COSIN

Algorithms for network traffic analysis

COSIN

***Co**evolution and **S**elf-organisation **I**n dynamical **N**etworks*

IST-2001-33555

Project: RTD FET Open Project IST-2001-33555

URL: http://www.cosin.org

Deliverable: nr.17, "Algorithms for network traffic analysis"

Contract: **-Preparation Form** http://www.cosin.org/doc/COSIN.xls

-Annex 1 http://www.cosin.org/doc/annex1.ps

Author: Maurizio Patrignani patrigna@dia.uniroma3.it

Date: 24th October 2004

Status: Final version.

Abstract

As routing in the Internet follows a hierarchical scheme, the analysis of the Autonomous System graph, where routing is determined at the higher level, is the first step towards the study of traffic flows, routing changes, and routing instabilities occurring in the Internet. In order to consolidate the theoretical foundations of such studies, we first address the problem of obtaining correct data from the available data sources. Second, we investigate the problem of computing the types of the relationships between Autonomous Systems (ASes), showing the NP-hardness of the problem and producing effective and efficient heuristics to approach it. Finally, we study how clustering techniques can be adapted and improved in order to be used in this domain and show how the clustered view of the AS graph can provide high level information about the network evolution at different time scales.

1. CONTENTS

1. CONTENTS 2
2. INTRODUCTION 3
3. AN INVESTIGATION ABOUT DATA SOURCES 4
   BGP updates archives 5
   BGP routing tables archives 5
4. COMPUTING THE RELATIONSHIPS BETWEEN AUTONOMOUS SYSTEMS 6
   Complexity analysis 6
   Algorithms and heuristics 8
5. AN EXPLORATION OF CLUSTERING TECHNIQUES 10
   Clustering indices and algorithms 10
   Comparing a novel approach with the state of the art 10
   Clustering the graph of Autonomous Systems 10
6. REFERENCES 11
7. APPENDIX 12

2. INTRODUCTION

In the Internet packets are delivered to their destinations through a cooperative process operated by intermediate systems called routers. The set of routers that are under a single independent administrative organization, together with the cables that connect them, compose an Autonomous System (AS). Currently, there are more than 18,000 ASes and their number is rapidly growing. Hierarchical routing imposes that first the routing is determined at the Autonomous System level, deciding the sequence of ASes to be traversed in order to reach the destination, and then at a lower level, choosing the sequence of routers to be traversed inside each AS. Thus, the comprehension of the relationships between ASes is the first step towards the analysis of network flows, routing changes, and routing instabilities occurring in the Internet. In fact, changes in the routing at the AS level will cause (sometimes considerable) changes of the routing at the router level.
The cooperation between ASes guarantees the delivery of the packets through the network, as they interact to coordinate the IP traffic delivery, exchanging routing information with a protocol called Border Gateway Protocol (BGP) [1]. BGP is a *distance vector* protocol which adopts a peculiar way of measuring the distance of a destination. In fact the distance is not an absolute value, as it is for other distance vector protocols, but is the length of the AS-path, that is the sequence of ASes to be traversed in order to reach the destination. The AS-path is explicitly communicated to the adjacent ASes when signalling the existence of a remote destination. This is why BGP is sometimes classified as a *path vector* protocol. Thus, although each AS exchanges traffic flows with a few neighbors (*BGP peers*), local and remote adjacencies can be gathered from the logs of the BGP conversations between peers, which contain the AS-path attached to each announcement. Due to this BGP peculiarity, plenty of up-to-date and historic data is potentially available about the ASes, their peerings, and the AS-level routes used to reach remote destinations.

The sequence of ASes used to reach a destination is decided by the kind of relationships between the ASes that may allow or not to be traversed. These relationships, as pointed out by several researchers (see, e.g. [2, 3]) can be roughly classified into categories that have both a commercial and a technical flavor. A pair of ASes such that one sells/offers Internet connectivity to the other is said to have a *provider-customer* relationship. If two ASes simply provide connectivity between their respective customers they are said to have a *peer-to-peer* relationship. Finally, if two ASes offer each other Internet connectivity they are said to be *siblings*. Of course, this classification does not capture all the shades of the possible commercial agreements and technical details that govern the traffic exchanges between ASes but should be considered as an important attempt toward understanding the Internet structure.

Gao [4] studies, for the first time, the following problem. ASes are the vertices of a graph (*AS graph*) where two ASes are adjacent if they exchange routing information; the edges of such a graph should be labelled in order to reflect the type of relationship they have. In order to infer the relationships between ASes, Gao uses the information on the degree of ASes together with the *AS paths* extracted from the BGP routing tables. An AS path is the sequence of the ASes traversed by a connectivity offer (*BGP announcement*). In [4] a heuristic is presented together with experimental results. An analysis on the properties of the labelled graphs obtained with such heuristics is provided in [5].

Subramanian et al. [6] formally define, as an optimization problem, a slightly simplified version of the problem addressed in [4] and conjecture its NP-completeness. They also propose a heuristic based on the observation of the Internet from multiple vantage points, which does not rely on the degree of the ASes. Further, they validate the results obtained by the heuristic against a rich collection of data sets.

Our contribution in the framework of the COSIN project is meant to consolidate the theoretical foundations of such studies. In particular, we first address the problem of obtaining correct data from the available data sources, namely archives of BGP updates and archives of BGP routing tables (this research is described in Section 3 of this deliverable). Second, following the line of research opened in [4, 6], we characterize the complexity of the problem of computing the type of the relationships between autonomous systems, showing both NP-completeness results and efficient algorithms for solving specific cases (Section 4). Motivated by the hardness of the general problem, we propose approximation algorithms and heuristics based on a novel paradigm and show their effectiveness against publicly available data sets. The experiments put in evidence that our algorithms performs significantly better than state of the art heuristics.
Finally, we compare existing clustering techniques and propose a new one [7] that is promising for coping with the complexity of this domain of research. In fact, we used it to study the evolution of AS clusters through time [8] showing how the clustered view could provide high level information about dynamic phenomena occurring in the network at different time scales (Section 5).

3. AN INVESTIGATION ABOUT DATA SOURCES

Data about Internet routing at the Autonomous System level come from two different, although strongly related, sources: BGP updates (information flowing through the links between ASes) and BGP routing tables (information stored into the nodes of the AS graph that can be dumped periodically). We considered both sources. It is important to notice that, even if one is interested in the BGP routing tables only, it is sometimes needed to retrieve the BGP updates falling into a specified time window. Suppose, for example, that a BGP routing table $R(t_x)$ at time t_x is needed, but only a BGP routing table at time $R(t_y)$, with $t_y < t_x$, is available. Then, if the BGP updates occurred in the interval $[t_y,t_x]$ are retrieved, it is possible to “drag” $R(t_y)$, taking into account the updates between t_y and t_x and producing the needed $R(t_x)$.

BGP updates archives

Several archives, such as the Routing Information System of the RIPE NCC (www.ripe.net) and the Oregon RouteViews database (www.routeviews.org), try to give an answer to the need for information about current and historic routing in the Internet by collecting BGP conversations between routers. We describe in [15] a work that aims at integrating the data from different BGP-update sources. We address the technological issues related to the data integration and the pitfalls that could give rise to imprecise information if the query process does not take into account the peculiarities and glitches of the repositories.
A peculiar problem arises when considering historical updates. In particular, updates preceding year 2001 may have a poor synchronization. A methodology to discover this is the following: measure the bursts (number of updates) of each source (for example, a RIS or an Oregon RouteViews collector) and graphic the correlation between a pair of sources. In some occasions it can be noticed that the correlation is maximum if one source had the times translated of a certain amount of seconds. If this comparison is made for all pairs of sources, this gives rise to a collection of times which is consistent one with the others. Starting with 2001 the clock of the machines was automatically set to ensure synchronization between the collectors.

BGP routing tables archives

Data on the actual routing can be also collected by logging the BGP routing tables. The RIS project dumps a BGP routing table three times per day (8:00, 16:00, and 24:00). The Oregon RouteViews project, instead, dumps a BGP routing table every two hours. This implies that if a collection of AS-path is needed, representing the status of the network at a specific time t_x, two routing tables $R(t_1)$ and $R(t_2)$, from RIS and Oregon RouteViews, respectively, must be retrieved and "dragged" to time t_x as described above, after retrieving the updates in the intervals $[t_1,t_x]$ and $[t_2,t_x]$ from RIS and Oregon RouteViews, respectively.
Usually the resulting data is not purified and contains misleading information. The most common problems are the following:

- **Repetitions of paths**. These can be undesired if the purpose of the research is to discover which links between pair of ASes are used for routing packets. In this case repeated paths can be efficiently eliminated after sorting the routing table with respect to the AS-path field.
- **Loops inside paths**. Loops shouldn't occur in AS-paths since the BGP protocol imposes that a border router of AS x must discard any BGP announcement received with an associated AS-path containing x. However, sometimes loops can be introduced into the AS-paths by an improper prepending command made on the BGP configuration of an intermediate router. In this case, two are the possible alternatives: trying to reconstruct the original AS-paths or discarding all AS-paths that contain loops.

- **Illegal AS numbers**. The last 1024 AS numbers are considered "private" and should not be propagated into the Internet. However, some AS-path contains a private AS numbers. In this case the AS-path can be discarded or, alternatively, the illegal AS numbers could be removed.
- **Prepending of ASes**. As for repetitions of AS-paths, prepending can be undesired if the purpose of the research is to discover the AS-paths used.

Further, other issues can alter data, like truncated data files, time-outs, lost connections, or connections closed too early [8]. These kinds of problems can be recognized by considering that a BGP dump is usually sorted by the prefix of the announcements, starting from 0.0.0.0 and ending with 255.255.255.255. It follows that incomplete data is easily recognizable from the fact that the prefixes falling in the higher part of the IPv4 address space will be missing from the list of prefixes.

4. COMPUTING THE RELATIONSHIPS BETWEEN AUTONOMOUS SYSTEMS

Complexity analysis

In [11-14] we solve a problem explicitly stated and left open in [6]. Namely, we characterize the complexity of determining the relationships between ASes while maximizing the number of valid paths, i.e., the number of paths coherent with the edge labelling.
The first formulation of this problem was given in [4], where for the first time it was noted that the types of relationships between ASes induce some constraints on the BGP AS-paths that can be observed in Internet. In fact, a customer will never announce a prefix obtained by a first provider to a second provider (because in this way traffic between the two providers could flow through the customer network). In a peer-to-peer relationship, both ASes announce their prefixes and the prefixes of their customers, but they will not announce the prefixes received from their providers and from other peer-to-peer links. These rules of thumb for network administrators offer a precious hint on the kind of relationship occurring between two adjacent ASes. Namely, if all the ASes obey to them, an AS-path should be composed by a (possibly null) sequence of edges going from a customer to a provider, a single (possibly missing) peer-to-peer edge, and a (possibly null) sequence of edges going from a provider to a customer. The amount of AS-paths that have this "valley-free" structure can be used as a measure of the goodness of the inference heuristics. In [4], several kinds of possible relationships between ASes are considered, and the problem of inferring them by observing the AS-paths propagated by the BPG protocol is for the first time addressed without explicitly formulating it as an optimization problem.
In [6], instead, the attention is focused on two kinds of relationships only: the customer-provider and the peer-to-peer relationships. With this simplification, the

problem of determining the relationships between ASes can be stated as an optimization problem on an undirected graph [6]:

> **Type-of-Relationship (ToR) Problem**: Given an undirected graph G and a set of paths P, give an orientation to some of the edges of G to maximize the number of valid paths in P.

In this formulation it is assumed that not directed edges correspond to peer-to-peer relationships, that directed edges point to the provider, and that a valid path is a "valley free" path with the structure described above.

In [11-14] we show that:

- We show that the problem is NP-hard in the general case and cannot even be approximated within a factor of $1/n^{1-\varepsilon}$ unless NP=coRP, where n is the number of given paths;

- We show that the problem remains NP-hard in the case where all given paths are short (i.e., have length at most l, where l is an arbitrary constant greater or equal to 2) - more specifically, we even show that there is a constant less than one such that it is NP-hard to approximate the problem within that constant factor.

- We produce a linear time algorithm for determining the AS relationships in the case in which the problem admits a solution that makes all paths valid (i.e., does not have any anomalies); and

- We use the linear time algorithm to show that for large portions of the Internet (e.g., data obtained from single points of view) it is often possible to determine the relationships between ASes without anomalies.

The first two complexity results are based on suitable reductions from problems whose complexity is well known. The algorithm for inferring the AS relationships in the case in which the problem admits a solution with all valid paths is obtained by producing an instance of the the 2SAT problem starting from an instance of the ToR problem.
First, in [11-14] it is shown that the ToR problem left open in [6] is equivalent to a similar problem in which all edges must be oriented:

> **ToR-simple Problem**: Given an undirected graph G and a set of paths P give an orientation to all the edges of G to maximize the number of valid paths in P.

Based on this observation, it is possible to build an instance of the 2SAT problem corresponding to the ToR-simple instance with the following rules [11-14]:

- First give an arbitrary orientation to each edge (v_i, v_j) of G.
- For each directed edge (v_i, v_j) of G a variable $x_{i,j}$ is introduced. A true value for $x_{i,j}$ means that, in the final orientation, (v_i, v_j) will be directed from v_i to v_j (that is, the direction of the initial arbitrary orientation will be preserved), while a false value means that (v_i, v_j) will be directed from v_j to v_i (that is, the direction of the initial arbitrary orientation will be reversed).
- For each path p and three consecutive vertices v_{i-1}, v_i, v_{i+1} of p the following four cases are possible, according to the arbitrary orientations that we have given to the edges between v_{i-1}, v_i, and v_{i+1}.
  - Both edges are directed toward v_i, i.e. such directed edges are (v_{i-1}, v_i) and (v_{i+1}, v_i). We introduce clause ($x_{i-1,i}$ OR $x_{i+1,i}$).
  - Both edges are directed away from v_i, i.e. such directed edges are (v_i, v_{i-1}) and (v_i, v_{i+1}). We introduce clause ($x_{i,i-1}$ OR $x_{i,i+1}$).
  - One edge is directed toward v_i and the other toward v_{i+1}, i.e. such directed edges are (v_{i-1}, v_i) and (v_i, v_{i+1}). We introduce clause ($x_{i-1,i}$ OR $x_{i,i+1}$).
  - One edge is directed toward v_{i-1} and the other toward v_i, i.e. such directed edges are (v_i, v_{i-1}) and (v_{i+1}, v_i). We introduce clause ($x_{i,i-1}$ OR $x_{i+1,i}$).

This procedure introduces n-2 clauses of two literals for each path of P with n vertices. If the 2SAT formula composed by all the clauses admits a solution, it can be found in linear time by using standard techniques, and the 2SAT solution corresponds to an orientation for the ToR problem with all valid paths.

Algorithms and heuristics

Since in the general case the ToR problem is NP-hard, it is of practical importance to develop efficient heuristics to find a good solution for it. We introduce algorithms, based on novel approaches, for determining the relationships between ASes with a large number of valid paths (or equivalently, a small number of anomalous paths). For some of the algorithms we can prove that their solution is guaranteed to be within a constant factor of the optimal solution [11-14]. Also, we experimentally show that the proposed approaches lead to algorithms that perform significantly better than the cutting edge heuristics of [6].

Our heuristics are mainly based on two different strategies, both relying on the relationship between the ToR instance and the 2SAT instance described above. The first strategy consists of using approximation algorithms for MAX2SAT, while the second strategy consists of using the mapping of the ToR instance to the 2SAT instance in order to find a maximal subset of paths admitting a solution without invalid paths. Both approaches are described in [11-14], and their effectiveness is compared with that of [6].

Using approximation algorithms for MAX2SAT. An approximation algorithm for MAX2SAT finds an assignment to the variables of the 2SAT instance so that the number of satisfied clauses is satisfied. As it is shown above, for each path of P with n vertices of the ToR instance the corresponding 2SAT instance has n-2 clauses. Thus, an approximation algorithm for MAX2SAT which maximize the number of satisfied clauses is not guaranteed to maximize the number of the corresponding valid paths of the ToR problem. However, in [11-12] it is shown that by using a SDP-based heuristics for approximating the MAX2SAT problem, a good solution can be found for the original ToR instance too.

