U(s,t)$, thus,

$$\forall U, U' \subseteq V, C^U(s,t) \geq C^{U \cup U'}(s,t). \tag{4.1}$$

Moreover, adding one vertex may reduce the cost by at most one. In other words,

$$\forall U \subseteq V \text{ and } \forall v \in V, C^U(s,t) \leq C^{U \cup \{v\}}(s,t) + 1. \tag{4.2}$$

Recall that V is a feasible solution, thus if $C^U(s,t) > 0$ then $\exists v \in V$ such that $C^U(s,t) = C^{U \cup \{v\}}(s,t) + 1$, namely, there is at least one vertex that reduces the cost for each pair of vertices as long as the cost is positive. We use the notation C^U to denote the total cost induced by all pairs, namely $C^U = \sum_i C^U(s_i, t_i)$.

Algorithm *ISPR* $(G = (V,E), Q)$

1. $U_1 = \phi$
2. $i = 1$
3. while $C^{U_i} > 0$
4. $\quad v = \arg\min_{u \in V \setminus U_i} C^{U_i \cup \{u\}}$
5. $\quad U_{i+1} = U_i \cup \{u\}$
6. $\quad i = i + 1$
7. return U_i.

As discussed above, when $C^U = 0$ the set of vertices U is a feasible solution. Thus, the algorithm returns a feasible solution[3]. At each iteration the algorithm selects a vertex that brings the current set of vertices closer to a feasible solution. In particular, this vertex reduces the cost more than any other vertex (finding this vertex is not a trivial task and this procedure will be discussed later).

Theorem 4.2.2 *Given an instance of the* ISPR *problem, denote by U_{opt} an optimal solution (i.e., U_{opt} is a solution to the corresponding* MIN-ISPR *problem), and denote by U the solution returns by the Algorithm* ISPR*. Algorithm* ISPR *is $O(\log(|V|))$ approximation algorithm to the* MIN-ISPR *problem, namely, $|U| \leq |U_{opt}| \cdot O(\log(|V|))$*

Proof In the i'th iteration the current set of intermediate nodes is U_i and it induces a cost of C^{U_i}. Adding the optimal solution U_{opt} to U_i induces a feasible solution (i.e., $C^{U_i \cup U_{opt}} = 0$). Thus, on average every vertex in the optimal solution reduces the cost by $\frac{C^{U_i}}{|U_{opt}|}$. Denote by B_i the cost that is reduced in the i'th iteration, namely $B_i = C^{U_i} - C^{U_{i+1}}$. Since B_i is the maximum cost reduced in iteration i, then

$$B_i \geq \frac{C^{U_i}}{|U_{opt}|} \tag{4.3}$$

therefore,

$$1 \leq \frac{B_i \cdot |U_{opt}|}{C^{U_i}}. \tag{4.4}$$

Without loss of generality, assume that the algorithm runs k iteration, thus, the feasible solution that the algorithm returns is U_k and $|U_k| = k$ (since every iteration one vertex is

[3]We use the fact that in this case V is a feasible solution and each iteration one vertex is added, thus in the worst case, the algorithm returns V.

added). According to Equation 4.12 and since $C^{U_{i+1}} = C^{U_i} - B_i$ we have:

$$|U_k| = k = \sum_{i=1}^{k} 1 \leq \sum_{i=1}^{k} \frac{B_i \cdot |U_{opt}|}{C^{U_i}} =$$

$$= |U_{opt}| \cdot \sum_{i=1}^{k} \underbrace{\left(\frac{1}{C^{U_i}} + \frac{1}{C^{U_i}} + ... + \frac{1}{C^{U_i}} \right)}_{\times B_i} \leq$$

$$\leq |U_{opt}| \cdot \sum_{i=1}^{k} \left(\frac{1}{C^{U_i}} + \frac{1}{C^{U_i} - 1} + ... + \frac{1}{C^{U_i} - B_i + 1} \right) =$$

$$= |U_{opt}| \cdot \left(\frac{1}{C^{U_1}} + \frac{1}{C^{U_1} - 1} + ... + \frac{1}{C^{U_1} - B_1 + 1} + \right.$$

$$+ \frac{1}{C^{U_2}} + \frac{1}{C^{U_2} - 1} + ... + \frac{1}{C^{U_2} - B_2 + 1} + ... +$$

$$\left. + \frac{1}{C^{U_k}} + \frac{1}{C^{U_k} - 1} + ... + \frac{1}{C^{U_k} - B_k + 1} \right) =$$

$$= |U_{opt}| \cdot \left(\frac{1}{C^{U_1}} + \frac{1}{C^{U_1} - 1} + \frac{1}{C^{U_1} - 2} + ... + \frac{1}{1} \right) =$$

$$= |U_{opt}| \cdot O\left(\log\left(C^{U_1} \right) \right) \quad (4.5)$$

The initial cost, $C^{U_1}(s,t)$, is bounded by the distance between s and t, which is at most $|V|$, namely $C^{U_1}(s,t) \leq d(s,t) \leq |V|$. Therefore, $C^{U_1} \leq |Q| \cdot |V| \leq |V|^3$, and we get that,

$$|U_k| = |U_{opt}| \cdot O(\log(|V|^3)) = |U_{opt}| \cdot O(\log(|V|)). \quad (4.6)$$

■

Next, we explain how to find the cost between a pair of vertices (s,t). This procedure should find a shortest physical path between s and t that traverses a minimum number of valid paths. The number of valid paths that are traversed (ignoring vertices that have already added to the set of intermediate nodes), is equal to the cost between s and t. Denote by $G_U = (V_U, A_U)$ the directed graph imposes by the set of the valid paths P, with respect to the set of intermediate nodes U. In particular, $V_U = V$ (where V is the set of node in the original graph), and $A_U = \{(v,u) | \forall v,u \in V_U, v \neq u\}$. Thus, G_U is a complete directed graph. We also assign a distance function w to the edges as follows:

$$w(v,u) = \begin{cases} d_v(v,u) + \epsilon & v \notin U \text{ and } v \neq s \\ d_v(v,u) & \text{otherwise} \end{cases} \quad (4.7)$$

Denote by $\tilde{d}((s,t),U)$ the shortest distance between s and t with respect to G_U and the distance function w. Every simple path in G_U contains at most $V-1$ edges, therefore, if we set $\epsilon < \frac{\Delta d}{|V|}$ where Δd is the smallest difference between any shortest paths, then $d(s,t) + \Delta d > \tilde{d}((s,t),U) \geq d(s,t)$. In this case, $C^U(s,t) = \frac{\tilde{d}((s,t),U) - d(s,t)}{\epsilon}$. In order to find a vertex that satisfies Step 4 of the algorithm we should perform this procedure with respect to $U \cup \{v\}$ for each $v \notin U$ and for each pair.

4.2.1 Complexity and Practical Implementation

In the worst case, the algorithm performs $|V|$ iterations (adding one vertex each iteration, up to V vertices), and in every iteration it finds the most beneficial vertex v. As explained above, for this computation one should find for all $v \in V \setminus U$ and for all $(s,t) \in Q$ the cost $C^U(s,t)$ (where U is the current set of intermediate vertices). Thus, if $Q = \{(s,t)|s,t \in V\}$ (i.e., Q contains all possible pairs) one should execute all-pair shortest path algorithm for each $v \in V \setminus U$. An efficient implementation of the Dijkstra algorithm [19] requires $O(|E|)$ steps, so finding all-pair shortest path for each vertex requires $O(|V|^2 \cdot |E|)$ steps. Putting all parts together, we get that the time complexity of the algorithm is $O(|V|^3 \cdot |E|)$. While this complexity is polynomial, in practice the AS connectivity map is very big and therefore the algorithm may be impractical. In particular, the current Internet consists of about 25000 ASes and over 150000 links [16]. Thus, the time complexity of the algorithm is $O(25000^3 \cdot 150000) \cong 2^{60}$. To overcome this problem we should refine the algorithm and reduce its complexity. In particular, we show a more efficient way to find a vertex satisfying Step 4 of the algorithm. Using this improvement, the practical time complexity of each iteration is reduced to $O(|V|^2)$ and the total time complexity of the algorithm is $O(|V|^3)$ which can be accomplished by a typical desktop in a reasonable time.

Denote by B_v^U the benefit imposed by a node v with respect to a set of vertices U, namely,

$$B_v^U = C^U - C^{U \cup \{v\}}. \tag{4.8}$$

Using this notation, the vertex that is selected in Step 4 of the algorithm is the one that maximizes the benefit, i.e., $v = \arg\max_{u \in V \setminus U_i} B_u^{U_i}$. Thus, at each iteration the algorithm should compute the benefit of each vertex and select a vertex with the maximum benefit. We also denote by $B_v^U(s,t)$ the benefit of a node induced by the pair (s,t), namely,

$$B_v^U(s,t) = C^U(s,t) - C^{U \cup \{v\}(s,t)}. \tag{4.9}$$

Note that according equations 4.1 and 4.2, $\forall U, s, t, B_v^U(s,t) \in \{0,1\}$. Using this notation

$$B_v^U = \sum_{(s,t) \in Q} B_v^U(s,t). \tag{4.10}$$

First, we compute $\forall s,t \in V, d(s,t)$ and $C^\phi(s,t)$ as explained above (i.e., executing the Dijkstra algorithm from every node over the graph G_ϕ). This task can be done in $O(|V| \cdot |E|)$ steps. In order to calculate the benefit of each vertex we check if adding this vertex to the set of intermediate nodes, reduces the cost with respect to a pair of vertices s,t. The total amount of cost, reduced by this vertex with respect to all possible pairs is the benefit of the vertex. The computation of the benefit can be done using the following procedure:

Subroutine *Benefit*$(G = (V,E), d, C)$

1. for each $v \in V$
2. $\quad B_v^\phi = 0$
3. $\quad$ for each pair $(s,t) \in Q$
4. $\quad\quad$ if $d(s,v) + d(v,t) = d(s,t)$ and $C^\phi(s,v) + C^\phi(v,t) < C^\phi(s,t)$
5. $\quad\quad\quad B_v^\phi = B_v^\phi + 1$

The time complexity of the procedure is $O(|V|^3)$ and its correctness is derived from the following observation:

Observation 4.2.1 *A vertex v reduces the cost between a pair of vertices s,t if and only if v traverse a shortest path between s and t, and if the sum of the cost of each sub-path reduces the cost. In other words, given a set of intermediate nodes $U \subseteq V$ and a pair of vertices $s,t \in V$. For all $v \in V$, $C^{U\cup\{v\}}(s,t) = C^U(s,t) - 1$ (i.e., $B_v^U(s,t) = 1$) if and only if the following two conditions hold:*

1. $d(s,v) + d(v,t) = d(s,t)$
2. $C^U(s,v) + C^U(v,t) < C^U(s,t)$

At each iteration, a vertex v with the maximal benefit should be selected. This task can be easily done by passing the benefit vector (i.e., the benefit of all vertices) and finding the maximal benefit. The time complexity of this task is $(O(|V|)$. The more complex task is to update the cost and the benefit. These tasks are performed by the following procedure:

Subroutine *Update* $(G = (V,E), Q, d, C, B, U)$

1. $L = \phi$
2. $v = \arg\max_{u \in V \setminus U} B_u^U$
3. $U' = U \cup \{v\}$
4. for each pair $(s,t) \in Q$
5. if $d(s,v) + d(v,t) = d(s,t)$ and $C^U(s,v) + C^U(v,t) < C^U(s,t)$
6. $C^{U'}(s,t) = C^U(s,t) - 1$
7. for each pair $(s,t) \in Q$
8. if $d(s,v) + d(v,t) = d(s,t)$ and $C^U(s,v) + C^U(v,t) \leq C^U(s,t)$
9. $L = L \cup \{(s,t)\}$
10. for each vertex $u \in V \setminus U'$
11. for each pair $(s,t) \in L$
12. if $d(s,u) + d(u,t) = d(s,t)$ and $C^U(s,u) + C^U(u,t) < C^U(s,t)$ and $C^{U'}(s,u) + C^{U'}(u,t) = C^{U'}(s,t)$
13. $B_u = B_u - 1$
14. if $d(s,u) + d(u,t) = d(s,t)$ and $C^U(s,u) + C^U(u,t) = C^U(s,t)$ and $C^{U'}(s,u) + C^{U'}(u,t) < C^{U'}(s,t)$
15. $B_u = B_u + 1$

Step 2 finds a vertex with the maximal benefit, and in steps 4-6 the cost is updated by passing over all pairs with a time complexity of $O(|V|^2)$ (the correctness of this cost update is

derived from Observation 4.2.1). A naive approach for updating the benefit is to recalculate it for each vertex. in this case, for each vertex one needs to examine the cost of each pair as described in the *Benefit* procedure. Nevertheless, this computation requires $O(|V|^3)$ steps and using it will bring the total time complexity to $O(|V|^4)$. Thus, in order to reduce the complexity, we update the benefit by examining only subset of the pairs. In particular a pair $(s,t) \in Q$ is examined only if the vertex v (selected in Step 2 of the procedure) traversed a shortest path between s and t, and this shortest path has minimum cost with respect to U (see Lemma 4.2.1). Steps 7-9 find these pairs, and in steps 10-15 the benefit is updated according to Observation 4.2.1.

Lemma 4.2.1 *Given a set of intermediate nodes $U \subseteq V$, a pair of vertices $s,t \in V$ and a vertex $u \in V \setminus U$. For all $u \in V \setminus U \cup \{v\}$ if $B_u^U(s,t) \neq B_u^{U\cup\{v\}}(s,t)$ then the following two conditions hold:*

1. $d(s,v) + d(v,t) = d(s,t)$
2. $C^U(s,v) + C^U(v,t) \leq C^U(s,t)$

Proof Adding one vertex to the set of intermediate node can reduce the cost by at most 1 (see equations 4.1 and 4.2), therefore, for all $U \subseteq V$ and for all $v \in V$, $B_v^U(s,t) \in \{0,1\}$. Therefore, if $B_u^U(s,t) \neq B_u^{U\cup\{v\}}(s,t)$ then

$$B_u^U(s,t) = 1 \text{ and } B_u^{U\cup\{v\}}(s,t) = 0$$

or

$$B_u^U(s,t) = 0 \text{ and } B_u^{U\cup\{v\}}(s,t) = 1.$$

In the first case $B_u^U(s,t) = C^U(s,t) - C^{U\cup\{u\}}(s,t) = 1$ and $B_u^{U\cup\{v\}}(s,t) = C^{U\cup\{v\}}(s,t) - C^{U\cup\{v\}\cup\{u\}}(s,t) = 0$. According to Equation 4.1, $C^{U\cup\{u\}}(s,t) \geq C^{U\cup\{v\}\cup\{u\}}(s,t)$, thus $C^U(s,t) > C^{U\cup\{u\}}(s,t) \geq C^{U\cup\{v\}\cup\{u\}}(s,t) = C^{U\cup\{v\}}(s,t)$. The fact that $C^U(s,t) > C^{U\cup\{v\}}(s,t)$ means that adding v to the subset U reduces the cost. Thus, according to Equation 4.2 and Observation 4.2.1 $d(s,v) + d(v,t) = d(s,t)$ and $C^U(s,v) + C^U(v,t) < C^U(s,t)$.

In the second case $B_u^U(s,t) = C^U(s,t) - C^{U\cup\{u\}}(s,t) = 0$ and $B_u^{U\cup\{v\}}(s,t) = C^{U\cup\{v\}}(s,t) - C^{U\cup\{v\}\cup\{u\}}(s,t) = 1$. According to Equation 4.1 $C^{U\cup\{u\}}(s,t) = C^U(s,t) \geq C^{U\cup\{v\}}(s,t) > C^{U\cup\{v\}\cup\{u\}}(s,t)$, namely, $C^{U\cup\{u\}}(s,t) > C^{U\cup\{v\}\cup\{u\}}(s,t)$. Thus, according to Observation 4.2.1 $d(s,v) + d(v,t) = d(s,t)$. Moreover, the fact that $C^{U\cup\{u\}}(s,t) > C^{U\cup\{v\}\cup\{u\}}(s,t)$ implies that $C^{U\cup\{u\}}(s,v) + C^{U\cup\{u\}}(v,t) < C^{U\cup\{u\}}(s,t)$. It is easy to show that if u traverse a shortest path between s and v it cannot traverse a shortest path between v and t, and vice versa. Thus, according to Observation 4.2.1 if $C^U(s,v) > C^{U\cup\{u\}}(s,v)$ then $C^U(s,v) \leq C^{U\cup\{u\}}(s,v) + 1$ and $C^U(v,t) = C^{U\cup\{u\}}(v,t)$; and if $C^U(v,t) > C^{U\cup\{u\}}(v,t)$ then $C^U(v,t) \leq C^{U\cup\{u\}}(v,t) + 1$ and $C^U(s,v) = C^{U\cup\{u\}}(s,v)$. In both cases $C^U(s,v) + C^U(v,t) \leq C^{U\cup\{u\}}(s,v) + C^{U\cup\{u\}}(v,t) + 1 < C^{U\cup\{u\}}(s,t) \leq C^U(s,t)$. Thus, $C^U(s,v) + C^U(v,t) \leq C^U(s,t)$ ■

While in the worst case the number of pair to be examined in Step 11 of the procedure is $O(|V|^2)$, in the AS connectivity graph the average number of pairs that satisfy the first condition (i.e., $d(s,u) + d(u,t) = d(s,t)$) is $O(|V|)$. Thus, when the algorithm requires $|V|$ iterations, the average time complexity is $O(|V|^3)$.