Computing a maximal ToR instance admitting an easy solution. A simple strategy for computing a maximal set of paths admitting a solution without invalid paths could be the following. Starting from the empty set, add all the paths one-by-one, each time testing if the set admits an orientation without anomalies. The test can be performed in linear time by exploiting the algorithm described above. If the insertion of a path makes the set not orientable, then it is discarded, otherwise it is added to the set. At the end of the process we have a maximal set of paths. However, this simple strategy is infeasible. In fact we would have to run the testing algorithm millions of times. Even if each run takes only one second, it could take weeks until the maximal set is computed.

Motivated by the above consideration, we propose in [13-14] a two-phase approach. In the first phase, we compute a very large (albeit not maximal) set of valid paths with an ad-hoc technique. In the second phase, we check if the discarded paths can be reinserted with the method described above.

The first phase, i.e., the computation of the initial very large set of valid paths, is performed by starting with all paths in P and removing iteratively all the paths that use an edge (v_i, v_j) corresponding to a variable $x_{i,j}$ of the corresponding 2SAT formula which generates a contradiction (that is, can not be satisfied). Observe that at each iteration, since we remove all the paths traversing a specific edge of the AS graph, the literals associated with such an edge disappear from 2SAT. At the end of the first phase, we have a large set P' of paths for which there is an orientation of the graph that makes all paths in P' valid. In the second phase, we consider each path π in $P \setminus P'$ and check whether there is an orientation that makes all paths in P' union $\{\pi\}$ valid. If this is the case, we add π to P', otherwise we discard π and leave P' unchanged. At the end of the second phase, we have a maximal set P' of paths that admits an orientation without anomalies, and we compute such an orientation with the above described algorithm.

5. AN EXPLORATION OF CLUSTERING TECHNIQUES

Clustering indices and algorithms

Clustering is an important issue in the analysis and exploration of data. Roughly speaking, clustering consists in discovering natural groups of similar elements in data sets. An interesting and important variant of data clustering is graph clustering, also motivated by the growing interest in network analysis.
A natural notion of graph clustering is the separation of sparsely connected dense subgraphs from each other. Several formalizations have been proposed. However, the understanding of current algorithms and indices is still rather intuitive. In particular, we note that it is not precisely known how well indices formalizing the relation between the number of intra-cluster and inter-cluster edges measure the quality of a graph clustering. Moreover, there exists no conclusive evaluation of algorithms that focus on such indices.

Comparing a novel approach with the state of the art

As a first step towards understanding the consequences of particular conceptions, we concentrate on the indices and algorithms mentioned above. We give in [7] a summary of clustering indices and conduct an experimental evaluation of graph clustering approaches. The already known algorithms under comparison are the iterative conductance cut algorithm presented in [9] and the Markov clustering approach from [10]. By combining proven techniques from graph partitioning and geometric clustering, we also introduce a new approach that compares favorably with respect to flexibility and running time [7].

Clustering the graph of Autonomous Systems

In [8] we applied the novel approach to graph clustering introduced in [7] to the problem of clustering the graph of the Autonomous Systems with the purpose of understanding the dynamic phenomena occurring in the network. We analyzed the data collected by the Oregon RouteViews project (www.routeviews.org) during the last few years. Our results confirmed known facts about the stability of the Internet. We were able to measure major trends and noticed that events occurring on a smaller time-frame, like worm-attacks, misconfigurations, outages, DDoS attacks, etc. seem to have a very diverse degree of impact on the AS graph structure resulting from the clustering, which suggests that these techniques could be used to distinguish some of them.

6. REFERENCES

1. John W. Stewart, *BGP4: Inter-Domain Routing in the Internet*, Addison-Wesley, Reading, MA, 1999.
2. C. Alaettinoglu, *Scalable Router Configuration for the Internet*, Proc. IEEE IC3N, October 1996.
3. Geoff Huston, Interconnection, Peering, and Settlements, Proc. INET, June 1999.
4. Lixin Gao, On inferring autonomous system relationships in the internet, IEEE/ACM Transactions on Networking, **9**(6), pag. 733—745, Dec 2001.
5. Zihui Ge, Daniel Ratton Figueiredo, Sharad Jaiswal, and Lixin Gao, On the Hierarchical Structure of the Logical Internet Graph, Proc. SPIE ITCom 2001.
6. Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and R.H. Katz, Characterizing the Internet Hierarchy from Multiple Vantage Points, Proc. IEEE INFOCOM 2002, 2002.
7. Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. Experiments on Graph Clustering. In Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03), volume 2832 of Lecture Notes in Computer Science, pages 568-579. Springer-Verlag, 2003.
8. Marco Gaertler and Maurizio Patrignani, "Dynamic Analysis of the Autonomous System Graph", in Proceedings of IPS 2004, International Workshop on Inter-domain Performance and Simulation, Budapest, Hungary, 22-23 March, 2004.
9. Kannan, R., Vampala, S., Vetta, A.: On Clustering — Good, Bad and Spectral. In: Foundations of Computer Science 2000. (2000) 367-378
10. van Dongen, S.M.: Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht (2000)
11. Thomas Erlebach, Alexander Hall and Thomas Schank, “Classifying Customer-Provider Relationships in the Internet” Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN 2002), Cambridge, USA, November 4-6, 2002, pp. 538-545.
12. Thomas Erlebach, Alexander Hall, Thomas Schank: Classifying Customer-Provider Relationships in the Internet. TIK Report Nr. 145, July, 2002.
13. Giuseppe Di Battista, Maurizio Patrignani, and Maurizio Pizzonia, "Computing the types of the Relationships between Autonomous Systems", Technical Report RT-DIA-73-2002, Dipartimento di Informatica e Automazione, Università di Roma Tre, Rome, 2002.
14. Giuseppe Di Battista, Maurizio Patrignani, and Maurizio Pizzonia, "Computing the Types of the Relationships between Autonomous Systems", in Proceedings of IEEE INFOCOM 2003, The Conference on Computer Communications, The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies.
15. Giuseppe Di Battista, Federico Mariani, Maurizio Patrignani, and Maurizio Pizzonia, "Archives of BGP Updates: Integration and Visualization" in Proceedings of IPS 2003, International Workshop on Inter-domain Performance and Simulation, Salzburg, Austria, 20-21 February, 2003, pages 123-129.

7. APPENDIX

The following papers are attached to this deliverable

Appendix 7.1 Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. Experiments on Graph Clustering. In Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03), volume 2832 of Lecture Notes in Computer Science, pages 568-579. Springer-Verlag, 2003.

Appendix 7.2 Marco Gaertler and Maurizio Patrignani, "Dynamic Analysis of the Autonomous System Graph", in Proceedings of IPS 2004, International Workshop on Inter-domain Performance and Simulation, Budapest, Hungary, 22-23 March, 2004.

Appendix 7.3 Thomas Erlebach, Alexander Hall, Thomas Schank: Classifying Customer-Provider Relationships in the Internet. TIK Report Nr. 145, July, 2002.

Appendix 7.4 Giuseppe Di Battista, Maurizio Patrignani, and Maurizio Pizzonia, "Computing the Types of the Relationships between Autonomous Systems", in Proceedings of IEEE INFOCOM 2003, The Conference on Computer Communications, The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies.

Appendix 7.5 Giuseppe Di Battista, Federico Mariani, Maurizio Patrignani, and Maurizio Pizzonia, "Archives of BGP Updates: Integration and Visualization" in Proceedings of IPS 2003, International Workshop on Inter-domain Performance and Simulation, Salzburg, Austria, 20-21 February, 2003, pages 123-129.

Appendix 7.1

Experiments on Graph Clustering Algorithms*

Ulrik Brandes[1], Marco Gaertler[2], and Dorothea Wagner[2]

[1] University of Passau, Department of Mathematics & Computer Science, 94030 Passau, Germany. brandes@algo.fmi.uni-passau.de
[2] University of Karlsruhe, Faculty of Informatics, 76128 Karlsruhe, Germany. {dwagner,gaertler}@ira.uka.de

Abstract. A promising approach to graph clustering is based on the intuitive notion of intra-cluster density vs. inter-cluster sparsity. While both formalizations and algorithms focusing on particular aspects of this rather vague concept have been proposed no conclusive argument on their appropriateness has been given.
As a first step towards understanding the consequences of particular conceptions, we conducted an experimental evaluation of graph clustering approaches. By combining proven techniques from graph partitioning and geometric clustering, we also introduce a new approach that compares favorably.

1 Introduction

Clustering is an important issue in the analysis and exploration of data. There is a wide area of applications as e.g. data mining, VLSI design, computer graphics and gene analysis. See also [1] and [2] for an overview. Roughly speaking, clustering consists in discovering natural groups of similar elements in data sets. An interesting and important variant of data clustering is graph clustering. On one hand, similarity is often expressed by a graph. On the other hand, there is a growing interest in network analysis in general.

A natural notion of graph clustering is the separation of sparsely connected dense subgraphs from each other. Several formalizations have been proposed. However, the understanding of current algorithms and indices is still rather intuitive. As a first step towards understanding the consequences of particular conceptions, we concentrate on indices and algorithms that focus on the relation between the number of intra-cluster and inter-cluster edges.

In [3] some indices measuring the quality of a graph clustering are discussed. Conductance, an index concentrating on the intra-cluster edges is introduced and a clustering algorithm that repeatedly separates the graph is presented. A graph clustering algorithm incorporating the idea of performing a random walk on the graph to identify the more densely connected subgraphs is presented in [4] and the index *performance* is considered to measure the quality of a graph

* This work was partially supported by the DFG under grant BR 2158/1-1 and WA 654/13-1 and EU under grant IST-2001-33555 COSIN.

clustering. The idea of random walks is also used in [5] but only for clustering geometric data. Obviously, there is a close connection between graph clustering and the classical graph problem minimum cut. A purely graph-theoretic approach using this connection more or less directly is the recursive minimum cut approach presented in [6]. Other more advanced partition techniques involve spectral information as in [3,7,8,9].

It is not precisely known how well indices formalizing the relation between the number of intra-cluster and inter-cluster edges measure the quality of a graph clustering. Moreover, there exists no conclusive evaluation of algorithms that focus on such indices. In this paper, we give a summary of those indices and conduct an experimental evaluation of graph clustering approaches. The already known algorithms under comparison are the iterative conductance cut algorithm presented in [3] and the Markov clustering approach from [4]. By combining proven techniques from graph partitioning and geometric clustering, we also introduce a new approach that compares favorably with respect to flexibility and running time.

In Section 2 the notation used throughout the paper is introduced and clustering indices considered in the experimental study are presented. Section 3 gives a detailed description of the three algorithms considered. The graph generators used for the experimental evaluation are described in Section 4.1 and the results of the evaluation are summarized in Section 4.3.

2 Indices for Graph Clustering

Throughout this paper we assume that $G = (V, E)$ is a connected, undirected graph. Let $|V| =: n, |E| =: m$ and $\mathcal{C} = (C_1, \dots, C_k)$ a partition of V. We call $\mathcal{C}$ a *clustering* of G and the C_i *clusters*; $\mathcal{C}$ is called *trivial* if either $k = 1$, or all clusters C_i contain only one element. In the following, we often identify a cluster C_i with the induced subgraph of G, i.e. the graph $G[C_i] := (C_i, E(C_i))$, where $E(C_i) := \{\{v, w\} \in E : v, w \in C_i\}$. Then $E(\mathcal{C}) := \bigcup_{i=1}^{k} E(C_i)$ is the set of *intra-cluster edges* and $E \setminus E(\mathcal{C})$ the set of *inter-cluster edges*. The number of intra-cluster edges is denoted by $m(\mathcal{C})$ and the number of inter-cluster edges by $\overline{m}(\mathcal{C})$. A clustering $\mathcal{C} = (C, V \setminus C)$ is also called a *cut* of G and $\overline{m}(\mathcal{C})$ the *size* of the cut. A cut with minimum size is called a *mincut*.

2.1 Coverage

The *coverage*$(\mathcal{C})$ of a graph clustering $\mathcal{C}$ is the fraction of intra-cluster edges within the complete set of edges, i.e.

$$coverage(\mathcal{C}) := \frac{m(\mathcal{C})}{m} = \frac{m(\mathcal{C})}{m(\mathcal{C}) + \overline{m}(\mathcal{C})}.$$

Intuitively, the larger the value of *coverage*$(\mathcal{C})$ the better the quality of a clustering $\mathcal{C}$. Notice that a mincut has maximum coverage and in this sense would be an "optimal" clustering. However, in general a mincut is not considered to be a good clustering of a graph. Therefore, additional constraints on the number of clusters or the size of the clusters seem to be reasonable. While a mincut can be computed in polynomial time, constructing a clustering with a fixed number k, $k \geq 3$ of clusters is NP-hard [10], as well as finding a mincut satisfying certain size constraints on the clusters [11].

2.2 Performance

The *performance*$(\mathcal{C})$ of a clustering $\mathcal{C}$ counts the number of "correctly interpreted pairs of nodes" in a graph. More precisely, it is the fraction of intra-cluster edges together with non-adjacent pairs of nodes in different clusters within the set of all pairs of nodes, i.e.

$$\mathit{performance}(\mathcal{C}) := \frac{m(\mathcal{C}) + \sum_{\{v,w\} \notin E, v \in C_i, w \in C_j, i \neq j} 1}{\frac{1}{2} n(n-1)}.$$

Calculating the performance of a clustering according to this formula would be quadratic in the number of nodes. Especially, if the performance has to be computed for a sequence of clusterings of the same graph, it might be more efficient to count the number of "errors" instead (Equation (1)). Maximizing the performance is reducible to graph partitioning which is NP-hard [12].

$$1 - \mathit{performance}(\mathcal{C}) = \frac{2m\left(1 - 2\mathit{coverage}(\mathcal{C})\right) + \sum_{i=1}^{k} |C_i|\left(|C_i| - 1\right)}{n(n-1)} \tag{1}$$

2.3 Intra- and Inter-cluster Conductance

The *conductance of a cut* compares the size of the cut and the number of edges in either of the two induced subgraphs. Then the *conductance* $\phi(G)$ *of a graph* G is the minimum conductance value over all cuts of G. For a clustering $\mathcal{C} = (C_1, \ldots, C_k)$ of a graph G, the *intra-cluster conductance* $\alpha(\mathcal{C})$ is the minimum conductance value over all induced subgraphs $G[C_i]$, while the *inter-cluster conductance* $\delta(\mathcal{C})$ is the maximum conductance value over all induced cuts $(C_i, V \setminus C_i)$. For a formal definition of the different notions of conductance, let us first consider a cut $\mathcal{C} = (C, V \setminus C)$ of G and define *conductance* $\phi(C)$ and $\phi(G)$ as follows.

$$\phi(C) := \begin{cases} 1, & C \in \{\emptyset, V\} \\ 0, & C \notin \{\emptyset, V\} \text{ and } \overline{m}(\mathcal{C}) = 0 \\ \frac{\overline{m}(\mathcal{C})}{\min\left(\sum_{v \in C} \deg v, \sum_{v \in V \setminus C} \deg v\right)}, & \text{otherwise} \end{cases}$$

$$\phi(G) := \min_{C \subseteq V} \phi(C)$$

Then a cut has small conductance if its size is small relative to the density of either side of the cut. Such a cut can be considered as a bottleneck. Minimizing the conductance over all cuts of a graph and finding the according cut is

NP-hard [10], but can be approximated with poly-logarithmic approximation guarantee in general, and constant approximation guarantee for special cases, [9] and [8]. Based on the notion of conductance, we can now define intra-cluster conductance $\alpha(\mathcal{C})$ and inter-cluster conductance $\delta(\mathcal{C})$.

$$\alpha(\mathcal{C}) := \min_{i\in\{1,\dots,k\}} \phi\left(G[C_i]\right) \quad \text{and} \quad \delta(\mathcal{C}) := 1 - \max_{i\in\{1,\dots,k\}} \phi\left(C_i\right)$$

In a clustering with small intra-cluster conductance there is supposed to be at least one cluster containing a bottleneck, i.e. the clustering is possibly too coarse in this case. On the other hand, a clustering with small inter-cluster conductance is supposed to contain at least one cluster that has relatively strong connections outside, i.e. the clustering is possibly too fine. To see that a clustering with maximum intra-cluster conductance can be found in polynomial time, consider first $m = 0$. Then $\alpha(\mathcal{C}) = 0$ for every non-trivial clustering $\mathcal{C}$, since it contains at least one cluster C_j with $\phi\left(G[C_j]\right) = 0$. If $m \neq 0$, consider an edge $\{u, v\} \in E$ and the clustering $\mathcal{C}$ with $C_1 = \{u, v\}$, and $|C_i| = 1$ for $i \geq 2$. Then $\alpha(\mathcal{C}) = 1$, which is maximum.

So, intra-cluster conductance has some artifical behavior for clusterings with many small clusters. This justifies the restriction to clusterings satisfying certain additional constraints on the size or number of clusters. However, under these constraints maximizing intra-cluster conductance becomes an NP-hard problem. Finding a clustering with maximum inter-cluster conductance is NP-hard as well, because it is at least as hard as finding a cut with minimum conductance.

3 Graph Clustering Algorithms

Two graph clustering algorithms that are assumed to perform well with respect to the indices described in the previous section are outlined. The first one iteratively emphazises intra-cluster over inter-cluster connectivity and the second one repeatedly refines an initial partition based on intra-cluster conductance. While both essentially operate locally, we also propose another, more global method. In all three cases, the asymptotic worst-case running time of the algorithms depend on certain parameters given as input. However, notice that for meaningful choices of these parameters, the time complexity of the new algorithm GMC is better than for the other two.

All three algorithms employ the *normalized adjacency matrix* of G, i.e., $M(G) = D(G)^{-1}A(G)$ where $A(G)$ is the adjacency matrix and $D(G)$ the diagonal matrix of vertex degrees.

3.1 Markov Clustering (MCL)

The key intuition behind *Markov Clustering* (MCL) [4, p. 6] is that a "random walk that visits a dense cluster will likely not leave the cluster until many of its vertices have been visited." Rather than actually simulating random walks, MCL iteratively modifies a matrix of transition probabilities. Starting from $M =$

$M(G)$ (which corresponds to random walks of length at most one), the following two operations are iteratively applied:

– *expansion*, in which M is taken to the power $e \in \mathbb{N}_{>1}$ thus simulating e steps of a random walk with the current transition matrix (Algorithm 1, Step 1)
– *inflation*, in which M is re-normalized after taking every entry to its rth power, $r \in \mathbb{R}^+$. (Algoritm 1, Steps 2–4)

Note that for $r > 1$, inflation emphasizes the heterogeneity of probabilities within a row, while for $r < 1$, homogeneity is emphasized. The iteration is halted upon reaching a recurrent state or a fixpoint. A recurrent state of period $k \in \mathbb{N}$ is a matrix that is invariant under k expansions and inflations, and a fixpoint is a recurrent state of period 1. It is argued that MCL is most likely to end up in a fixpoint [4]. The clustering is induced by connected components of the graph underlying the final matrix. Pseudo-code for MCL is given in Algorithm 1. Except for the stop criterion, MCL is deterministic, and its complexity is dominated by the expansion operation which essentially consists of matrix multiplication.

Algorithm 1: Markov Clustering (MCL)

```
   Input: $G = (V, E)$, expansion parameter $e$, inflation parameter $r$

   $M \leftarrow M(G)$
   while $M$ is not fixpoint do
1      $M \leftarrow M^e$
2      forall $u \in V$ do
3          forall $v \in V$ do $M_{uv} \leftarrow M_{uv}^r$
4          forall $v \in V$ do $M_{uv} \leftarrow \frac{M_{uv}}{\sum_{w \in V} M_{uw}}$

   $H \leftarrow$ graph induced by non-zero entries of $M$
   $\mathcal{C} \leftarrow$ clustering induced by connected components of $H$
```

3.2 Iterative Conductance Cutting (ICC)

The basis of *Iterative Conductance Cutting* (ICC) [3] is to iteratively split clusters using minimum conductance cuts. Finding a cut with minimum conductance is NP–hard, therefore the following poly-logarithmic approximation algorithm is used. Consider the vertex ordering implied by an eigenvector to the second largest eigenvalue of $M(G)$. Among all cuts that split this ordering into two parts, one of minimum conductance is chosen. Splitting of a cluster ends when the approximation value of the conductance exceeds an input threshold α^* first. Pseudo-code for ICC is given in Algorithm 2. Except for the eigenvector computations, ICC is deterministic. While the overall running time depends on the number of iterations, the running time of the conductance cut approximation is dominated by the eigenvector computation which needs to be performed in each iteration.

Algorithm 2: Iterative Conductance Cutting (ICC)

```
Input: $G = (V, E)$, conductance threshold $0 < \alpha^* < 1$
$\mathcal{C} \leftarrow \{V\}$
while there is a $C \in \mathcal{C}$ with $\phi(G[C]) < \alpha^*$ do
    $x \leftarrow$ eigenvector of $M(G[C])$ associated with second largest eigenvalue
    $\mathcal{S} \leftarrow \left\{ S \subset C \colon \max_{v \in S}\{x_v\} < \min_{w \in C \setminus S}\{x_w\} \right\}$
    $C' \leftarrow \arg\min_{S \in \mathcal{S}}\{\phi(S)\}$
    $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C\}) \cup \{C', C \setminus C'\}$
```

3.3 Geometric MST Clustering (GMC)

Geometric MST Clustering (GMC), is a new graph clustering algorithm combining spectral partitioning with a geometric clustering technique. A geometric embedding of G is constructed from d distinct eigenvectors $x_1, \ldots, x_d$ of $M(G)$ associated with the largest eigenvalues less than 1. The edges of G are then weighted by a distance function induced by the embedding, and a minimum spanning tree (MST) of the weighted graph is determined. A MST T implies a sequence of clusterings as follows: For a threshold value τ let $F(T, \tau)$ be the forest induced by all edges of T with weight at most τ. For each threshold τ, the connected components of $F(T, \tau)$ induce a clustering. Note that there are at most $n - 1$ thresholds resulting in different forests. Because of the following nice property of the resulting clustering, we denote it with $\mathcal{C}(\tau)$. The proof of Lemma 1 is omitted. See [13].

Lemma 1. *The clustering induced by the connected components of $F(T, \tau)$ is independent of the particular MST T.*

Among the $\mathcal{C}(\tau)$ we choose one optimizing some measure of quality. Potential measures of quality are, e.g., the indices defined in Section 2, or combinations thereof. This genericity allows to target different properties of a clustering. Pseudo-code for GMC is given in Algorithm 3. Except for the eigenvector computations, GMC is deterministic. Note that, different from ICC, they form a preprocessing step, with their number bounded by a (typically small) input parameter. Assuming that the quality measure can be computed fast, the asymptotic time and space complexity of the main algorithm is dominated by the MST computation. GMC combines two proven concepts from geometric clustering and graph partitioning. The idea of using a MST that way has been considered before [14]. However, to our knowledge the MST decomposition was only used for geometric data before, not for graphs. In our case, general graphs without additional geometric information are considered. Instead, spectral graph theory is used [15] to obtain a geometric embedding that already incorporates insight about dense subgraphs. This induces a canonical distance on the edges which is taken for the MST computation.

Algorithm 3: Geometric MST Clustering (GMC)

```
**Input**: $G = (V, E)$, embedding dimension $d$, clustering valuation $quality$

$(1, \lambda_1, \dots, \lambda_d) \leftarrow d + 1$ largest eigenvalues of $M(G)$
$d' \leftarrow \max\{i \colon 1 \leq i \leq d, \lambda_i > 0\}$
$x^{(1)}, \dots, x^{(d')} \leftarrow$ eigenvectors of $M(G)$ associated with $\lambda_1, \dots, \lambda_{d'}$
**forall** $e = (u, v) \in E$ **do** $w(e) \leftarrow \sum_{i=1}^{d'} \left| x_u^{(i)} - x_v^{(i)} \right|$
$T \leftarrow$ MST of $G$ with respect to $w$
$\mathcal{C} \leftarrow \mathcal{C}(\tau)$ for which $quality(\mathcal{C}(\tau))$ is maximum over all $\tau \in \{w(e) \colon e \in T\}$
```

4 Experimental Evaluation

First we describe the general model used to generate appropriate instances for the experimental evaluation. Then we present the experiments and discuss the results of the evaluation.

4.1 Random Uniform Clustered Graphs

We use a random partition generator $\mathcal{P}(n, s, v)$ that determines a partition $(P_1, \dots, P_k)$ of $\{1, \dots, n\}$ with $|P_i|$ being a normal random variable with expected value s and standard deviation $\frac{s}{v}$. Note that k depends on the choice of n, s and v, and that the last element $|P_k|$ of $\mathcal{P}(n, s, v)$ is possibly significantly smaller than the others. Given a partition $\mathcal{P}(n, s, v)$ and probabilities p_{in} and p_{out}, a uniformly random clustered graph $(G, \mathcal{C})$ is generated by inserting intra-cluster edges with probability p_{in} and inter-cluster edges with probability p_{out} [1]. For a clustered graph $(G, \mathcal{C})$ generated that way, the expected values of m, $m(\mathcal{C})$ and $\overline{m}(\mathcal{C})$ can be determined. We obtain

$$\mathbb{E}\left[\overline{m}(\mathcal{C})\right] = \frac{p_{\text{out}}}{2}\left(n(n-s)\right) \qquad \text{and} \qquad \mathbb{E}\left[m(\mathcal{C})\right] = \frac{p_{\text{in}}}{2}\left(n(s-1)\right),$$

and accordingly for coverage and performance

$$\mathbb{E}\left[coverage(\mathcal{C})\right] = \frac{(s-1)p_{\text{in}}}{(s-1)p_{\text{in}} + (n-s)p_{\text{out}}}$$
$$1 - \mathbb{E}\left[performance(\mathcal{C})\right] = \frac{(n-s)p_{\text{out}} + (1-p_{\text{in}})(s-1)}{n-1}.$$

In the following, we can assume that for our randomly generated instances the initial clustering has the expected behavior with respect to the indices considered.

[1] In case a graph generated that way is not connected, additional edges combining the components are added.

4.2 Technical Details of the Experiments and Implementation

For our experiments, randomly generated instances with the following values of (n, s, v) respectively $p_{\text{in}}, p_{\text{out}}$ are considered. We set $v = 4$ and choose s uniformly at random from $\left\{\frac{n}{\ell} : 2 \leq \ell \leq \sqrt{n}\right\}$. Experiments are performed for $n = 100$ and $n = 1000$. On one hand, all combinations of probabilities p_{in} and p_{out} at a distance of 0.05 are considered. On the other hand, for two different values $p_{\text{in}} = 0.4$ and $p_{\text{in}} = 0.75$, p_{out} is chosen such that the ratio of $\overline{m}(\mathcal{C})$ and $m(\mathcal{C})$ for the initial clustering $\mathcal{C}$ is at most 0.5, 0.75 respectively 0.95.

The free parameters of the algorithms are set to $e = 2$ and $r = 2$ in MCL, $\alpha^* = 0.475$ and $\alpha^* = 0.25$ in ICC, and dimension $d = 2$ in GMC. As objective function *quality* in GMC, *coverage*, *performance*, intra-cluster conductance α, inter-cluster conductance δ, as well as the geometric mean of *coverage*, *performance* and δ is considered [2].

All experiments are repeated at least 30 times and until the maximal length of the confidence intervals is not larger than 0.1 with high probability. The implementation is written in C++ using the GNU compiler g++(2.95.3). We used LEDA 4.3[3] and LAPACK++[4]. The experiments were performed on an Intel Xeon with 1.2 ($n = 100$) and 2.4 ($n = 1000$) GHz on the Linux 2.4 platform.

4.3 Computational Results

We concentrate on the behavior of the algorithms with respect to running time, the values for the initial clustering in contrast to the values obtained by the algorithms for the indices under consideration, and the general behavior of the algorithms with respect to the variants of random instances. In addition, we also performed some experiments with grid-like graphs.

Running Time. The experimental study confirms the theoretical statements in Section 3 about the asymptotic worst-case complexity of the algorithms. MCL is significantly slower than ICC and GMC. Not surprisingly as the running time of ICC depends on the number of splittings, ICC is faster for $\alpha^* = 0.25$ than for $\alpha^* = 0.475$. Note that the coarseness of the clustering computed by ICC results from the value of α^*.

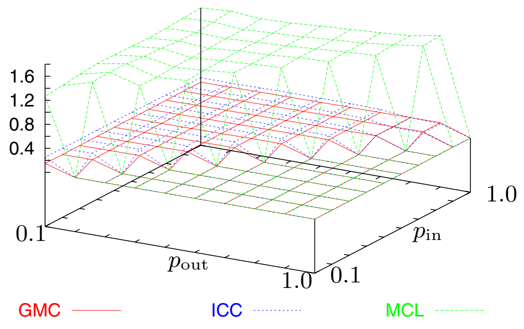
For all choices of *quality* except intra-cluster conductance, GMC is the most efficient algorithm. Note that the higher running time of GMC with *quality* set to intra-cluster conductance is only due to the elaborate approximation algorithm for the computation of the intra-cluster conductance value. In summary, GMC with *quality* being the geometric mean of *coverage*, *performance* and inter-cluster conductance, respectively *quality* being an appropriate combination of those indices is the most efficient algorithm under comparison. See Figure 1.

[2] Experiments considering the geometric mean of all four indices showed that incorporation of intra-cluster conductance did not yield significantly different results. We therefore omit intra-cluster conductance because of efficiency reasons.

[3] `http://www.algorithmic-solutions.com`

[4] `http://www.netlib.org/lapack/`

a)



b)

$(p_{\text{in}}, p_{\text{out}})$	GMC	ICC
$(0.25, 0.25)$	71	102
$(0.50, 0.25)$	72	103
$(0.50, 0.50)$	72	73
$(0.75, 0.25)$	74	101
$(0.75, 0.50)$	74	78
$(0.75, 0.75)$	74	73

Fig. 1. Running-time in seconds for $n = 100$ (a) and $n = 1000$ (b).

Indices for the Initial Clustering. Studying *coverage*, *performance*, intra- and inter-cluster conductance of the initial clustering gives some useful insights about these indices. Of course, for *coverage* and *performance* the highest values are achieved for the combination of very high p_{in} and very low p_{out}. The *performance* value is greater than the *coverage* value, and the slope of the *performance* level curves remains constant while the slope of the *coverage* level curves decreases with increasing p_{in}. This is because *performance* considers both, edges inside and non-edges between clusters, while *coverage* measures only the fraction of intra-cluster edges within all edges.

The fluctuations of the inter-cluster conductance values for higher values of p_{out} can be explained by the dependency of inter-cluster conductance $\delta(\mathcal{C})$ from the cluster $C_i \in \mathcal{C}$ maximizing ϕ. This shows that inter-cluster conductance is very sensitive to the size of the cut induced by a single small cluster. Due to the procedure how instances are generated for a fixed choice of n, the initial clustering often contains one significantly smaller cluster. For higher values of p_{out}, this cluster has a relatively dense connection to the rest of the graph. So, in many cases it is just this cluster that induces the inter-cluster conductance value.

In contrast to the other three indices, intra-cluster conductance shows a completely different behavior with respect to the choices of p_{in} and p_{out}. Actually, intra-cluster conductance does not depend on p_{out}.

Comparing the Algorithms. A significant observation when comparing the three algorithms with respect to the four indices regards their behavior for dense graphs. All algorithms have a tendency to return a trivial clustering containing only one cluster, even for combinations of p_{in} and p_{out} where p_{in} is significantly higher than p_{out}. This suggests a modification of the algorithms to avoid trivial clusterings. However, for ICC such a modification would be a significant deviation from its intended procedure. The consequences of forcing ICC to split even if the condition for splitting is violated are not clear at all. On the other hand, the approximation guarantee for intra-cluster conductance is no longer maintained if ICC is prevented from splitting even if the condition for splitting is satisfied. For MCL it is not even clear how to incorporate the restriction to non-trivial clusterings. In contrast, it is easy to modify GMC such that only non-trivial clusterings are computed. Just the maximum and the minimum threshold values τ are ignored.