4.3 Experiment Results

In this section we examine the algorithms presented above over real data gathered from the Route-Views project. We use data from May 2007 that consists of 25463 ASes, 57276 links and over 1,980,000 paths. Similar to [27], we have found that about 20% of the routing paths are inflated. Since a single BGP routing table contains valid paths only from its source to the entire set of ASes, the Route-Views database covers less than 1% of all possible paths (less than 60 BGP routing tables out of over 25000). In order extend the set of valid paths, and based on the policy guidelines discussed in Chapter 1, we consider also sub-paths as valid paths. For instance, if the path $(AS1, AS2, AS3, AS4)$ is a valid path, the following sub-paths $(AS1, AS2, AS3)$, $(AS2, AS3, AS4)$, $(AS1, AS2)$, $(AS2, AS3)$, $(AS3, AS4)$ are added to the set of valid paths as well. In practice, in some cases the length of a sub-path between pair of ASes, derived from a valid path, may be longer (and rarely shorter) than the real shortest valid path between this pair. Nevertheless, in most cases the length of a sub-path between pair of ASes indicates the actual length of the path. In addition, although BGP paths are directed, we take into account the reverse path. For example, if the path $(AS1, AS2, AS3, AS4)$ is a valid path (or sub-path), we assume that the following path $(AS4, AS3, AS2, AS1)$ is a valid path as well.

We execute the algorithm, over two sets of pairs, considering two different scenarios. In the first case, our goal is to find the minimal number of relay servers needed in order to have a shortest routing path between any pair of Route-Views BGP sources. This scenario consists of 3080 pairs and it can present an organization that have several dozens of branches and it wants to improve the communication between its sites. Thus, it may deploy a set of relay servers that are not necessarily located in its branches, in order to enable routing via shortest paths between its sites. Note that the organization may obtain the set of valid paths from its sources to other ASes by several means such as *traceroute*. Henceforward, we refer this scenario as the VPN scenario.

We have arranged the set of relay servers according to the order they have been found by the algorithm. Note that in This scenario 10% of the paths are inflated. Figures 4.3 and 4.4 depict the number of relay servers required to enable routing via shortest paths between the subset of pairs that have an inflated routing path (i.e., the length of the valid routing path of these pairs is longer than the length of the shortest physical path), and the average path length between these pairs. While 28 relay servers are required to obtain shortest paths between the entire set of pairs, and they reduce the average path length of the inflated paths by 47%, one can achieve almost an optimal performance using less than 10 relay servers (clearly we use the first servers found by the algorithm). In this case, we obtain more than 82% of the shortest paths, reducing the average path length by 36%. In addition, we consider a different deployment in which the relay servers are co-located in the organization site domains (this approach is motivated by the fact that the cost associated with a co-located relay server is considerable lower than the cost of a server located in a different domain). In this case, a set of 56 relay servers reduce the average path length by 11% and only 30% of the shortest paths are obtained.

Figure 4.5 depicts the complementary distribution function of the ratio between the length of available routing paths (induced by the deployment of relay servers), and the length of the shortest physical paths. One can see that using 10 relay servers the paths that are still

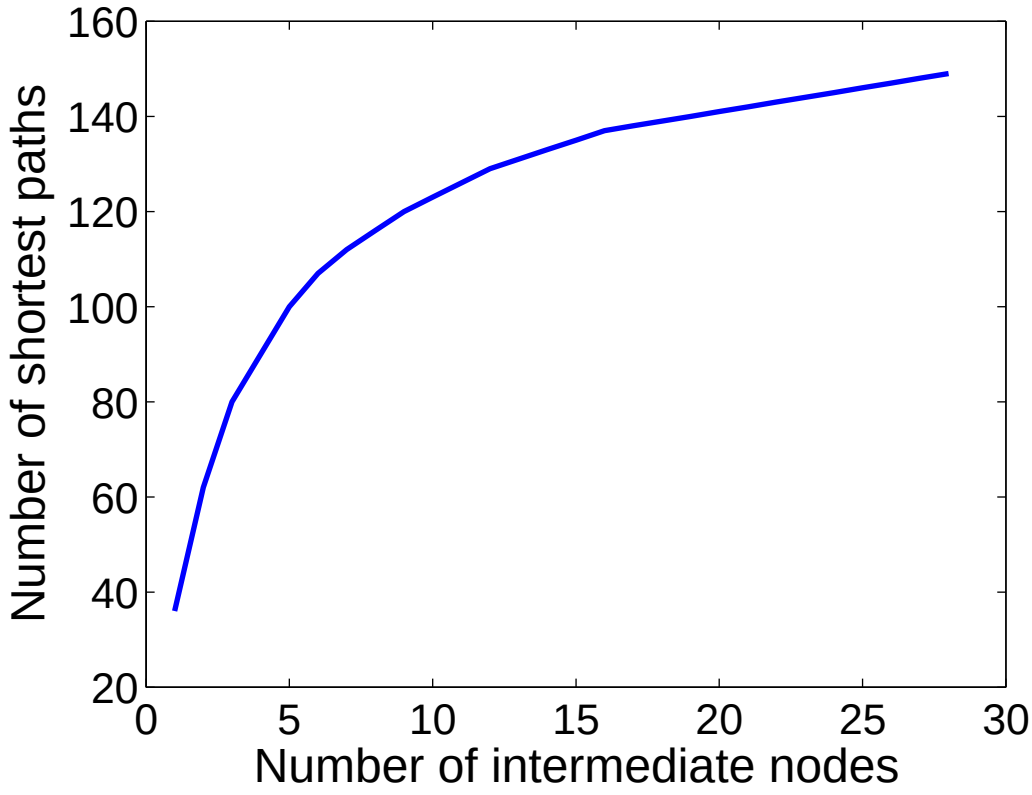


Figure 4.3: Number of new shortest paths, VPN scenario

longer than the shortest path are inflated by less than 50% while without any relay server, there are some paths that are inflated by 200% and many paths are inflated by over than 50%. When the relay servers are co-located in the organization sites, still we have some paths that are inflated by 200%.

In the second scenario we study the set of pairs contains all possible pairs such that the first AS in each pair is a source of a BGP routing table. This scenario, that consists of more than 1,400,000 pairs, can be adopted by an ICP that has a distributed server farm consisting of several dozens of centers located in different ASes. The ICP may deploy relay servers in order to improve the service to its clients, located in all other ASes, namely the objective target of the ICP is to reduce the routing path length between its clients and each one of its servers. In this scenario over than 20% of the paths are inflated. Henceforward, we refer this scenario as the ICP scenario.

Similar to the VPN scenario, figures 4.6 and 4.7 depict the number of relay servers required to enable routing via shortest paths between the subset of pairs that have an inflated routing path, and the average path length between these pairs, in the ICP scenario. In this case, the number of servers requires to enable routing via shortest path between all possible pair is quite big (1895 relay servers), and they reduce the average path length by 39%. Nevertheless, a relative small number of the first 100 relay servers found by the algorithm are sufficient to achieve 81% of the shortest paths, reducing the average path

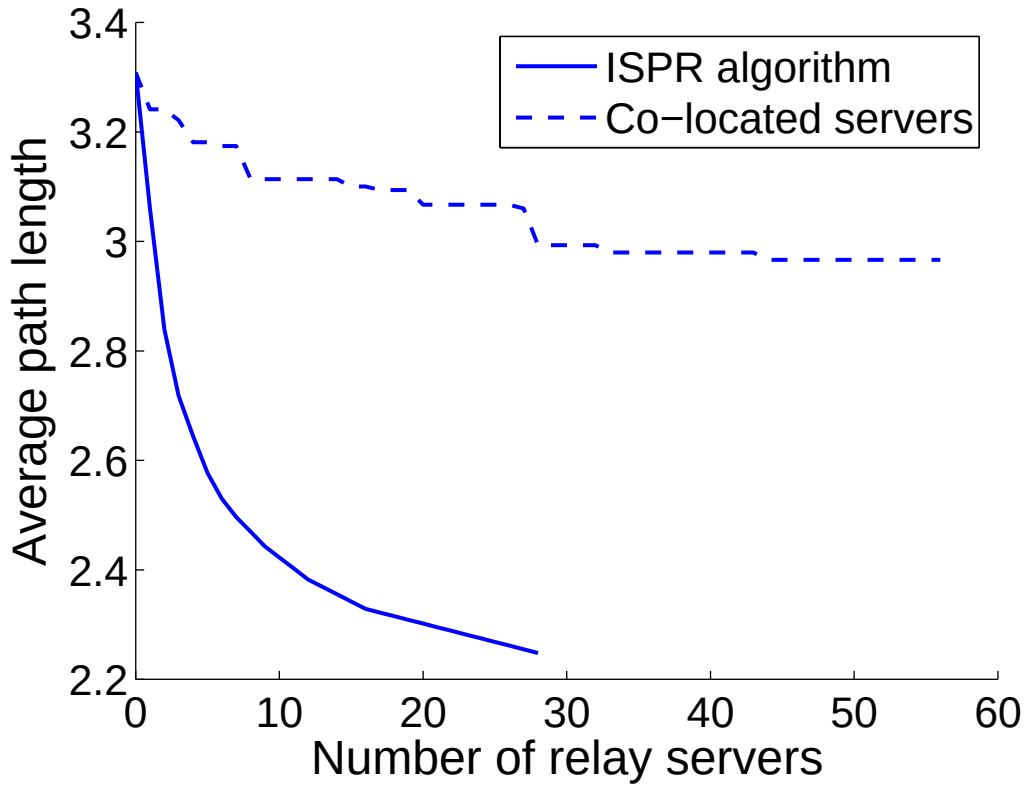


Figure 4.4: Average path length, VPN scenario

length by 30%.

The complementary distribution function of the ratio between the length of available routing paths (induced by the deployment of relay servers), and the length of the shortest physical paths, depicted in Figure 4.8, shows that most of the paths that remain longer than the shortest paths, using 100 relay servers, are inflated by less than 50%, while when no intermediate routing is used, a considerable amount of paths (over 100000) are inflated by over 50% and there are paths that are inflated by 350%.

Discussion: We have found out that there is a strict correlation between the ASes selected by the algorithm in the ICP scenario and the most connected ASes in the AS connectivity graph (i.e., the ASes with the highest degree). In particular, among the first 10 ASes selected by the algorithm, there are the five most connected ASes, and among the first 100 ASes selected by the algorithm, there are the 30 most connected ASes (and many others from the top 100 most connected ASes). Using the observation presented in [24] that the size of an AS is usually proportional to its vertex degree one can derive that the best location for the relay servers is typically in the top providers. This observation may support the assumption discussed in [27] and in [26] that an AS prefer to route packet via its customers rather than its providers or peers. Consider, for example, the AS topology depicts in Figure 4.9. While the length of the shortest path between AS3 and AS6 is three (over the path AS3, AS2, AS1, AS6), if AS2 prefer to route via its customer, then the length of the valid routing path

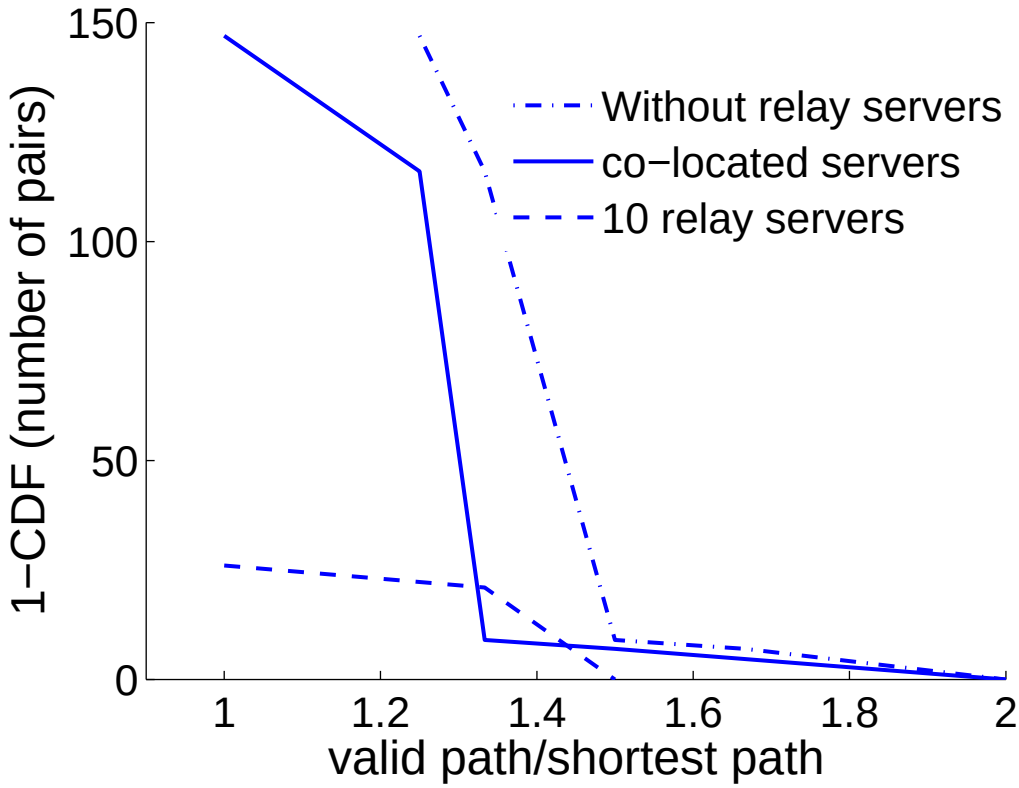


Figure 4.5: (routing path length)/(shortest path length) distribution, VPN scenario

between AS3 and AS6 is four, using the routing path AS3, AS2, AS4, AS5, AS6. Thus, the shortest routing path can be achieved by deploying a relay server in AS1. If AS1 is a top provider with a large vertex degree, this pattern may be repeated with many other pairs. In this case, the overall benefit of AS1 will be significant, and it will be selected by the algorithm.

The results presented in this section show that one can run our algorithm over very large data sets and that the theoretical algorithms can be used to improve the performances of applications in the Internet. The same scheme can also be deployed in other practical cases, where the actual routing paths fall short of providing needed QoS properties needed by actual applications.

4.4 Problem Extensions

In the problem definition described in Section 4.1 we have considered an unweighed version of the problem where each node has equal cost. Thus, minimizing the total cost is equal to minimizing the total number of intermediate servers. Nevertheless, one may consider a weighted version where a cost function is attached to the nodes. In this case, the objective is to minimize the total cost imposed by this cost function. Next, we define the weighed

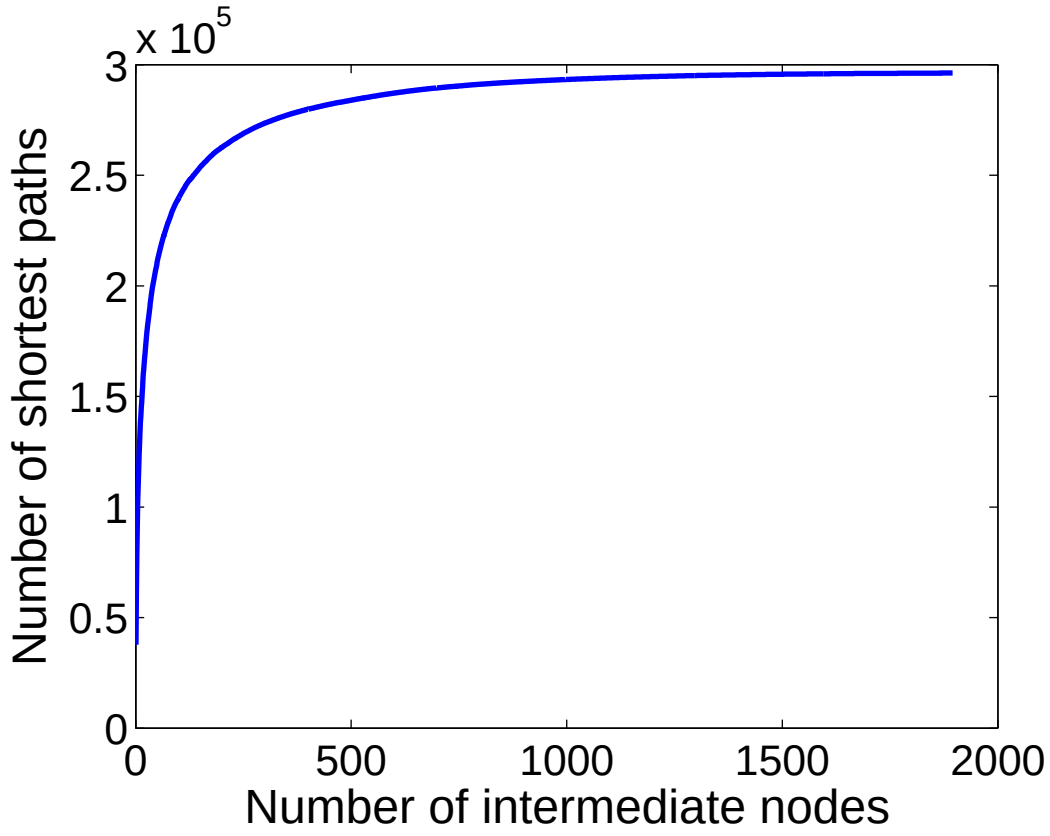


Figure 4.6: Number of new shortest paths, ICP scenario

version of the *ISPR* problem, and present a $O(\log(n))$ approximation algorithm.