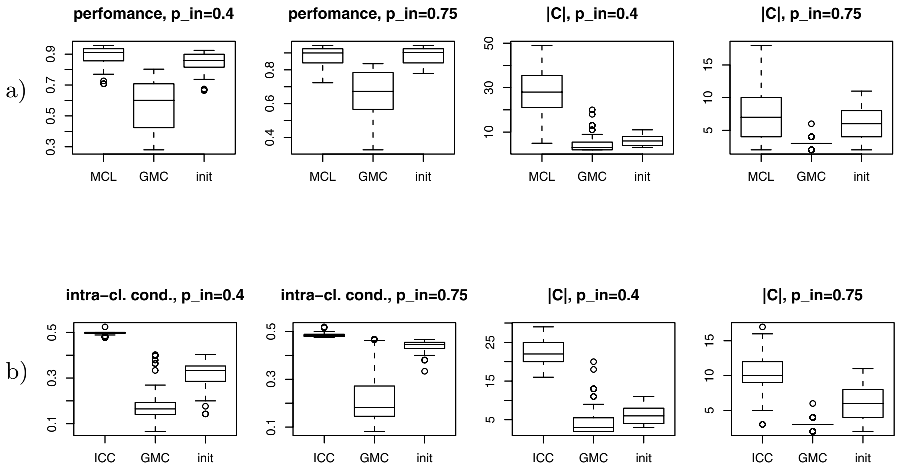


Fig. 2. The diagrams show the distribution of performance respectively intra-cluster conductance and the number of clusters for $p_{\text{in}} = 0.4$ respectively $p_{\text{in}} = 0.75$, and p_{out} such that at most one third of the edges are inter-cluster edges. The boxes are determined by the first and the third quantile and the internal line represents the median. The shakers extend to 1.5 of the boxes' length (interquartile distance) respectively the extrema. The first two diagrams in 2a) compare the performance values for MCL, GMC and the initial clustering, whereas the last two compare the number of clusters. The first two diagrams in 2b) compare the intra-cluster conductance for MCL, GMC and the initial clustering, whereas the last two compare the number of clusters.

Regarding the cluster indices, MCL does not explicitely target on any of those. However, MCL implicitly targets on identifying loosely connected dense subgraphs. It is argued in [4] that this is formalized by *performance* and that MCL actually yields good results for *performance*. In Figure 2a), the behavior of MCL and GMC are compared with respect to *performance*. The results suggest that MCL indeed performs somewhat better than GMC. The *performance* values for MCL are higher than for GMC and almost identical to the values of the initial clustering. However, MCL has a tendency to produce more clusters than GMC and actually also more than contained in the initial clustering. For instances with high p_{in}, the results for MCL almost coincide with the initial clustering but the variance is greater. ICC targets explicitely at intra-cluster conductance and its behavior depends on the given α^*. Actually, ICC computes clusterings with intra-cluster conductance α close to α^*. For $\alpha^* = 0.475$, ICC continues the splitting quite long and computes a clustering with many small clusters. In [3] it is argued that *coverage* should be considered together with intra-cluster conductance. However, ICC compares unfavorable with respect to *coverage*. For both choices of α^*, the variation of the *performance* values obtained by ICC is comparable while the resulting values are better for $\alpha^* = 0.475$. This suggests that besides intra-cluster conductance, ICC implicitly targets at *performance* rather than at coverage. Comparing the performance of ICC (with $\alpha^* = 0.475$) and GMC with respect to intra-cluster conductance suggests that ICC is much superior to GMC. Actually, the values obtained by ICC are very similar to the intra-cluster conductance values of the initial clustering. However, studying the number of clusters generated shows that this is achived at the cost of generating many small clusters. The number of clusters is even significantly bigger than in the initial clustering. This suggests the conclusion that targeting at intra-cluster conductance might lead to unintentional effects. See Figure 2b). Finally, Figure 3 confirms that ICC tends to generate clusterings with many clusters. In contrast, GMC performs very well. It actually generates the ideal clustering.

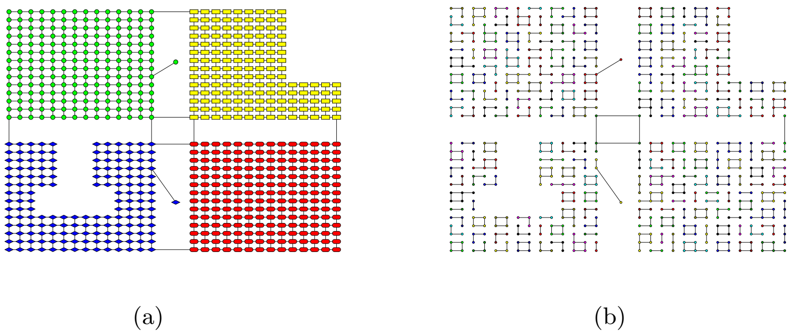


Fig. 3. In 3(a) the clustering determined by GMC for a grid-like graph is shown. The clusters are shown by the different shapes of vertices. In contrast, 3(b) shows the clustering determined by ICC. Inter-cluster edges are not omitted to visualize the clusters.

5 Conclusion

The experimental study confirms the promising expectations about MCL, i.e. in many cases MCL seems to perform well. However, MCL often generates a trivial clustering. Moreover, MCL is very slow. The theoretical result on ICC is reflected by the experimental study, i.e., ICC computes clusterings that are good with respect to intra-cluster conductance. On the other hand, there is the suspect that the index intra-cluster conductance does not measure the quality of a clustering appropriately. Indeed, the experimental study shows that all four cluster indices have weaknesses. Optimizing only with respect to one of the indices often leads to unintended effects. Considering combinations of those indices is an obvious attempt for further investigations. Moreover, refinement of the embedding used by GMC offers additional potential. So far, only the embedding canonically induced by the eigenvectors is incorporated. By choosing different weightings for the distances in the different dimensions, the effect of the eigenvectors can be controlled. Actually, because of its flexibility with respect to the usage of the geometric clustering and the objective function considered, GMC is superior to MCL and ICC. Finally, because of its small running time GMC is a promising approach for clustering large graphs.

References

1. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall (1988)
2. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys **31** (1999) 264–323
3. Kannan, R., Vampala, S., Vetta, A.: On Clustering — Good, Bad and Spectral. In: Foundations of Computer Science 2000. (2000) 367–378
4. van Dongen, S.M.: Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht (2000)
5. Harel, D., Koren, Y.: On clustering using random walks. Foundations of Software Technology and Theoretical Computer Science **2245** (2001) 18–41
6. Hartuv, E., Shamir, R.: A clustering algorithm based on graph connectivity. Information Processing Letters **76** (2000) 175–181
7. Spielman, D.A., Teng, S.H.: Spectral partitioning works: Planar graphs and finite element meshes. In: IEEE Symposium on Foundations of Computer Science. (1996) 96–105
8. Chung, F., Yau, S.T.: Eigenvalues, flows and separators of graphs. In: Proceeding of the 29th Annual ACM Symposium on Theory of Computing. (1997) 749
9. Chung, F., Yau, S.T.: A near optimal algorithm for edge separators. In: Proceeding of the 26th Annual ACM Symposium on Theory of Computing. (1994) 1–8
10. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation – Combinatorial optimization problems and their approximability properties. Springer-Verlag (1999)
11. Wagner, D., Wagner, F.: Between Min Cut and Graph Bisection. In Borzyszkowski, A.M., Sokolowski, S., eds.: Lecture Notes in Computer Science, Springer-Verlag (1993) 744–750
12. Garey, M.R., Johnson, D.S., Stockmeyer, L.J.: Some simplified NP-complete graph problems. Theoretical Computer Science **1** (1976) 237–267
13. Gaertler, M.: Clustering with spectral methods. Master's thesis, Universität Konstanz (2002)
14. Zahn, C.: Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Transactions on Computers **C-20** (1971) 68–86
15. Chung, F.R.K.: Spectral Graph Theory. Number 52 in Conference Board of the Mathematical Sciences. American Mathematical Society (1994)

Appendix 7.2

Dynamic Analysis of the Autonomous System Graph[*]

Marco Gaertler
University of Karlsruhe, Faculty of Informatics
E-mail: `gaertler@ira.uka.de`

Maurizio Patrignani
Università di Roma Tre
E-mail: `patrigna@dia.uniroma3.it`

Abstract

In this paper we investigate to what extent the information provided by BGP routing tables about the graph of the Autonomous Systems (ASes) can be used to understand dynamic phenomena occurring in the network. First, we classify the time scales at which such an analysis can be performed and, consequently, the kinds of phenomena that could be anticipated. Second, we improve cutting-edge technologies used to analyze the structure of the network, most notably spectral methods for graph clustering, in order to be able to analyze a whole sequence of consecutive snapshots that capture the temporal evolution of the network. Finally, we use such tools to analyze the data collected by the Oregon RouteViews project [20] during the last few years. We confirm stable properties of the AS graph, find major trends and notice that events occurring on a smaller time-frame, like worm-attacks, misconfigurations, outages, DDoS attacks, etc. seem to have a very diverse degree of impact on the AS graph structure, which suggests that these techniques could be used to distinguish some of them.

1 Introduction

Several researchers have started studying the Internet and its properties. In fact, such a rich and dynamic environment can be analyzed with respect to both the adjacency relationships between the entities composing it and the complex information flowing through its links and stored in its nodes. Often, this field of research produces intriguing results. For example, self-similarity laws, which are usually found when studying natural or biological phenomena, were used to explain the apparently chaotic behavior of traffic loads or the uneven distribution of the number of adjacencies between the network nodes.

Recently, the general attention focused on the Autonomous Systems (ASes), the independent organizations whose cooperation guarantees the delivery of the packets through the network. Since each AS exchanges traffic flows with some neighbors (BGP peers), and local and remote adjacencies can be gathered from the logs of the BGP conversations between peers, plenty of current and historic data is potentially available about the ASes, their peerings, and the IP prefixes that are contained inside them. This kind of information is used to produce reliable statistics on the scalability of Internet technology and on the state of health of global routing [17].

Usually, the AS graph is reconstructed by merging information collected by a number of repositories managed by private and public research organizations around the world. Several studies have the purpose of analyzing the static properties of the AS graph, including the kind of relationships between its nodes, the distribution of their degrees, or other graph-theoretic measures [14, 3, 10, 12, 25, 22].

In this paper we investigate to what extent the information available from the BGP repositories can be used to understand the dynamic phenomena taking place in the Internet. Since, to our knowledge, this is the first attempts to systematically study dynamic properties of the AS graph, we begin by classifying the timescales at which such observations can be carried out and, consequently, the kinds of phenomena which may be anticipated, namely, stability, major trends, and spot events. Second, we consider the cutting-edge technologies described in the literature to analyze the structure of the AS graph, most notably spectral methods for graph clustering, and we advance and modify them in order to be able to analyze not only a single snapshot, but a sequence of consecutive AS graphs capturing the temporal evolution of the network. Finally, we use such tools to analyze the data collected by the Oregon RouteViews project [20] during the last few years, confirming known results and sensible hypotheses about the stable properties of the Internet, measuring how the network is

[*]Work partially supported by European Commission - Fet Open project COSIN - COevolution and Self-organisation In dynamical Networks - IST-2001-33555, by European Commission - Fet Open project DELIS - Dynamically Evolving Large Scale Information Systems - Contract no 001907, by "Progetto ALINWEB: Algoritmica per Internet e per il Web", MIUR Programmi di Ricerca Scientifica di Rilevante Interesse Nazionale, and by "The Multichannel Adaptive Information Systems (MAIS) Project", MIUR Fondo per gli Investimenti della Ricerca di Base.

evolving, and showing how this approach may turn out to be a valuable tool for monitoring the network against some types of malicious attacks. In fact, the various kinds of transient phenomena affecting the Internet seem to have a very different magnitude of impact on our measures, from none at all (worm attacks, misconfigurations, etc.) to a significant one (DDoS against DNS servers), offering a way to distinguish them.

The paper is organized as follows: Section 2 describes the motivations that prompted our research. Section 3 provides the necessary background and describes our methodological contribution to this domain of research. Section 4 contains the results of our short- and long-term analysis of the last few years of networks evolution and incidents. The conclusions complete the paper.

2 Motivations

Data retrieved from BGP repositories is generally analyzed to determine the structure and properties of the network of the Autonomous Systems at a specific time [13, 14, 3, 10, 12, 25]. However, a more sophisticated investigation can only be accomplished by considering a whole sequence of snapshots taken at subsequent moments. This is particularly true when the properties to be analyzed are related to temporal aspects, like stability, growth, or the dynamics of transitory phenomena. This kind of analysis can be performed with different time granularity, addressing different needs:

Long-term analysis Typically, the measures of the AS graph are not constant, but heavily affected by noise, rapid fluctuations, and sudden changes. A more stable view, obtained by considering a sufficiently long time window, would allow us to decide whether a particular snapshot actually fits into the 'average' network behavior, to understand if a peculiar measure captures a persistent property of the network or if it must be ascribed to fortuitous fluctuations, and to spot slow phenomena that take place over a period of months or even of years. Combining acquired knowledge about long-term temporal behavior allows to identify current trends and helps to ensure that the network continues to work properly.

Short-term analysis: The accessibility of detailed BGP logs offers the opportunity to perform short-term analysis, that is to investigate changes occurring in a short time period (and sometimes in a small neighborhood) with the purpose of classifying them as noise, incidents, outages, misconfigurations or malicious attacks. Of course, an effective classification would allow us to detect anomalies that occurred in the past, but the objective of this research may be much more ambitious: devise techniques to spot anomalies as they occur, by means of a preemptive monitoring of the real-time and near-real-time behavior of the network.

3 Methodology

In this section we give an overview of the terminology we use, the data collection process, and our techniques.

3.1 Terminology

As usual, we build the graph of the Autonomous Systems starting from a set P of AS paths gathered from BGP logs. Each AS path is associated with a prefix (an interval of IP addresses) and consists of the sequence of the numbers of the Autonomous Systems traversed by the traffic to be delivered to the associated prefix. The first number in the sequence is called *vantage point* and is the source that recorded the AS path, while the last number is the *announcing AS* hosting the prefix IP numbers. In order to preserve information about vantage points and their relative importance we enrich the AS graph $G(V, E)$ with a mapping $\phi\colon V \times E \longrightarrow [0, \infty[$, where:

- each AS is represented by a node,
- there exists an undirected edge between two nodes, if there exists a path in P in which the corresponding AS numbers are consecutive, and
- the number $\phi(v, e)$ equals the number of different paths in P that have v as vantage point and contain e.

This model allows us to keep track of the edge set seen by each vantage point. AS graphs are very difficult to compare, even if built from data sets taken at short intervals of time. In fact, the very vantage points collecting data are often temporarily unavailable, inducing significant changes in the observed portion of the network. Therefore, it is sometimes desirable being able to consider the stable, common part of two subsequent AS graphs. We define the *commonly observed AS graph* of two AS graphs G_1 and G_2 (with attached mappings ϕ_1 and ϕ_2) as the graph G_c where an edge $\{u, v\}$ is introduced if it has a value different from zero for both $\phi_1(v, \{u, v\})$ and $\phi_2(v, \{u, v\})$ for some node v. If needed, the value of $\phi_c(v, \{u, v\})$ is chosen to be the minimum of the two values above. Isolated nodes are removed from G_c.

3.2 Data Collection

Data on the actual routing can be collected by logging the BGP routing tables. Unfortunately, some well-known testbeds available on the web, e.g. [2], do not provide the needed granularity with respect to time. In order to have the needed width and density, we used data collected by the *Oregon RouteViews Project* ([20]). This project was especially founded to grant real-time information about the global routing system from the perspectives of several different backbones and locations around the Internet. Its data is widely used by other network researchers and operators. It started to operate on April 20th, 2001 and provides snapshots every two hours. The provided data is not purified and contains misleading information, like repetitions of paths, loops inside paths, or prepending of ASes. Further, other issues can alter data, like truncated data files, time-outs, lost connections, or connections closed too early. We therefore filtered data according to the following simple rules:

1. discard any data set that is truncated or shows signs of technical problems, like the IP space is too limited,
2. enumerate all paths only once, and
3. delete loops and prepending ASes.