The *MIN-W-ISPR* problem is defined as follows:

Definition 4.4.1 *Given an instance of the* ISPR *problem, and a weight function* $W : V \rightarrow R^+$*, denote by* F *the set of feasible solutions. Find a solution* U_{opt} *such that* $U_{opt} = \arg\min_{U \in F} \sum_{u \in U} w(u)$ *(*U_{opt} *is called optimal solution).*

Similar to the algorithm presented in Section 4.2, the algorithm to the *MIN-W-ISPR* starts with an empty set of intermediate nodes and adds vertices to the set gradually in a greedy fashion.

Denote by B_U^v the benefit derived by adding the vertex v to the set of intermediate nodes U. Namely, $B_U^v = C^U - C^{U \cup v}$. According to this definition, every iteration, Algorithm *ISPR* presented in Section 4.2 selects the vertex v that maximize the benefit. In the weighted version of the algorithm (presented in this section) the benefit of a vertex v is normalized according to the weight of the vertex $w(v)$.

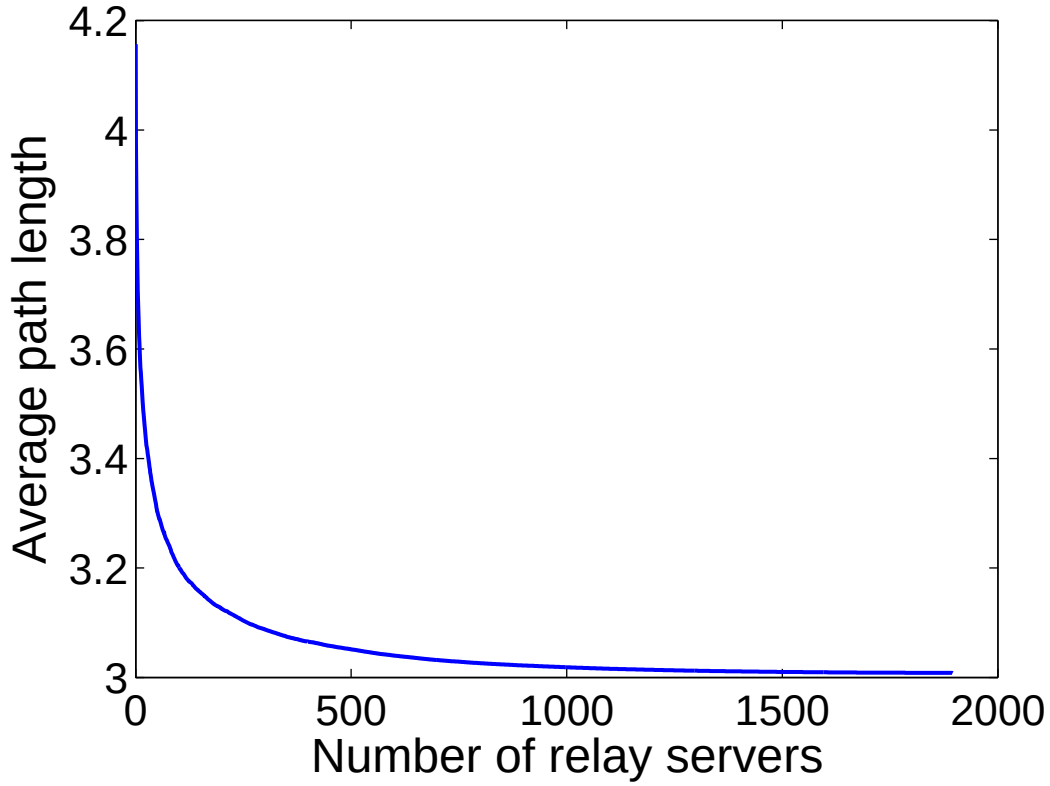


Figure 4.7: Average path length, ICP scenario

Algorithm *W-ISPR*$(G = (V, E), Q, W)$

1. $U_1 = \phi$
2. $i = 1$
3. while $C^{U_i} > 0$
4. $\quad v = \arg\max_{u \in V \setminus U_i} \frac{B^v_{U_i}}{w(v)}$
5. $\quad U_{i+1} = U_i \cup \{u\}$
6. $\quad i = i + 1$
7. return U_i.

Theorem 4.4.1 *Given an instance of the* ISPR *problem, denote by* U_{opt} *an optimal solution (i.e.,* U_{opt} *is a solution to the corresponding* MIN-W-ISPR *problem), and denote by* U *the solution returns by the Algorithm* W-ISPR*. Algorithm* W-ISPR *is* $O(\log(|V|))$ *approximation algorithm to the* MIN-ISPR *problem, namely,* $W(U) \leq W(U_{opt}) \cdot O(\log(|V|))$

Proof In the i'th iteration the current set of intermediate nodes is U_i and it induces a cost of C^{U_i}. Adding the optimal solution U_{opt} to U_i induces a feasible solution (i.e., $C^{U_i \cup U_{opt}} = 0$).

Thus, the average benefit of a vertex in the optimal solution is $\frac{C^{U_i}}{W(U_{opt})}$. Since $\frac{B^v_{U_i}}{w(v)}$ is the maximum normalized benefit in iteration i, then

$$\frac{B^v_{U_i}}{w(v)} \geq \frac{C^{U_i}}{W(U_{opt})} \tag{4.11}$$

therefore,

$$w(v) \leq \frac{B^v_{U_i} \cdot W(U_{opt})}{C^{U_i}}. \tag{4.12}$$

Similar to the analysis presented in Section 4.2.

Without loss of generality, assume that the algorithm runs k iteration, and the vertex v_i is added in the i'th iteration. Thus, Similar to the analysis presented in Section 4.2 we have:

$$\begin{aligned}
W(U_k) = \sum_{i=1}^{k} w(v_i) \leq \sum_{i=1}^{k} \frac{B^v_{U_i} \cdot W(U_{opt})}{C^{U_i}} = \\
= W(U_{opt}) \cdot \sum_{i=1}^{k} \underbrace{\left(\frac{1}{C^{U_i}} + \frac{1}{C^{U_i}} + ... + \frac{1}{C^{U_i}} \right)}_{\times B^v_{U_i}} \leq \\
\leq W(U_{opt}) \cdot \sum_{i=1}^{k} \left(\frac{1}{C^{U_i}} + \frac{1}{C^{U_i}-1} + ... + \frac{1}{C^{U_i}-B^v_{U_i}+1} \right) = \\
= W(U_{opt}) \cdot \left(\frac{1}{C^{U_1}} + \frac{1}{C^{U_1}-1} + ... + \frac{1}{C^{U_1}-B_1+1} + \right. \\
+ \frac{1}{C^{U_2}} + \frac{1}{C^{U_2}-1} + ... + \frac{1}{C^{U_2}-B_2+1} + ... + \\
\left. + \frac{1}{C^{U_k}} + \frac{1}{C^{U_k}-1} + ... + \frac{1}{C^{U_k}-B_k+1} \right) = \\
= W(U_{opt}) \cdot \left(\frac{1}{C^{U_1}} + \frac{1}{C^{U_1}-1} + \frac{1}{C^{U_1}-2} + ... + \frac{1}{1} \right) = \\
= W(U_{opt}) \cdot O\left(\log\left(C^{U_1} \right) \right)
\end{aligned} \tag{4.13}$$

As we show in Section 4.2 $O\left(\log\left(C^{U_1}\right)\right) = O(\log(|V|)$ thus

$$W(U_k) = W(U_{opt}) \cdot O(\log(|V|)). \tag{4.14}$$

■

Another extension of the problem that one can consider is to find an intermediate path which is not necessarily the shortest path but it is short enough. In other words, the objective function is to find a set of intermediate nodes such that the length of the shortest valid path between each pair does not exceed a predefined threshold. In this case, rather than using a shortest path algorithm to compute the cost, one should use an algorithm that finds a restricted shortest path. While the restricted shortest path problem is NP-hard, it has a FPTAS [30, 38]. Thus, computing the cost using an approximation algorithm to the restricted shortest path problem imposes a $O(\log(n))$ approximation algorithm to this extended version of the *ISPR* problem.