3.3 Observed Indices

A common approach is to define or use indices that provide a succinct measure of the properties of the graph which are to be monitored. We consider the number of nodes, edges, observing peers and paths. The first two are the classical graph theoretic measures and give a rough measure of the size and the density of the network. The other two are specific to this domain. They are needed to reflect the various influences on the extracted network view. So far, the total number of BGP-peerings between ASes currently present in the Internet is unknown. It is only known that a larger set of vantage points often implies a larger set of paths. More paths often also result in more edges. Unfortunately, the peer and path sets are affected by various fluctuations themselves. Such changes can be roughly classified as: Problems due to the collecting software's faults, connectivity problems that affected the collection process, and actual changes in the routing graph. The first two may lead to false conclusions and therefore need to be removed. Most of these events can be eliminated by an online cleaning process. Other indices that capture structural properties are the maximum core level and the size of the induced core graph (both will be introduced in Section 3.4.1) . Besides this static indices, we also considered the similarity of consecutive snapshots, by considering the commonly observed AS graph and its size.

3.4 Abstract Views

In order to cope with the amount of data, we had to compute a very concise description. We used different kinds of simplification and clustering techniques, which offer the opportunity to get highlevel views of the network structure and behavior.

3.4.1 Cleaning the Graph

When dealing with large amounts of data, a usual technique is filtering out irrelevant information. Several authors attempted to investigate relevance concepts for the AS graph. The most common is the degree of a node, used for example by Govindan and Reddy [16], Gao [14] and Tauro et. al. [25]. We used the concept of coreness which is strictly related to the degree of nodes and was introduced in [23] and [5]. The *k-core* of a graph is defined as the unique subgraph obtained by recursively removing all nodes of a degree less than k. A node has *coreness* value ℓ, if it belongs to the ℓ-core but not to the $(\ell+1)$-core. Coreness can be used to filter out peripheral ASes. A suitable value of k can be chosen, for example, based on the number of the remaining ASes. There are no evidences that the scale-free property of the AS graphs might influence this coreness-based cleaning. The coreness concept can be used to generate a reduced model of the graph. We define the *core graph* $G_{\text{core}}(H) = (V', E')$ of a graph $H = (V, E)$ as follows. Let $E_{i,j}$ be the subset of all edges (in H) incident to nodes of coreness i and j. In $G_{\text{core}}(H)$ there is a node for each distinct coreness value, and two nodes i and j are connected, if $E_{i,j}$ is not empty. An edge-weighting function w of H can be transformed into an edge weight w' of G_{core} by setting $w'(i, j)$ to the total weight of all edges in $E_{i,j}$.

The AS graph has a very special structure, i.e. it is the result of merging paths. This composition reflects fundamental properties and thus a good filtering technique should respect the inherent path properties. The quality of a filtering technique could be measured by the number of AS paths that remain connected in the reduced graph. For the k-core filtering we define $\mu_k(i)$ to be the fraction of paths that have i connected components in the k-core. This number depends on the input parameter k. In order to avoid such a dependency and increase the insight into its compatibility, we will consider lower bounds for $\mu_k(1)$. Such a bound is given by the fraction of paths that remain connected in all k-cores and is denoted by $\mu(1)$. On the other hand, we also like to judge the degree of fragmentation. Therefore we count the fraction of paths that would be cut into i components in a k-core for some value k and $i > 1$. This value, which is denoted by $\mu(i)$, gives an upper bound on $\mu_k(i)$. An experimental validation on the coreness is presented in Section 4.1.1.

3.4.2 Clustering

Clustering is a well-known technique to explore the inner structure of relationships and the roles of the participating entities. It was already used for this purpose in research areas such as social networks [6, 11], data mining [18], and VLSI [4]. Gkantsidis et. al. [15] performed spectral analysis and clustering on the AS graph. As part of their preprocessing they delete all nodes of degree one or two, which is equivalent to consider the 3-core. Other operations involved normalization or third-party information. Their main analysis is based on spectral information and involves eigenvectors. In this special scenario eigenvectors are mappings from the node set to the real numbers and proximity in these values often corresponds to dense subgraphs. They used a common heuristic to partition, which is based on selecting connected subgraphs that share the same sign in a specific eigenvector. By suitably choosing a subset of the eigenvectors, they were able to produce clusters characterized by strong geographic or business relationships. Good eigenvectors for this task are among those that correspond to the largest eigenvalues.

We applied a clustering method, called Geometric MST Clustering (GMC), that embodies the same intuition of the proximity provided by the eigenvectors and that was introduced in [8]. Although it involves more complex algorithmic steps, it can be fully automated, it explores the eigenvector structure to a larger extent, and requires only the graph (and optionally an edge weighting). The general idea is to interpret several eigenvectors, that are associated with the largest eigenvalues of the normalized adjacency matrix, as embedding for the graph and search within this geometric object for dense parts. The search itself is performed by a Minimum-Spanning-Tree (MST) computation. By considering only the span of tree edges that have (geometric) distance smaller than a given threshold, the MST represents a hierarchy of clusterings. Evaluating this hierarchy with different clustering indices allows to chose a good clustering as well as to incorporate several user-defined aspects.

The AS graph is quite inhomogeneous with respect to the density. The major part of the nodes are only loosely connected ([24]). In order to overcome this property and concentrate on relevant ASes, we choose the minimum k-core that has at most 200 elements. This choice is based on a rough estimate on the number of important ASes and experiments. As objective function for the clustering we used $\mathrm{quality}(\mathcal{C}) = \sqrt[4]{\mathrm{coverage}(\mathcal{C}) \cdot \mathrm{performance}(\mathcal{C})^3}$, where $\mathrm{coverage}$ is the ratio of numbers of edges within all clusters and the total number of edges; and $\mathrm{performance}$ is the number of all adjacent node pairs lying in the same cluster plus the number of all non-adjacent node pairs lying in a different cluster divided by the total number of node pairs. The $\mathrm{coverage}$ score is an indicator for the global density of a clustering. While $\mathrm{performance}$ rates the clustering with respect to the disjoint-clique model. We chose this objective function to focus more on structural issues than on pure density. Again experiments verified the usability of this objective function. Some properties of the resulting clusterings are presented in Section 4.1.2.

The integration of dynamic aspects, like the clustering of a sequence of graphs, entails lots of problems. First of all, the result format is not clear. Is a single partition of the node set still required or for each point of time? What is the importance ratio between edge set and time horizon? Second, how can static algorithms be adapted? And finally what should be the relation between static and dynamic clusterings? In the few instances such problems already arose, the following two methods were used to reduce the dynamic case to the static one, thus always calculating a single partition of the node set. The first one collapsed all graphs into one. Edges are present if their frequency is large enough. This corresponds to consider the 'average' graph over time and cluster it. The second approach clusters each graph individually and combines the results afterwards to a single partition. In this case, the 'average' clustering is output. Both versions have their disadvantages; the major problem is to find suitable thresholds for the combination step. We will use the second approach because it allows us to observe node movement more easily. However, statistical disturbances are the major source of false conclusions for both methods, especially when the given time frame is very short.

4 Experiments and Results

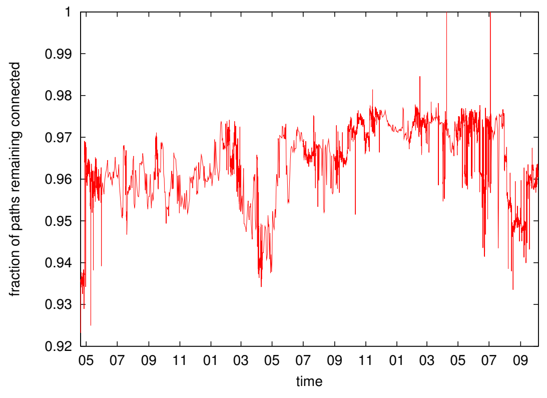
Basic properties and the quality of our auxiliary reduction techniques are presented first. This is necessary to increase the understanding of the impact of these utilities and to avoid wrong conclusions. It is followed by the major elements of dynamic analysis – baselines, trends and anomalies.

4.1 Evaluation of Reduction Techniques

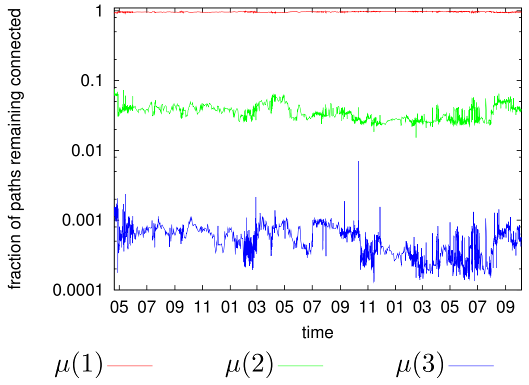
The reduction techniques consist of cleaning and filtering on the one side and clustering on the other. The major difference is that the first keeps atomic elements – nodes still represent single ASes – while clustering produces many non-singleton groups. In this first part we present some considerations that justify the use of the reduction techniques in the later sections.

4.1.1 Validation of Cleaning and Filtering

It can be observed that $\mu(1)$ always exceeded 0.9, thus more than 90% of all AS paths remained connected in every k-core. Furthermore, only $\mu(2)$ and $\mu(3)$, which are the upper bounds on the fraction of paths that are split into two and three components, have values that are significantly larger than zero. Figure 1 shows the temporal evolution.



(a) Lower bound $\mu(1)$



(b) Temporal evolution of $\mu(i)$ for $i = 1, 2, 3$

Figure 1. Fraction of paths that are guaranteed to be connected and are split into two or three components over time

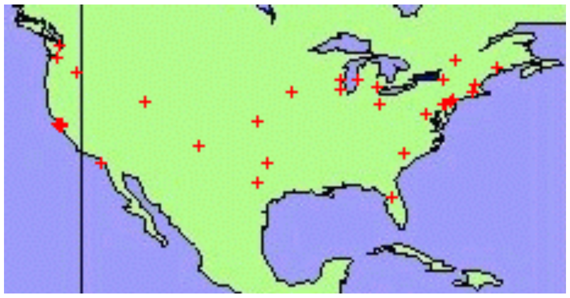
These observations are good indicators that the core structure of the graph is compatible with the path structure in general and therefore may be used for filtering purposes without altering the graph structure too much.

4.1.2 Significance of Clustering

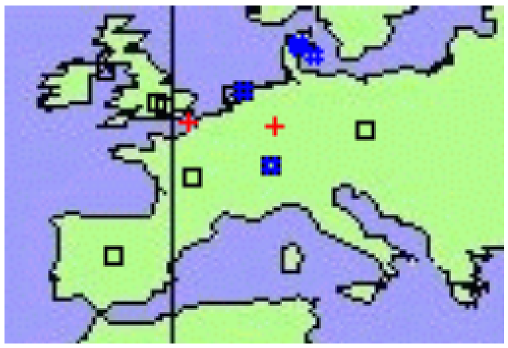
We selected several random snapshots and clustered them individually. The observed features were averaged, used as hypotheses for common properties, and verified against other randomly selected samples. Thus, we found the following characteristics:

- two to six clusters have more than four elements; these clusters cover together more than 70% of all elements
- the remaining clusters often contained only one element
- clusters reflect geographical issues, i.e. the major clusters separated US, Europe and Asia

The Cooperative Association for Internet Data Analysis (CAIDA) provides limited geographic positions for several ASes. These may include coordinates or affiliation to certain states. The relationship between clusters and this data is shown in Figure 2 and Table 1. Thus, clusters reflect actual coherences of ASes as well as geographical issues. Therefore it might be useful to detect and classify changes in the network structure.



(a) United States of America



(b) Europe

Figure 2. Geographical positions of some ASes. Different shapes represent different clusters, i.e. crosses for cluster 1 (33/86 positions available), squares for cluster 2 (10/17 positions available), and sharps for cluster 3 (4/6 positions available).

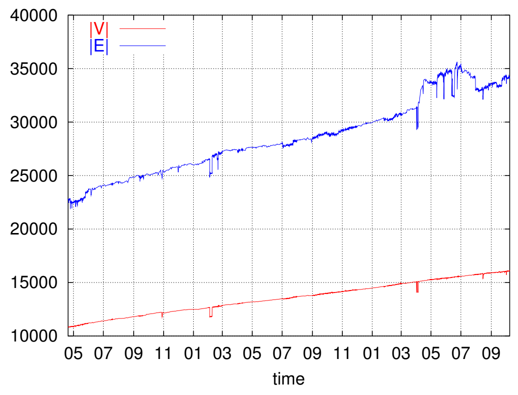
	Clusters			
Country	1.	2.	3.	remaining
US	75	3	1	22
Europe	3	10	4	24
Africa	2	1	1	2
Asia	5	1	-	2
unknown	1	2	-	2

Table 1. Distribution of countries on clusters

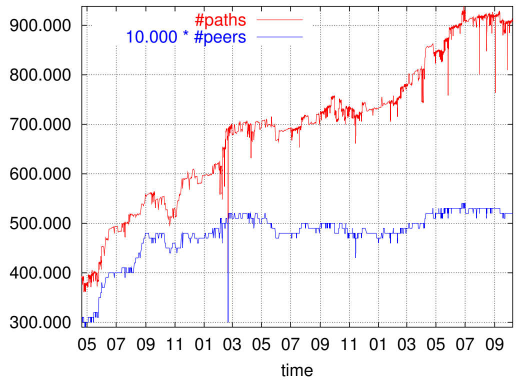
4.2 Baselines and Trends

Determining the standard undisturbed behavior is one fundamental task of dynamic analysis. As an initial step we investigate the evolution of static indices. The temporal evolution is shown in Figure 3(a) and 3(b). The number of nodes and edges seem to be locally stable over time. In order to verify this observation, we calculated the standard deviations of time frames of different length. Both indices seem to be non-constant, therefore a larger time window results in a larger deviation. In Figure 4 the time frame length is plotted versus the average standard deviation (ASD). As expected the ASDs are isotonic in the frame length, furthermore their increases can be expressed by linear function, thus verifying the local stability. The slope for the number of edges is three times larger than for the number of nodes. This can be explained by the fact, that edges are subject to more consistent changes and depend on the number of paths which in turn depend on the number of participating peers. For example, number of edges and paths as well as number of paths and peers are highly correlated (≈ 0.977 and ≈ 0.897, respectively). In this special scenario, we can utilize the path structure to normalize the views by using the commonly observed AS graph of two consecutive AS graphs. The size of the node set and the edge set will be denoted by $|V'|$ and $|E'|$. This enables us to decide whether a change in the number of nodes (or edges) is only due to a change in the vantage points or not. Figure 5(a) displays the evolution of the number of nodes and edges for the AS graphs and the commonly observed AS graphs. Utilizing the detailed view of Figure 5(b) we conclude that the first two increases (2am and 10am) of peers led to a larger view (new edges were added) while the third increase (2pm) did not effect it (only redundant information was added). However, we have to be careful judging decreases, since the absence of vantage points can have many reasons. Some of these causes are independent of the (global) network status, like internal reorganization or connection failures during the collection process.

Another approach to judge the undisturbed behavior of the network involves a more structural and high-level view: the core graph (Section 3.4.1). We extend the concept by adding a node-weighting function that represents the fraction of nodes (in the original graph) having a certain coreness. We observed that most nodes have a very low coreness score (one, two, or three). When edges are uniformly weighted, there are two different types of edges with large weight: edges that connect cores with a small index and edges that connect low-index cores with high-index cores (*transition edges*). In the case where edges are weighted according to the number of paths that use them, transition edges and edges connecting cores with large index are present. Figure 6 shows such a core graph with the two different edge weightings. This reflects the general intuition, that many links are needed to connect the customer to their provider and that the links that occur more frequently in paths are links to or within the 'backbone' and not in-



(a) Number of nodes and edges



(b) Number of AS paths and peers

Figure 3. Graph theoretic and domain specific measures. The x-axis represents time, starting in May 2001 till September 2003.

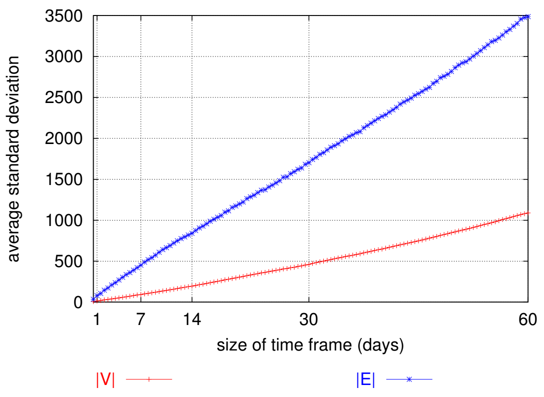
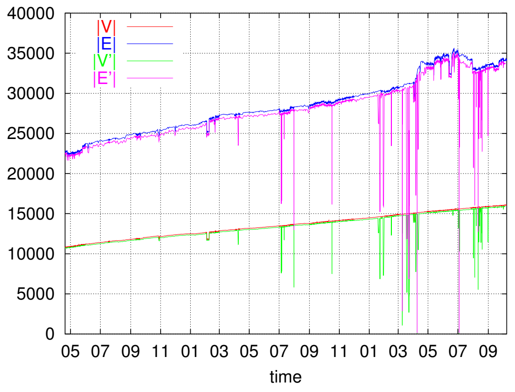
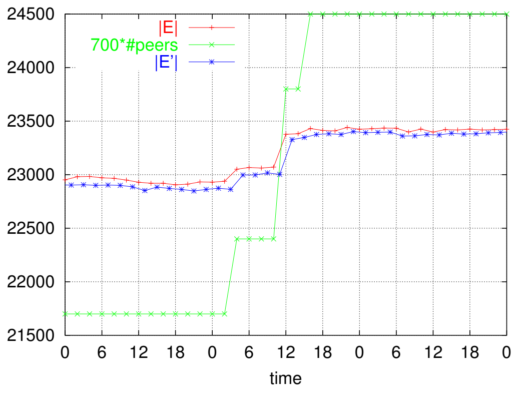


Figure 4. Local stability of number of nodes and edges. The time frame length is plotted versus the average standard deviations.



(a) Evolution over time of the different indices



(b) Detailed view (2001 May from 24th till 27th).

Figure 5. Sizes of AS graphs and commonly observed AS graphs.

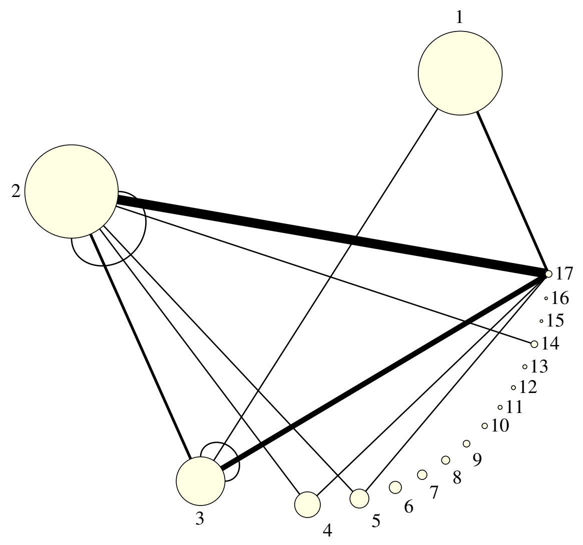
side the periphery. This description is stable with respect to time. In Figure 7 the relative distribution of coreness values are drawn; from the diagram it is evident that fractions of nodes with the lowest and the highest coreness are stable.

Long-term Patterns

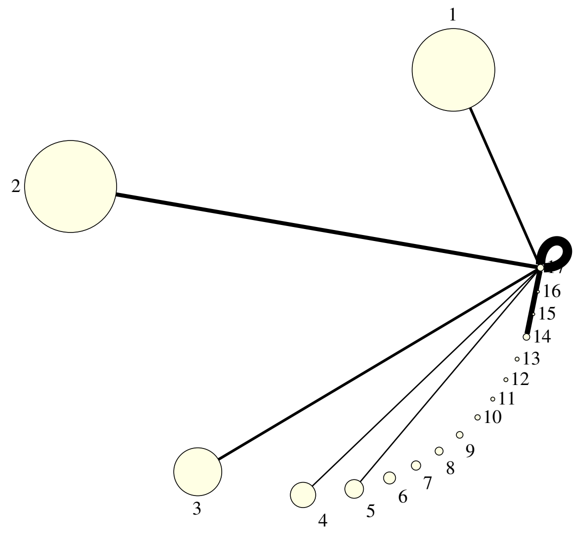
Using these observed phenomena, we recognized two major patterns. The first one is the linear growth of the number of nodes and edges. Although it seems to be a trivial observation (Figure 3(a)), it is not at all expected. Many researchers have reported an exponential growth in the time frame of 1997 till 2000. Because both indices grow in a linear fashion, the ratio of observed edges to the total number of possible edges decreases. The second pattern is the distribution of growth. ASes are separated according to their importance in the core graph. By inspecting the evolution of the core graph, we observed that most nodes enter the graph with a very low coreness score, while edges (in the path-weighted version) connect low-score nodes with high-score nodes or only high-score nodes with each other. This verifies the general intuition, that more customers than (high-level) providers enter the system and that more connections are established to connect the small providers with the backbone or enlarge the backbone.

4.3 Short-Term Analysis – Anomalies

In this section we describe the results of short-term analysis. Namely, we measure the impact on our measurements of dramatic short-lived events that occurred in the network in the last few years. This section also offers the opportunity to test the effectiveness of the techniques described in Section 3.4.2 to overcome the difficulties implied by the use of the clustering approach in a dynamic scenario on real-world data. As already mentioned, events that last very short are problematic. In these cases a higher granularity of the data would be needed for better observations, but is often absent. However, our measurements suggest that events occurring on a small time-window may be effectively classified with our clustered analysis. In particular, we show in this section that some kind of sporadic phenomena, like worm attacks or misconfigurations, even if they may have a dramatic effect from the user's point of view, do not seem to have a measurable impact on the AS graph structure. On the other hand, some very specific events, notably DDoS attacks against DNS root servers, happen to cause a significant change of the structure of the clusters. Thus, this kind of analysis may be a promising tool, to be used as a litmus-paper to test if one of these specific events is occurring in



(a) Uniform edge weight



(b) Edge weight according to used paths

Figure 6. Core graph of AS graph (May 1st, 2001 0:00). The area of the nodes is proportional to the number of nodes having that coreness score. The thickness of the edges is proportional to their weight. (Note, that edges with very little weight are omitted.)

the network.

BGP storm due to worm attack. On July 19th 2001, the `Code Red II` worm was spreading on the Internet exploiting the indexing service vulnerability in the Microsoft Internet Information Server MS-IIS ([9]). Although an exponentially growth in the advertisement rate was observed by [7], we could not assess a significant change in the number of nodes and edges. In fact, Figure 8 shows that the number of nodes and edges is quite stable while the drop in the number of peers accounts for the loss of paths. These missing peers - AS715 and AS19092 - usually contribute with more than 15,000 paths each. Using the clustering technique, we could not spot any significant changes. In fact, about 75% of all nodes were either perfectly stable or switched between two clusters, the other nodes blinked in and out of our observed view or could not be associated to one cluster. The number and sizes of the clusters were stable. A similar behavior could be observed when on September 18th 2001, a extremely virulent worm, called `W32.NIMDA`, spread throughout the Internet using multiple methods to inflect both Windows servers and user machines ([27]).

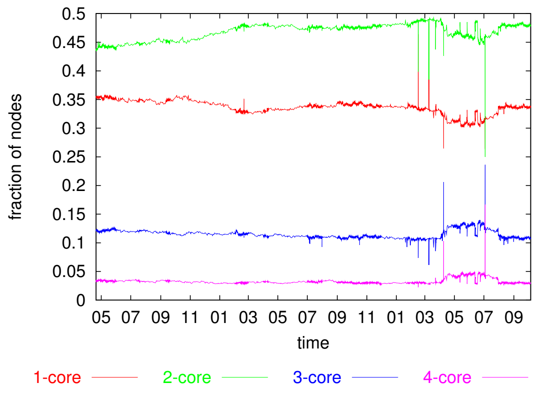
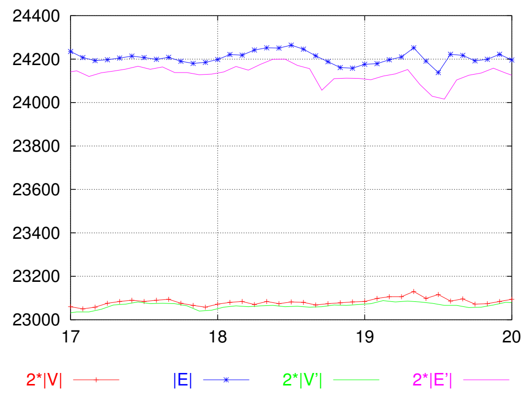
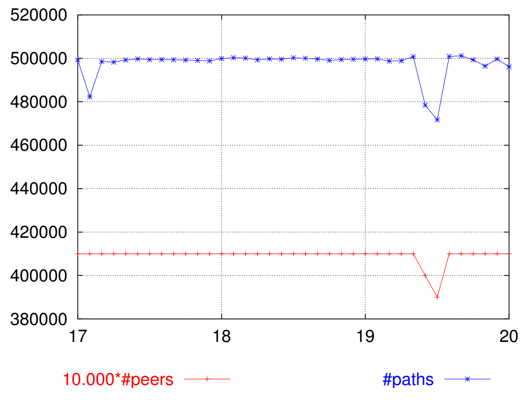


Figure 7. Distribution of coreness values (relative to the number of nodes in the graph)

Shutdown of KPNQwest/Ebone and RIPE NCC Test Traffic measures a 50% increase of alarms. In July 2002, KPNQwest, one of Europe's largest Internet backbone provider had a time-out. The company went offline on the 3rd and returned on the 25th ([19]). At the beginning of this event, the RIPE NCC measures a 50% increase of alarms on their TTMs (Test Traffic Measurement Boxes, [26]). We observed a drop in the number of peers, paths and edges. While the first two lasted only a few days, the last one was present during the whole period. Figure 9 shows this evolution. The clustering is rather stable when we considered the three time periods (before, during and after the shutdown) separately. However, we could observe temporal migrations (small subsets formed new clusters or were swallowed up), new ASes 'entered' the system and

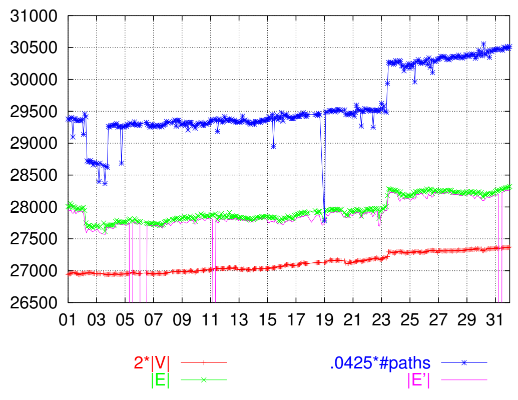


(a) Number of nodes and edges

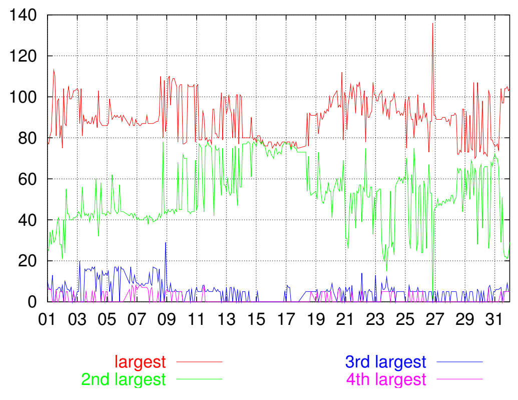


(b) Number of peers and paths

Figure 8. Evolution around July 19th 2001



(a) Number of nodes, edges and paths



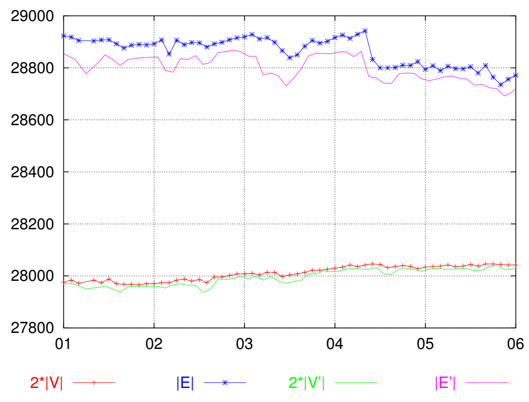
(b) Sizes of clusters (sorted according to size)

Figure 9. Evolution during July 2002

old ASes 'left' during the transitions. The few drops in the number of edges of the commonly observed AS graph (see Figure 9(a)) are probably due to technical issues and do not reflect changes.

Misconfiguration in UUNet. UUNet (WorldCom) stopped talking to the rest of the Internet on October 3rd 2002 between 12am and 9pm (UTC), due to misconfiguration in some of their routers. This caused increases of respond times, delays, and packet losses ([21]). It affected many other ASes, because UUNet carried half of the world's Internet traffic. Although the number of edges seems to be affected, we could not observe a significant change (see Figure 10).

DDoS Attacks against 13 Internet Root Servers. On October 21st 2002, a series of well-coordinated, simultaneous DDoS attacks were launched from various points around the world, against each of the 13 Root Servers that are used for the Internet's Domain Name System (DNS) ([21]). The attack, which disabled nine of the 13 Root Servers, started at 8:45pm (UTC) and lasted approximately two hours. We could only partially observe the event, because *Oregon Routeview* had a 6-hours blackout, due to connection time-outs, starting at October 22nd 2am (UTC) and the peer AS701 disappeared, causing a loss of $\approx 17,600$ paths earlier on the 21st (8am UTC). It never returned as a vantage point. Thus, we can only analyze the structure of the network before and during the attack, but not immediately afterwards. Also, the loss of paths limits the viewpoint. Similar to the previous worm attacks, the number of nodes and edges is stable (see Figure 11). The clusterings are very stable, except for the point in time of 21st 8pm. In that instant a kind of splitting occurred. A subset of 29 elements and a single node emerged from the



(a) Number of nodes and edges

Figure 10. Evolution around October 3rd 2002

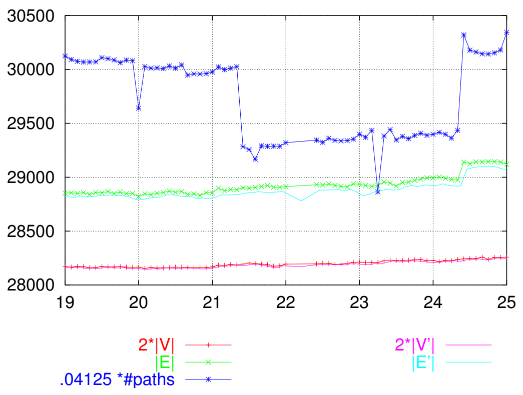


Figure 11. Evolution around October 21st 2002

largest cluster and built their own clusters. For simplicity we denote the remaining part of the largest cluster by 1' and the other two with 1" and 1"' respectively, according to their size. Table 2 contains the geographic distribution of the ASes located in those clusters. Most of the elements in 1" and 1"' were perfectly stable while belonging to cluster 1 during the other time frames. The four Asian ASes – AS2518, AS4143, AS4728, and AS4766 – are kind of exchange servers. The cluster 1" contains also several well-know ASes like AT&T, Cable and Wireless, former Genuity, Globalcrossing, Sprintlink, UUnet, Verio, Microsoft (three times) and Google. A more fascinating fact is that in each cluster 1' and 1" an unaffected or a less affected DNS root server (AS297 and AS3557) is present. This is a strong indicator that the clustering was affected. However, the degree of interaction remains open. The current granularity is not high enough for a further and more detailed analysis.