4.5 Related Work

While the concept of routing via intermediate nodes has been studied before, as far as we know, our work is the first that analyzes the placement problem of intermediate nodes from inter domain perspective where the objective is to reduce the length of the routing paths with the present of routing policy. In [44] the authors show that in 30%-80% of the cases, there is an alternate path with better quality compare to the default path. In [9] the authors describe a Resilient Overlay Network (RON) that consists of overlay nodes located in a variety of routing domains and are used to forward data on behalf off any pair of nodes. The main purpose of RON is to enable communication between nodes when the underlying Internet path does not exist (for example due to BGP policy) or to improve the communication between nodes using an alternate routing path. In both works the authors do not consider the placement problem but they focus on the quality of these alternate paths [44] and on the design and the implementation of the system [9]. Other works considered an intra-domain routing scheme in which default routing is done along shortest paths. In this case, deflecting the routing paths is usually done in order to improve the throughput and the robustness of the network. In [18] the authors address the computational complexity of the N-hub Shortest Path problem in which each routing path that is off the shortest path, is a concatenation of one or more shortest paths and the objective target is to minimize the maximum load of the network. They show that this problem is NP-hard and does not have a polynomial time approximation scheme (PTAS). They also present simulation results based on a probabilistic approximation algorithm and an on-line algorithm. In [12] the authors study the relay placement problem, in which k relay nodes should be placed in an intra-domain network. An overlay path, in this case, is a path that consists of two shortest paths, one from the source to a relay node and the other from the relay node to the destination. The objective function in this work is to find for each source-destination pair an overlay path that is maximally disjoint from the default shortest path. This problem is motivated by the request to increase the robustness of the network in case of router failure. In [35] the authors introduce a routing strategy, which replaces the shortest path routing, that routes traffic to destination via predetermined intermediate node in order to avoid network congestion under high traffic variability.

4.6 Summary

In this chapter we have considered an intermediate routing scheme to overcome the path inflation problem, typically cause by BGP routing policy, in the Internet. Understanding that the number of relay servers, which are the core of this routing scheme, should be minimized, we have presented an efficient algorithm that determines where to deploy these relay servers. Then, considering couple of practical scenarios, we showed that a relative small number of relay servers can be used to overcome the problem in these cases.

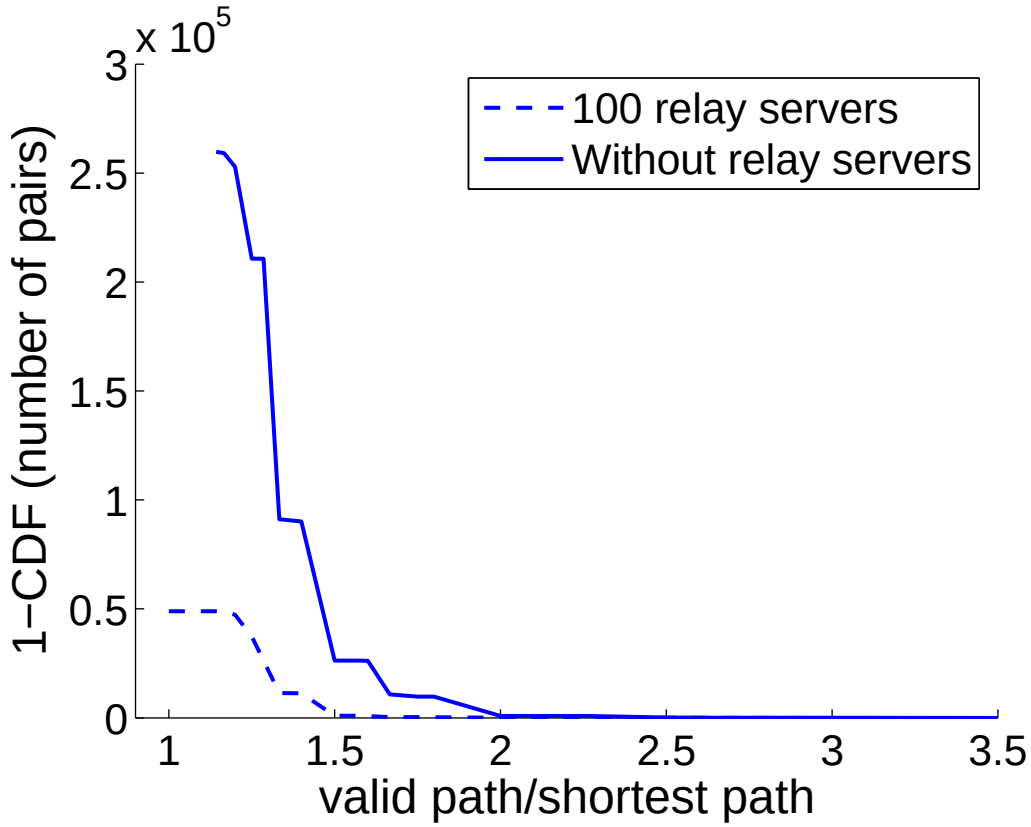


Figure 4.8: (routing path length)/(shortest path length) distribution, ICP scenario

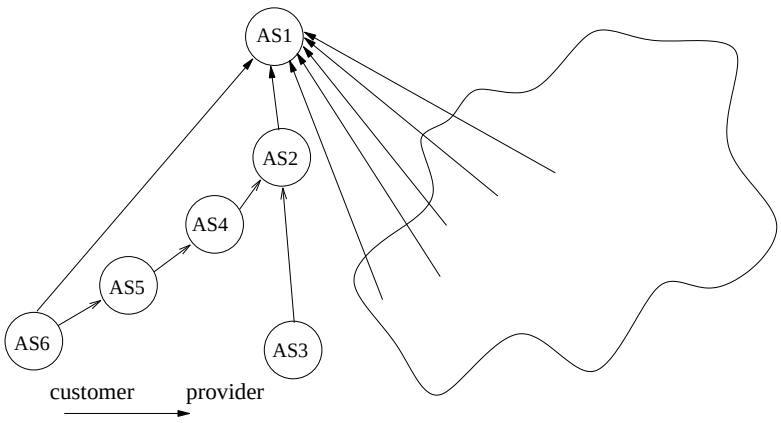


Figure 4.9: Prefer customer routing policy

Chapter 5

Discussion

In this work we studied several problems and addressed some open questions related to the topological structure of the Internet in the AS level. In particular we concentrate on problems arising from the practical routing policy that is used by BGP.

We showed that the difficulty to measure the size and the structure of the Internet using a set of BGP routing tables is derived from the routing policy, and indeed at least 35% of the links are still missing from all known databases where less conservative estimations indicate that more than 50% of the link remain hidden. By understanding the gathering process of databases such Route-Views and IRR we showed that almost all missing links are of type *peer-peer* while a considerable amount of *customer-provider* links are revealed. Thus, trying to disclose the full AS connectivity graph by an increasing set of BGP routing table or by a set of agents performing periodic *traceroute* (both discover mostly *customer-provider* links) may be insufficient in order to fully unveil the *peer-peer* subgraph. A better understanding and modeling the structure of these unveiled *peer-peer* links and their location in the hierarchical structure is a subject to future work.

We also studied the vertex degree distribution of the AS connectivity graph and showed that the distribution of the *peer-peer* subgraph is considerately different from the one of the *customer-provider* subgraph. These inferences, may lead to new models describing the AS connectivity map that consist of two separate models. One describing the *peer-peer* subgraph and another describing the *customer-provider* subgraph. In particular, these models should take into account our finding regarding the vertex degree distribution of each subgraph. Namely, the vertex degree distribution of the *customer-provider* subgraph follows the power-law and the the vertex degree distribution of the *peer-peer* subgraph is similar to the distribution of a Waxman graph.

Then we studied the Type of Relationship problem that deals with the hierarchical structure of the Internet as derived from the ASes policy. We observed that the conventional definition of the problem does not consider the hierarchical acyclic structure of the AS connectivity map. Base on this observation we defined a new problem, the Acyclic Type of Relationship problem, that takes into account this hierarchical structure. We proved that determining if there is a solution without invalid paths can be solved in a polynomial time, and presented an efficient algorithm for this case. We also presented a $\frac{2}{3}$ approximation algorithm for a variant of the problem, considering the total number of valleys. Our experiments and simulation results show that our algorithms classify the type of relationship between

ASes much better than all previous approaches.

Finally we have considered an intermediate routing scheme to overcome the path inflation problem, typically cause the routing policy, in the Internet. Understanding that the number of relay servers, which are the core of this routing scheme, should be minimized, we have presented an efficient algorithm that determines where to deploy these relay servers. Then, considering couple of practical scenarios, we showed that a relative small number of relay servers can be used to overcome the problem in these cases.

While understanding the topological and the hierarchical structure of the Internet is far from being complete, and a lot of research in this area should be done, this thesis sheds light on this subject. We hope that the insights presented in this thesis will help to utilize the Internet in more efficient way by developing and improving protocols and algorithms.

Bibliography

[1] Internet routing registry. http://www.irr.net.

[2] National laboratory for applied network research (nlanr). http://www.nlanr.net/.

[3] Reseaux ip europeenne (ripe). http://www.ripe.net.

[4] University of oregon route views projects. http://www.routeviews.org.

[5] Internet protocol. *RFC 791*, 1981.