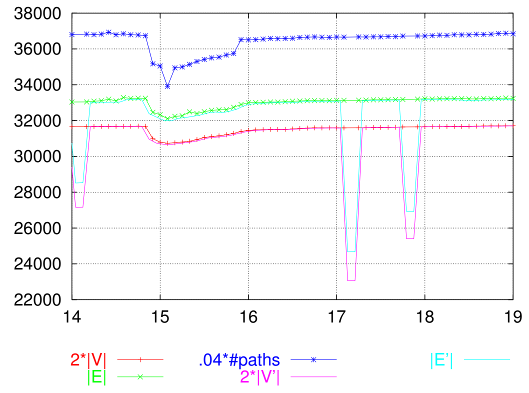
countries	clusters			
	1	1'	1"	1"'
US	74	48	26	-
Europe	9	9	-	-
Asia	11	7	3	1
Africa	1	1	-	-
Australia	1	1	-	-
unknown	1	1	-	-

Table 2. The distribution of countries of the cluster and its segmentation.

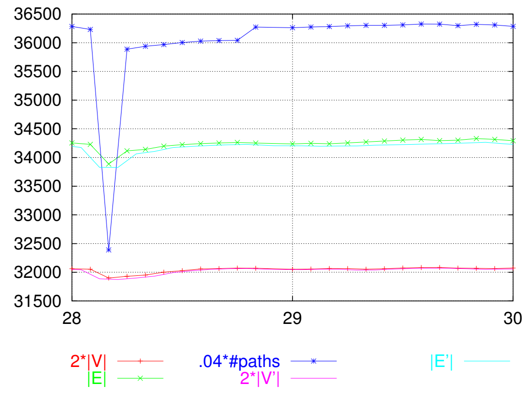
Sapphire worm attack. In the morning of 25th January 2003, the `Sapphire worm` (also known as `SQL slammer`) was released on the Internet. Exploiting a vulnerability in Microsoft SQL server it multiplied itself rapidly and soon spread out over networks worldwide. From news headlines and activity on mailing lists it was clear the attack had an impact on the Internet's performance. Similar to the *DDoS Attack* our data source *Oregon Routeview* had a blackout from 8am till 12am due to connection timeouts. Also two large peers - AS7660 and AS8297 - were absent on 26th. Each contributed with more than 16,000 paths. Thus our view point is very limited and can hardly be used to distinguish between worm attack and peer loss.

DDoS Attack on the RIPE NCC. Starting from 2pm (UTC) February 27th, 2003, the RIPE NCC network suffered a large DDoS attack (a distributed ICMP echo attack). Their network structure was affected not only ICMP traffic. Network condition returned back to normal the same day at 4:30pm UTC. Shortly before (2am till 2pm) the peer AS5459 did not contribute, causing a loss of $\approx 7,100$ paths. Nothing unusual could be observed.

Blackouts. During August and September 2003, several blackouts happened in different geographic regions. On August 14th, around 6pm (UTC) started the cascading chain of events that lead to a gigantic blackout in the north-eastern part of America and Canada (refer to [1] for details). A 40-minute blackout took place in London on August 28th (around 6 pm). The last one struck parts of France, Italy and Switzerland on September 28th. It started around 2:30 am and power returned during the afternoon. The event in London could not be observed, since it was too short. The other two exhibited a similar pattern: a



(a) US and Canada



(b) France, Italy and Switzerland

Figure 12. Evolution of different blackouts

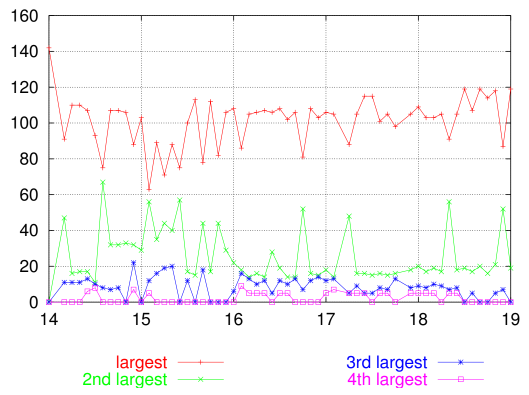


Figure 13. Sizes of the different clusters during the blackout in the US and Canada

large drop in the number of paths occurred at the start of the event. This drop caused also a drop in the static indices (number of nodes and number of edges). The recovery phase was very short and the indices quickly got back to their old scores. The drops in the number of nodes and edges of the commonly observed AS graph (Figure 12(a)) is not due to the blackout itself. They are results of the disruption in the collection process. The reduced data set was relatively stable. Most node were present more than 80% of the time. Only a small set (≈ 20 nodes) appeared only once or twice. The clustering itself is relatively stable. Figure 13 shows the temporal development of the sizes of the clusters. Smaller migrations, splittings or merging phenomena could be observed. Although there is no clear pattern or regularity apparent, it is a fact that these changes occurred more frequently during the recovery phase, thus giving a good indicator that clustering reflects aspects of the event. Similar observations could be made during the blackout in Europe. However, the impact on the structure was not comparable with respect to size and the recovery time much shorter.

5 Conclusions

Clustering and reduction techniques are well-known and widely-used approaches for analyzing large amounts of data. We adapted these tools to the domain of the dynamic analysis of the Internet graph at the AS level. Several experiments were performed to deepen the knowledge of short- and long-term phenomena. The first target was to describe the baseline behavior of the network in the absence of pathologic phenomena. Although the analysis was relatively plain, we could verify several intuitional properties. A second effort went in the direction of measuring the impact of short-lived events, such as viruses, misconfigurations and blackouts. We found, that viruses and worms had no measurable effect on the network structure, even if Internet services might have been disrupted. In contrast, blackouts and DDoS showed measurable consequences. Obviously, the analysis showed indisputable and precise evidences of blackouts and power outages, but the symptoms registered for DDoS are perhaps more promising from a practical point of view.

Although hampered by lots of technical problems, as the continuous changing of the available vantage points or the uncertain identification of the ever-changing clusters through time, we think that the dynamic analysis of the AS-graph is a promising field of research that could have many useful outcomes in the future. In this paper we made a first step towards the definition of a practical methodical approach to this intriguing problem.

References

[1] A timeline of the 2003 blackout. http://www.cnn.com/2003/US/08/16/blackout.chron.ap/index.html.

[2] Characterizing the internet hierarchy from multiple vantage points. http://www.cs.berkeley.edu/~sagarwal/research/BGP-hierarchy/.

[3] S. Agarwal, L. Subramanian, J. Rexford, and R. Katz. Characterizing the internet hierarchy from multiple vantage points. In *Proceedings of IEEE Infocom 2002*, 2002.

[4] Charles J. Alpert and Andrew B. Kahng. Recent directions in netlist partitioning: A survey. *Integration: The VLSI Journal*, 19:1–81, 1995.

[5] V. Batagelj and M. Zaveršnik. Generalized cores. Preprint 799, Universtiy of Ljibljana, 2002.

[6] Vladimir Batagelj, Anuska Ferligoj, and Patrick Doreian. Generalized blockmodeling. *Informatica (Slovenia)*, 42(4), 1999.

[7] Code Red II and Nimda Worms and BGP Instability. http://www.renesys.com/projects/bgp_instability.

[8] Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. Experiments on graph clustering algorithms. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03)*, Lecture Notes in Computer Science. Springer-Verlag, 2003. To appear.

[9] The Code Red Worm. http://www.ciac.org/ciac/bulletins/l-117.shtml.

[10] G. Di Battista, M. Patrignani, and M. Pizzonia. Computing the types of the relationships between autonomous systems. In *IEEE INFOCOM 2003*, 2003.

[11] Patrick Doreian, Vladimir Batagelj, and Anuska Ferligoj. Symmetric-acyclic decompositions of networks. *Journal of Classification*, 17(1):3–28, 2000.

[12] T. Erlebach, A. Hall, and T. Schank. Classifying customer-provider relationships in the internet. In *Proceedings of the IASTED International Conference on Communications and Computer Networks*, pages 538–545, 2002.

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

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

[15] Christos Gkantsidi, Milena Mihail, and Ellen Zegura. Spectral analysis of internet topologies. In *IEEE Infocom 2003*, 2003.

[16] R Govindan and R. Reddy. An analysis of internet inter-domain topology and route stability. In *IEEE INFOCOM 1997*, 1997.

[17] Geoff Huston. BGP table statistics. http://bgp.potaroo.net.

[18] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.

[19] KPNQwest limps back after shutdown. http://news.com.com/2104-1033_3-946262.html.

[20] David Meyer. University of oregon route views project. http://www.routeviews.org/.

[21] The internet outage and attacks of october 2002. http://www.isoc-chicago.org/internetoutage.pdf.

[22] M. Rimondini, M. Pizzonia, G. Di Battista, and M. Patrignani. Algorithms for the inference of the commercial relationships between autonomous systems: Results analysis and model validation. submitted to this workshop, 2004.

[23] S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5(5):269–287, 1983.

[24] G. Siganos, M. Faloutsos, P. Faloutsos, and C. Faloutsos. Power laws and the as-level internet topology. In *IEEE/ACM Transactions on Networking (TON)*, volume 11, pages 514–524. ACM Press, 2003.

[25] S. Tauro, C. Plamer, G. Siganos, and M. Faloutsos. A simple conceptual model for the internet topology. In *IEEE Globalcom 2001*, volume 3, pages 1667–1671, 2001.

[26] Test Traffic Analysis Update. http://www.ripe.net/ripe/meetings/archive/ripe-43/presentations/ripe43-tt-analysis/.

[27] The W32.nimda Worm. http://www.ciac.org/ciac/bulletins/l-144.shtml.

Classifying Customer-Provider Relationships in the Internet*

Thomas Erlebach, Alexander Hall[†]
Computer Engineering and Networks Lab
ETH Zürich
CH-8092 Zürich, Switzerland
{erlebach|hall}@tik.ee.ethz.ch

Thomas Schank
Department of Computer & Information Science
Universität Konstanz
D-78457 Konstanz, Germany
schank@fmi.uni-konstanz.de

July 2002

Abstract

The problem of inferring customer-provider relationships in the autonomous system topology of the Internet leads to the following optimization problem: given an undirected graph G and a set P of paths in G, orient the edges of G such that as many paths as possible are valid, meaning that they do not contain an internal node with both incident edges on the path directed away from that node. The complexity of this problem was left open by Subramanian et al. ("Characterizing the Internet hierarchy from multiple vantage points," INFOCOM 2002). We show that finding an orientation that makes all paths valid (if such an orientation exists) can be done in linear time and that the maximization version of the problem is $\mathcal{NP}$-hard and cannot be approximated within $1/n^{1-\varepsilon}$ for n paths unless $\mathcal{NP}$=co-$\mathcal{RP}$. We present constant-factor approximation algorithms for the case where the paths have bounded length and prove that the problem remains $\mathcal{APX}$-hard in this case. Finally, we report experimental results demonstrating that the approximation algorithm yields very good solutions on real data sets.

1 Introduction

The Internet is a huge, complex network whose present state is the outcome of a distributed growth process without centralized control. Because of the importance of the Internet as our

*Research supported by EU Thematic Network APPOL II, IST-2001-32007, and FET Open project COSIN, IST-2001-33555, with funding provided by the Swiss Federal Office for Education and Science (BBW).

[†]Supported by the joint Berlin/Zurich graduate program Combinatorics, Geometry, and Computation (CGC), financed by ETH Zurich and the German Science Foundation (DFG).

basic communication infrastructure, significant research efforts have recently been devoted to the discovery and analysis of its topology. The Internet topology can be considered either on the level of individual routers and hosts or on the level of *autonomous systems* (subnetworks under separate administrative control). In this paper, we focus on the autonomous system (AS) topology of the Internet. The AS topology can be represented as an undirected graph: each vertex corresponds to an AS, and two vertices are joined by an edge if there is at least one physical link between the corresponding ASs. The AS topology has been investigated by a number of authors, see e.g. [6, 13, 4, 15].

Different ASs exchange routing information using the Border Gateway Protocol (BGP). If an AS X is connected to another AS Y, it *announces* the routes to a certain set $A(X,Y)$ of destination addresses to Y. This implies that when Y has a packet with a destination in the set $A(X,Y)$, it can forward the packet to the AS X, because X knows a route to that destination. An AS may choose not to announce all routes that it knows to all of its neighbors; this decision is determined by the routing policy employed by the AS.

While the AS topology provides information about the connections between ASs, it has been pointed out in [8] that it is also important to know the economic relationships between ASs, because these relationships determine the routing policies and thus the paths that packets can potentially take in the network. If an AS X is connected to an AS Y, then Y can be a customer, provider, or peer of X. If Y is a customer of X, then X announces all its routes to Y. If Y is a provider or a peer of X, then X announces only the routes to destinations in its own AS and to destinations announced by its own customers; it does (usually) not announce routes to destinations that it can reach only through another provider or through peers. By knowing customer-provider relationships between ASs, one could have a sound basis for investigations of routing issues in the Internet.

Since data about these relationships is not easy to obtain directly, however, a natural idea is to infer AS relationships from the routing paths observed in the network (these paths can be determined from BGP information). Subramanian et al. [14] pursue this approach and propose a formulation of this inference problem as a combinatorial optimization problem, called the Type-of-Relationship problem: given a graph and a set of paths in the graph, classify the edges of the graph into customer-provider relationships and peer-peer relationships such that as many of the paths as possible are valid (consistent with this classification). Here, a path is valid if it consists of zero or more customer-provider edges, followed by at most one peer-peer edge, followed by zero or more provider-customer edges. Subramanian et al. write that they suspect the problem to be $\mathcal{NP}$-hard but have been unable to prove this, and they propose a heuristic algorithm.

1.1 Our results

In this paper, we consider two versions of the Type-of-Relationship problem: the problem of computing a classification such that *all* paths are valid (if such a classification exists), denoted AllToR, and the problem of computing a classification that maximizes the number of valid paths, denoted MaxToR. We prove that AllToR can be solved in linear time by reducing it to 2SAT and we show that MaxToR is $\mathcal{NP}$-hard, thus settling the complexity of the Type-of-Relationship problem left open in [14]. Our hardness proof implies that MaxToR cannot be approximated within $1/n^{1-\varepsilon}$ (for any $\varepsilon > 0$) for general instances with n paths unless $\mathcal{NP}$=co-$\mathcal{RP}$. Motivated by the characteristics of instances arising in practice, we then consider the case where the given paths are short. First, we show that MaxToR can be approximated within a factor of $(k+1)/2^k$ if all paths have length at most k. Then we give approximation algorithms achieving a better approximation ratio for $k = 2, 3, 4$. We also prove that MaxToR is $\mathcal{APX}$-complete for paths of bounded length. $\mathcal{APX}$-completeness (see e.g. [2]) implies that MaxToR cannot be approximated within a certain constant factor unless $\mathcal{P} = \mathcal{NP}$, even for instances with short paths. Finally, we have implemented our approximation algorithm and obtained very encouraging results on real data sets.

Independently of our work, Di Battista, Patrignani and Pizzonia [5] have recently also obtained the result that AllToR can be solved in linear time and that MaxToR is $\mathcal{NP}$-hard. Interestingly, they also use 2SAT to solve the AllToR problem and present an $\mathcal{NP}$-hardness proof by reducing Max2SAT. Their reduction is based on the same idea with which we are able to prove $\mathcal{APX}$-hardness. Di Battista et al. do not consider the question of approximability or inapproximability of MaxToR. Instead, they give an $\mathcal{NP}$-hardness result for the problem of maximizing peer-peer relationships and present a new heuristic algorithm for MaxToR. They do not give approximation bounds for the algorithm, but they show that it performs well on real data sets.

The remainder of the paper is structured as follows. Definitions and preliminaries are given in Section 2. Section 3 deals with the AllToR problem. The hardness results for general MaxToR are given in Section 4. In Section 5, the approximability of MaxToR instances with paths of bounded length is studied. Section 6 describes our experimental results.