[6] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M. Terpstra. Routing policy specification language (rpsl). *RFC 2622*, 1999.

[7] Cengiz Alaettinoglu. Scalable router configuration for the internet. In *Proc. of IEEE IC3N*, October 1996.

[8] Reka Albert and Albert-Laszlo Barabasi. Topology of evolving networks: Local events and universality. *Physical Review Letters*, 85(24):5234–5237, December 2000.

[9] David G. Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. In *18th ACM SOSP*, Banff, Canada, October 2001.

[10] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286:509–512, October 1999.

[11] Giuseppe Di Battista, Thomas Erlebach, Alexander Hall, Maurizio Patrignani, Maurizio Pizzonia, and Thomas Schank. Computing the types of the relationships between autonomous systems. *IEEE/ACM Transactions on Networking*, 2007. To appear.

[12] Meeyoung Cha, Sue Moon, Chong-Dae Park, and Aman Shaikh. Placing Relay Nodes for Intra-Domain Path Diversity. In *Proc. IEEE INFOCOM*, April 2006.

[13] H. Chang, S. Jamin, and W. Willinger. Internet connectivity at the as-level: An optimization-driven modeling approach. In *ACM SIGCOMM Workshop on MoMeTools*, 2003.

[14] Hyunseok Chang, Ramesh Govindan, Sugih Jamin, Scott J. Shenker, and Walter Willinger. Towards capturing representative as-level internet topologies. *Computer Networks Journal*, 44(6):737–755, April 2004.

[15] Benny Chor and Madhu Sudan. A geometric approach to betweenness. In *Third Annual European Symposium on Algorithms*, pages 227–237, September 1995.

[16] Rami Cohen and Danny Raz. The internet dark matter – on the missing links in the as connectivity map. *To Appear in IEEE INFOCOM*, 2006.

[17] Rami Cohen and Danny Raz. Acyclic type of relationships between autonomous systems. 2007.

[18] Reuven Cohen and Gabi Nakibly. On the computational complexity and effectiveness of n-hub shortest path routing. In *Proceedings of the IEEE INFOCOM*, Honk Kong, March 2004.

[19] Edsger. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[20] Xenofontas Dimitropoulos, Dmitri Krioukov, Marina Fomenkov, Bradley Huffaker, Young Hyun, kc claffy, and George Riley. As relationships: Inference and validation. *ArXiv Computer Science e-prints*, April 2006.

[21] Guy Even, Joseph Naor, Baruch Schieber, and Madhu Sudan. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica*, 20(2):151–174, 1998.

[22] C. Faloutsos, P. Faloutsos, and M. Faloutsos. On power-law relationships of the internet topology. In *Proc. of ACM SIGCOMM*, pages 251–262, September 1999.

[23] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[24] Lixin Gao. On inferring autonomous system relationships in the internet. *ACM/IEEE Transactions on Networking*, 9(6):733–745, December 2001.

[25] Lixin Gao, Tim Griffin, and Jennifer Rexford. Inherently safe backup routing with bgp. In *Proc. of IEEE INFOCOM*, 2001.

[26] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *ACM/IEEE Transactions on Networking*, 9(6):681–692, December 2001.

[27] Lixin Gao and Feng Wang. The extent of as path inflation by routing policies. In *IEEE Global Internet Symposium*, 2002.

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

[29] Ramesh Govindan and Anoop Reddy. An analysis of internet inter-domain topology and route stability. In *Proc. of IEEE INFOCOM*, 1997.

[30] Refael Hassin. Approximation schemes for the restricted shortest path problem. *Mathematics of Operations Research*, 17(1):36–42, 1992.

[31] Geoff Huston. Interconnection, peering and settlement. In *Proc. of INET*, June 1999.

[32] Telecommunication Standardization Sector Of ITU. ITU-T Recommendation G.114. Technical report, International Telecommunication Union, March 1993.

[33] Viggo Kann. *On the Approximability of NP-complete Optimization Problems*. Ph.d. thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, 1992.

[34] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103, 1972.

[35] Murali Kodialam, T. V. Lakshman, and Sudipta Sengupta. Efficient and Robust Routing of Highly Variable Traffic. In *HotNets III*, 2004.

[36] Sven Kosub, Moritz G. Maaß, and Hanjo Täubig. Acyclic type-of-relationship problems on the internet. In *Proceedings of the 3rd Workshop on Combinatorial and Algorithmic Aspects of Networking (CAAN'2006)*, pages 98–111, 2006.

[37] A. Lakhina, J. W. Byers, M. Crovella, and P. Xie. Sampling biases in ip topology measurments. In *Proc. of IEEE INFOCOM*, 2003.

[38] Dean H. Lorenz and Danny Raz. A simple efficient approximation scheme for the restricted shortest path problem. *Operational Research Letters*, 28(5):213–219, 2001.

[39] D. Meyer, J. Schmitz, C. Orange, M. Prior, and C. Alaettinoglu. Using rpsl in practice. *RFC 2650*, 1999.

[40] J. Moy. Ospf version 2. *Internet RFC 2328*, 1998.

[41] Jaroslav Opatrny. Total ordering problem. *SIAM J. Comput.*, 8(1):111–114, 1979.

[42] C. Perkins. Ip encapsulation within ip. *RFC 2003*, 1996.

[43] Y. Rekhter, T.J. Watson, and T. Li. Border gateway protocol 4. *RFC 1771*, 1995.

[44] Stefan Savage, Andy Collins, Eric Hoffman, John Snell, and Thomas Anderson. The end-to-end effects of internet path selection. In *Proc. of ACM SIGCOMM*, pages 289–299, 1999.

[45] Yuval Shavitt and Eran Shir. Dimes: Let the internet measure itself. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 71–74, October 2005.

[46] G. Siganos and M. Faloutsos. Analyzing bgp policies: Methodology and tool. In *Proc. of IEEE INFOCOM*, 2004.

[47] Neil Spring, Ratul Mahajan, and Thomas Anderson. The causes of path inflation. In *Proc. of ACM SIGCOMM*, pages 113–124, 2003.

[48] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H. Katz. Characterizing the internet hierarchy from multiple vantage points. In *Proc. of IEEE INFOCOM*, 2002.

[49] Bernard M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, December 1988.

שהאלגוריתם שלנו מסווג את סוג הקישוריות בין מ״א טוב יותר בהשוואה לאלגוריתמים הקודמים.

בפרק 4 אנו חוקרים את בעיית ניפוח המסלולים, שבה בגלל מדיניות הניתוב, מסלולי הניתוב בין מ״א הם לא בהכרח הקצרים ביותר. כדי להתגבר על בעיה זו ולנתב חבילות על המסלול הקצר ביותר, אנו מציעים שיטת ניתוב חדשה שבה החבילות מנותבות דרך צמתי בינים בהם מותקנים שרתי ממסר ייעודיים. אנו חוקרים בעיה זו כבעיית אופטימיזציה שבה המטרה היא לאפשר ניתוב דרך המסלולים הקצרים ביותר תוך שימוש במספר מינימאלי של שרתי ממסר. אנו מראים שבעיה זו הינה קשה להכרעה, מציגים אלגוריתם קירוב של $O(\log(n))$ ומראים שזה הקירוב הטוב ביותר שניתן להשיג. אנו בוחנים את השיטה על מספר תרחישים ומראים שבתרחישים אלו מספר קטן של שרתי ממסר מאפשר ניתוב במסלולים הקצרים בין כמעט כל זוגות הצמתים, וע״י כך מקצר את אורכם של המסלולים המנופחים בכ- 40%.

תזה זו שופכת אור על מספר בעיות פתוחות הקשורות למבנה הטופולוגי של האינטרנט שנגזרות ממדיניות הניתוב של מ״א. אנו מקווים כי התובנות שמוצגות בה יעזרו לשפר ולייעל את השימוש באינטרנט הן בשיפור פרוטוקולים קיימים והן בפיתוח אלגוריתמים ופרוטוקולים חדשים. אנו מסכמים את העבודה ומציגים מספר מסקנות בפרק 5.

מנחים ביחס למדיניות הניתוב ומידע בנוגע לדרך שבה הנתונים נאספו. לפי [7] ו-[31] כאשר מ״א מפיצה מידע על מסלולי הניתוב שעוברים דרכה לאחד מהספקים שלה (או אחד השכנים שווי המעמד שלה), היא תסנן ולא תפיץ מסלולי ניתוב שמקורם בספקים אחרים או בשכנים אחרים שווי מעמד של אותה מ״א. לפיכך, מסלולי הניתוב ברמת המ״א אינם מכילים ״עמק״ או ״צעד״, כלומר, לאחר מעבר בחיבור מסוג *ספק-לקוח* או חיבור מסוג *שווה-מעמד*, המסלול לא יעבור דרך חיבור מסוג *לקוח-ספק* או חיבור מסוג *שווה-מעמד* [24]. Gao היית ה הראשונה להסיק את סוג הקישוריות מתוך אוסף של טבלאות ניתוב, שמכילים קבוצה של מסלולי ניתוב חוקיים [24]. במאמר זה פיתחו המחברים אלגוריתם היוריסטי המניח שבדרך כלל הספק גדול יותר מהלקוח שלו, וגודל המ״א הוא יחסי לדרגתו בגרף הקישוריות. תוצאות ניסוייו ת הראו ש-90% מהחיבורים ברמת המ״א ברשת הם מסוג *ספק-לקוח* וכ-10% הם מסוג *שווה-מעמד*.

בהמשך לעבודה זו, הוגדרה בעיית סוג הקישוריות (Type Of Relationship) באופן פורמאלי ב-[48] כבעיית אופטימיזציה שבה יש לקבוע את סוג הקישוריות כך שמספר המסלולים החוקיים יהיה מקסימאלי. ב-[11] הראו המחברים שהבעייה היא קשה להכרעה במקרה הכללי, ושבעיית האופטימיזציה הינה קשה לקירוב. בנוסף הם הראו אלגוריתם לינארי שקובע אם קיים פתרון שבו כל המסלולים הם חוקיים.

בעיית הקישוריות הינה דוגמא למאמץ שנעשה עד כה כדי ללמוד ולהבין את המבנה הטופולוגי של האינטרנט. עם זאת אנחנו רחוקים מלדעת את התמונה האמיתית. מטרת תזה זו היא לחקור את המבנה הטופולוגי של האינטרנט, תוך התמקדות ברמת המ״א. בפרק 2 אנו חוקרים בעיה בסיסית: מהו גודל האינטרנט. ברמת המ״א המשמעות היא: כמה יחסי קישוריות (לינקים) קיימים בין מ״א. מציאת מספר זה אינה בעיה פשוטה מכיוון שאין דרך ישירה לקבל אינפורמציה מן הצמתים לגבי השכנים הישירים שלהם וכל הנתונים שלנו מבוססים על תהליכי דגימה. לכן, קשה מאוד לאפיין את האינטרנט מכיוון שכל אפיון שנציע יתבסס על מידע חלקי ולא ישקף את התמונה השלמה. בפרק זה אנו מביאים סימוכין לכך שלפחות 35% מהקישורים ברמת המ״א עדיין לא התגלו ואינם נמצאים בבסיסי הנתונים השונים. הממצאים שלנו מעידים על כך שרוב הקישורים האלה הם מסוג *שווה-מעמד*. בנוסף אנו בוחנים את התפלגות דרגות הצמתים בגרף הקישוריות ומראים שבעוד שתת הגרף המורכב מקשתות מסוג *ספק-לקוח* מקיים את כלל החזקה, תת הגרף המורכב מקשתות מסוג *שווה-מעמד* מתנהג באופן שונה.

בפרק 3 אנו חוקרים את בעיית סוג הקישוריות. מסתבר שההגדרה המקורית של הבעיה, כפי שהוצגה לעיל, אינה משקפת את הקשרים המסחריים בין מ״א באופן מלא. בפרט, היא אינה משקפת את המבנה ההירארכי של גרף הקישוריות שבו לא יתכנו מעגלים מכוונים. כדי לישב בעיה זו אנו מגדירים בעיה חדשה שנקראת בעיית סוג הקישוריות חסרת המעגלים (Acyclic Type of Relationship). אנו מציגים אלגוריתם יעיל שקובע האם ישנו פתרון חסר מעגלים שבו אין מסלולים לא חוקיים ומראים שהבעיה הכללית הינה קשה להכרעה. בנוסף אנו מציגים אלגוריתם קירוב של $\frac{2}{3}$ לגרסת המקסימום של בעיית האופטימיזציה וכן אנו דנים באספקטים המעשיים של מציאת סוג הקישוריות בין מ״א. כמו כן אנו מציגים תוצאות ניסוייות וסימולציה המראים

תקציר

המבנה הטופולוגי של האינטרנט הוא נושא חשוב ומעניין, שנחקר רבות בשנים האחרונות. מלבד האתגר האינטלקטואלי הכרוך בהבנת מערכת כה גדולה ומורכבת, הכרת מבנה האינטרנט חשובה לצורך שיפור פרוטוקולים קיימים ופיתוח אלגוריתמים חדשים לסביבה מורכבת זו.

העבודות בנושא זה מורכבות מאיסוף נתונים על מבנה האינטרנט, הסקת מסקנות מנתונים אלה לגבי המבנה האמיתי של האינטרנט ומציאת תכונות הנגזרות מניתוח מבנה זה. אחת הבעיות שמתעוררות בהקשר זה היא, שאיסוף הנתונים הוא חלקי בלבד ואינו מכיל את כל המידע על המבנה המלא של האינטרנט. הסיבה לכך היא שמעבר לעובדה שמדובר במערכת גדולה מאוד אין אפשרות ישירה לקבל אינפורמציה מצומת מסוים באינטרנט לגבי שכניו הישירים ולכן, קשה להסיק מסקנות לגבי מבנה האינטרנט, מכיוון שמסקנות אלה נגזרות מתהליך הדגימה של הרשת שהוא כאמור חלקי בלבד ואינו משקף את התמונה האמיתית.

האינטרנט מורכב מלמעלה מ- 20,000 מערכות אוטונומי ות (מ״א, Autonomous System), כאשר כל מ״א היא אוסף של נתבים המנוהלים תחת רשות מנהלתית יחידה. בעוד שהניתוב בתוך מ״א נעשה באמצעות פרוטוקול מעבר פנימי (Interior Gateway Protocol) כגון OSPF, הניתוב בין מ״א נעשה באמצעות פרוטוקול BGP (Border Gateway Protocol). אחד מיתרונותיו של BGP הוא יכולתו לנתב חבילות באינטרנט לפי מדיניות ניתוב, כאשר כל מ״א מגדירה את מדיניות הניתוב המקומית שלה. בפועל, משקפת מדיניות הניתוב של מ״א את יחסיה המסחריים עם מ״א אחרות. מסיבה זו, לאינטרנט יש מבנה הירארכי שבו אם מ״א קטנה מחוברת למ״א גדולה אזי המ״א הגדולה הינה הספקית והמ״א הקטנה הינה הלקוח (חיבור כזה נקרא *לקוח-ספק*, customer-provider). לעומת זאת כאשר שני מ״א שהינן באותו סדר גודל מחוברות אחת לשנייה אזי החיבור בניהן הוא *שווה-מעמד* (peer-peer).

במקביל למאמץ המעשי לאסוף מידע ולבנות בסיס נתונים שלם ככל האפשר, מחקרים תיאורטיים רבים נעשו במטרה לבנות מודל שיתאר את המבנה ההירארכי ואת מפת ה קישוריות של האינטרנט. ב-[22] Faloutsos ועמיתיו חקרו את המבנה הטופולוגי של האינטרנט והראו שמתקיים כלל חזקה (power-law) ברמת המ״א וכן ברמת הנתבים. אבחנה זו אומצה ע״י עבודות רבות שהציעו אלגוריתמים שמייצרים גרפים שמקיימים את כלל החזקה במטרה לתאר את מפת הקישוריות בין מ״א (למשל [8]). לעומת זאת עבודות אחרות הציעו מודלים אחרים לתיאו ר מבנה האינטרנט ברמת המ״א. למשל מודלים המבוססים על [49], לוקחים בחשבון את המיקום הגיאוגראפי של מ״א. התאם למודל זה, הסיכוי ששני מ״א יהיו מחוברות הוא יחסי למרחק הגיאוגראפי בניהן.

לא זו בלבד שנתונים שנאספים ע״י הפרויקטים השונים אינם מכילים את כל מפת הקישוריות, הם אינם מכילים גם את סוג הקישוריות בין מ״א. לכן, במטרה לתת תמונה שלמה יותר, אנו צריכים להסיק מהו סוג הקישוריות מתוך המידע שנאסף. דבר זה נעשה באמצעות הנחות וקווים

המחקר נעשה בהנחיית פרופ׳ דני רז בפקולטה למדעי המחשב.

אני רוצה להודות לדני על עזרתו הרבה והמסורה בכל שלבי המחקר.

אני מודה לטכניון על התמיכה הכספית הנדיבה בהשתלמותי.

המבנה הטופולוגי של האינטרנט: מתהליך הגילוי ועד לתמונה האמיתית

חיבור על מחקר

לשם מילוי חלקי של הדרישות לקבלת תואר
דוקטור לפילוסופיה

רמי כהן

הוגש לסנט הטכניון – מכון טכנולוגי לישראל
כסלו תשס״ח חיפה דצמבר 2007

המבנה הטופולוגי של האינטרנט: מתהליך הגילוי ועד לתמונה האמיתית

רמי כהן