2 Preliminaries

We are given a simple, undirected graph $G = (V, E)$ whose nodes correspond to the autonomous systems of the Internet and whose edges correspond to physical connections between those autonomous systems. Furthermore, we are given a set P of simple, undirected paths in G. (We allow that P contains a path several times, i.e., P can actually be a multi-set.) These paths are possible data routes in the Internet; they are usually obtained from BGP routing tables. Informally, the goal is to classify the edges of G as customer-provider or peer-peer relationships in a "correct" way by using only the information obtained from the set of paths P. Here, the basic assumption is that the routing policies of the ASs depend on these relationships in the way described in Section 1, thus motivating the following definition (given in [14]).

Definition 1 *For a classification of the edges of G into customer-provider and peer-peer relationships, a path $p \in P$ is valid if*

- *it starts with zero or more customer-provider edges*
- *followed by zero or one peer-peer edge*
- *followed by zero or more provider-customer edges.*

The problem of classifying the edges into peer-peer or customer-provider relationship such that a maximum number of paths are valid was posed as the Type-of-Relationship (ToR) problem in [14]. We consider two versions of this problem: In the first variant, denoted AllToR, the goal is to decide whether there is an edge classification such that all paths in P are valid, and to compute such a classification if it exists. In the second variant, denoted MaxToR, the goal is to compute an edge classification such that a maximum number of paths in P are valid.

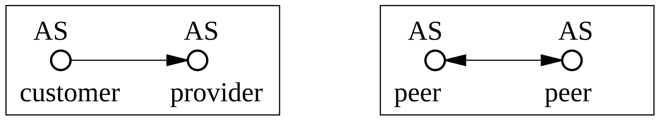


Figure 1: Representation of customer-provider and peer-peer relationship.

For convenience we represent a customer-provider edge as a directed edge from customer to provider and a peer-peer edge as a bidirected edge between the two peers as shown in Figure 1.

Figure 2 gives some examples of valid and invalid paths. Using the directed and bidirected edge formulation for customer-provider and peer-peer relationships, we can rephrase the validity of a path as follows, where we take a bidirected edge to be pointing away from both of its endpoints.

Lemma 2 *For a given edge classification (represented as directed and bidirected edges), a path p is valid if and only if it does not contain a node from which two edges of p are pointing away.*

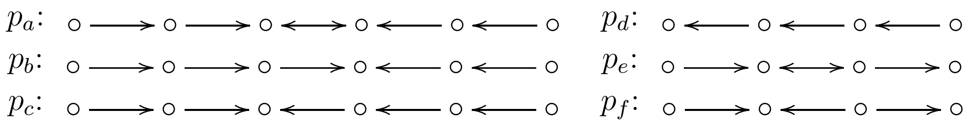


Figure 2: Paths $p_a, \ldots, p_d$ are valid, p_e and p_f are not valid.

Finally, we argue that peer-peer edges can be completely disregarded in this formulation of the problem. To see this, assume that for a graph G and a set of paths P, the relationships between the ASs have been classified as directed or bidirected edges. Now replace every bidirected edge by a directed edge in arbitrary direction. Using Lemma 2, we see immediately that all the valid paths remain valid after this operation. Figure 2 illustrates this transition for both directions from path p_a to path p_b and from path p_a to p_c, respectively. Therefore, we can assume without loss of generality that the solutions for MaxToR and AllToR computed by an algorithm do not contain any bidirected edges.[1]

An assignment of directions to the edges of an undirected graph is called an *orientation* of the graph. We can now state the AllToR and MaxToR problems simply as follows:

AllToR**:** Given a graph G and a set of paths P, decide whether there is an orientation of G such that all paths in P are valid, and compute such an orientation if it exists.

MaxToR**:** Given a graph G and a set of paths P, compute an orientation of G such that a maximum number of paths in P are valid.

In the case of MaxToR, we will be interested in approximation algorithms. Here, an algorithm for MaxToR is a ρ-approximation algorithm if it runs in polynomial time and always computes an orientation of G such that the number of valid paths is at least ρ times the number of valid paths in the optimal orientation. Note that $\rho \leq 1$.

The length of a path is defined as the number of edges on the path. All nodes on a path except its two endpoints are called *internal* nodes of the path.

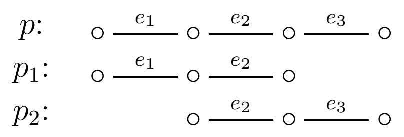
[1]From a practical point of view, it might be interesting to find an alternative formulation of the problem where peer-peer edges are actually necessary.

3 The AllToR problem

In this section we show that the AllToR problem is solvable in linear time. To this end we show that an instance of AllToR can be reduced to a 2SAT problem.

Lemma 3 *A path $p \in P$ of length k can be split up into $k-1$ paths p_i, $1 \leq i \leq k-1$, each of length 2, such that for any orientation of G, p is valid if and only if all p_i are valid.*

Proof: The construction of the $k-1$ paths works as follows: path p_1 consists of the first two edges of p. Path p_i consist of edge i and edge $i+1$ of path p. In this way, path p_i overlaps with path p_{i+1} by one edge. Figure 3 shows how a path of length 3 is split up into two paths of length 2. The correctness of the lemma follows directly from Lemma 2. □



directed path p_1	e_1	e_2	p_1 valid	2SAT clause
$\circ \xrightarrow{e_1} \circ \xleftarrow{e_2} \circ$	in	in	yes	$e_1 \vee e_2$
$\circ \longrightarrow \circ \longrightarrow \circ$	in	out	yes	$e_1 \vee \overline{e_2}$
$\circ \longleftarrow \circ \longleftarrow \circ$	out	in	yes	$\overline{e_1} \vee e_2$
$\circ \longleftarrow \circ \longrightarrow \circ$	out	out	no	$\overline{e_1} \vee \overline{e_2}$

Figure 3: Decomposition of paths (left), and truth table for a path of length 2 (right).

By Lemma 3, it suffices to consider the AllToR problem for instances with paths of length 2. (Paths of length 1 are always valid.) The truth table on the right-hand side of Figure 3 shows a one-to-one correspondence between the validity of a path of length 2 and the logical or of two literals. This suggests the use of 2SAT to solve the AllToR problem, resulting in the following algorithm:

1. Orient the edges of $G = (V, E)$ arbitrarily.
2. Split all paths $p \in P$ into $\sum_{p \in P}(\ell(p) - 1)$ paths of length 2 as shown above, where $\ell(p)$ is the length of path p.
3. Construct a 2SAT instance where each edge $e_i \in E$ corresponds to a variable and each path p (of length 2) corresponds to a clause in the following way: like in the truth table of Figure 3, e_i appears negated if the corresponding edge is pointing away from the internal node v of p and not negated if it is pointing towards v.
4. Solve the resulting 2SAT instance.
5. If the 2SAT instance is not satisfiable, the AllToR instance is not solvable. Otherwise, flip the directions of all edges whose corresponding variable has been assigned *false*. This gives an orientation where all paths are valid.

Correctness of the algorithm can be verified easily by looking at the four possible configurations for input paths of length two and the assignment of the variables in the corresponding clause. Steps 1, 2, 3 and 5 can obviously be performed in time linear in the size of the input. 2SAT is well known to be solvable in linear time, so the running-time for Step 4 is linear as well.

Theorem 1 AllToR *can be solved in linear time.*

4 The general MaxToR problem

In this section we show that MaxToR is $\mathcal{NP}$-hard and even cannot be approximated well. To achieve this we will give an approximation-preserving polynomial reduction from the well known $\mathcal{NP}$-hard maximum independent set problem (denoted MaxIS) to MaxToR. First, consider a graph G with two paths as shown in Figure 4. It can be verified that in this graph there is no orientation of the edges such that both paths are valid: Each of the three shared edges of both paths would require to flip the direction in one of the paths to make them valid. But since a change of direction of the edges in a valid path can happen only once, one of the two paths must be invalid.

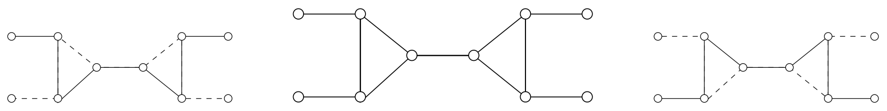


Figure 4: A graph (middle) with two paths (left and right) that cannot both be valid in any orientation.

Lemma 4 *There exists a graph G with two paths such that only one of the two paths can be valid.*

We will use this construction as a gadget to obtain a reduction from MaxIS to MaxToR. Let an instance of MaxIS be given by an undirected graph $H = (V_H, E_H)$. We create an instance (G, P) of MaxToR by mapping every node in H to a path in P such that any two paths in P

- are edge-disjoint if there is no edge between them in H and
- cannot be valid simultaneously in any orientation if there is an edge between them in H.

Obviously, such an instance (G, P) can be constructed in polynomial time using the gadget of Figure 4.

Now observe that there is a one-to-one correspondence between orientations of G with t valid paths and independent sets in H with cardinality t. In particular, if we could solve MaxToR in polynomial time or approximate it in polynomial time with ratio ρ, we could also solve MaxIS in polynomial time or approximate it with ratio ρ, respectively. Since MaxIS is known to be $\mathcal{NP}$-hard and not even approximable with ratio $1/n^{1-\epsilon}$ on graphs with n nodes for any $\varepsilon > 0$ unless $\mathcal{NP}$=co-$\mathcal{RP}$ [10], we obtain the following hardness result for MaxToR.

Theorem 2 MaxToR *is $\mathcal{NP}$-hard and cannot be approximated within a factor of $1/n^{1-\varepsilon}$ on instances with n paths for any $\varepsilon > 0$ unless $\mathcal{NP}$= co-$\mathcal{RP}$.*

Thus it is not feasible to get a good approximation ratio for the general MaxToR problem. We might hope, however, to be able to exploit the structure of real AS graphs or the observed routing paths in order to give an algorithm with good approximation ratio for a restricted case of MaxToR.

5 Approximating MaxToR instances with bounded path length

Motivated by the negative result concerning the approximability of the general MaxToR problem, we now consider a restricted version of the problem. Looking at real world data [14, 1], we noticed that most of the paths are actually quite short (see also Section 6). This leads us to consider instances of MaxToR where the maximum length of the paths is bounded by a constant. In the following, we first present a very simple randomized approach achieving constant approximation ratio for this case. Then we show how to use an approximation algorithm for Max2SAT to achieve significantly better approximation ratios for instances containing only paths of length at most k for $k = 2, 3, 4$. Finally, we prove $\mathcal{APX}$-completeness for instances where the path length is bounded by an arbitrary constant, by an approximation preserving reduction from the Max2SAT problem, which is well known to be $\mathcal{APX}$-complete [2].

5.1 A simple constant factor approximation algorithm

For paths of constant length there is a very easy randomized approximation algorithm: just select the directions of the edges independently at random. If each edge is oriented in one of the two possible ways with probability 1/2, a path of length k is valid with probability

$$p_k = \frac{k+1}{2^k}.$$

To see this, note that in a valid path the direction of the edges can change only once at one of the $k-1$ internal nodes or not at all. There are $k-1$ possibilities for the direction of all edges in the path in the former case and 2 possibilities in the latter case. Altogether there are 2^k possibilities to orient the edges. If all paths have length at most k, we get by linearity of expectation

$$\mathbb{E}(A_{rand}) \geq \frac{k+1}{2^k} n \geq \frac{k+1}{2^k} Opt,$$

where A_{rand} is the value of the approximate solution and Opt the value of the optimum. The algorithm can easily be derandomized in the standard way (method of conditional probabilities), giving the following theorem.

Theorem 3 *The* MaxToR *problem with paths no longer than k edges can be approximated within a factor of $\frac{k+1}{2^k}$ of the optimum in polynomial time.*

For example, this gives ratio 0.75 for paths of length at most 2, 0.5 for paths of length at most 3, and $5/16 = 0.3125$ for paths of length at most 4. We can also state the following corollary.

Corollary 5 *The* MaxToR *problem with average path length bounded by k can be approximated within a factor of $\frac{2k+1}{2 \cdot 4^k}$ of the optimum in polynomial time.*

Proof: Consider an instance of MaxToR with n paths. If the average path length is k, there are at least $n/2$ paths with length at most $2k$. Applying the simple randomized algorithm described above to these $n/2$ paths, we obtain a solution with at least

$$\frac{n}{2} \cdot \frac{2k+1}{4^k}$$

valid paths. □

5.2 Better ratios for paths of short length

To obtain better ratios for paths of length at most 4 or less we use the approximation algorithm for Max2SAT due to Lewin, Livnat and Zwick [12]. They achieve a ratio of $r = 0.940$ by enhancing the semidefinite programming (SDP) based approach first presented in the seminal paper by Goemans and Williamson [9]. Given an instance (G, P) of MaxToR, we construct an instance of Max2SAT from the paths as described in Section 3 (by splitting every path into paths of length two and adding a clause for each length 2 path) and apply the Max2SAT approximation algorithm to the resulting instance. Then we orient the edges of G according to the assignment returned by the Max2SAT algorithm. It may seem surprising that this approach gives a good ratio because there is not necessarily a one-to-one correspondence between paths and clauses.

First, a bit of notation: Let *Opt* denote the optimum value of the considered MaxToR instance and A_k the value of our approximate solution for an instance with path length bounded by k. Let g_k be the number of paths in P that have length exactly k. Note that paths of length 1 can be ignored, since they are always valid. Hence, we can assume $g_1 = 0$. Furthermore, note that for deriving lower bounds on the approximation ratio A_k/Opt, it suffices to consider only instances in which all paths have length exactly k: If an instance contains a shorter path, this path can easily be lengthened by adding an appropriate number of extra nodes and edges to the path at one of its ends. At most $k \cdot n$ edges and nodes are added, and a solution to this modified instance clearly gives a solution to the original instance with at least the same number of valid paths.

In the simplest case, when all paths have length 2, we can directly transfer the ratio $r = 0.940$ from Max2SAT to MaxToR. This is because each path is represented by exactly one clause which is satisfied if and only if the path is valid.

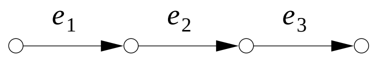


Figure 5: A path of length 3 which leads to the two clauses $e_1 \vee \overline{e_2}$ and $e_2 \vee \overline{e_3}$. See Section 3 for details about the definition of the clauses given a graph and paths.

Now consider a path of length 3. It is represented by two 2SAT clauses. The variable corresponding to the edge in the middle appears in both clauses, once negated and once not negated (see Figure 5 for an example). Therefore, one of the two clauses is always satisfied. Clearly, both clauses are satisfied if and only if the path is valid. This gap of either one or two clauses being satisfied can be used to derive a bound on the approximation ratio. Consider a MaxToR instance with $n = g_3$ paths of length 3. Note that an optimal solution of Max2SAT has the value $Opt + g_3$ and directly gives an optimal solution to the MaxToR problem (with value *Opt*). The Max2SAT approximation algorithm satisfies $A_3 + g_3$ clauses with

$$\frac{A_3 + g_3}{Opt + g_3} \geq r. \tag{1}$$

Because there is always a solution to Max2SAT such that at least 3/4 of the clauses are satisfied and there are $2 \cdot g_3$ clauses, we have $A_3 + g_3 \geq 3/2 \cdot g_3$ or $g_3 \leq 2 \cdot A_3$. Applying this to (1) leads to

$$\begin{aligned} A_3 &\geq r \cdot (Opt + g_3) - g_3 \geq r \cdot Opt + 2(r-1) \cdot A_3 \\ A_3 &\geq \frac{r}{3-2r} Opt \end{aligned}$$

giving approximation ratio $r/(3-2r) \geq 0.839$. Note that this is a considerable improvement over the ratio 1/2 obtained for paths of length three by Theorem 3.

In a MAX2SAT instance derived from paths of length 4 there will be three clauses for each of the paths. We refer to the three clauses of a path as a *triple*. With the same argumentation as for length 3 paths, we have that for each triple at least one clause is always satisfied and all three are satisfied if and only if the corresponding path is valid. This shows that any solution of this instance satisfies

$$x + y + g_4$$

clauses, where g_4, y and x are the number of triples with at least one, at least two and exactly three satisfied clauses, respectively. Note that x yields the number of valid paths and $x \leq y \leq g_4$.

We now compare a solution $S = A_4 + y + g_4$ computed by the MAX2SAT approximation algorithm with a solution $S' = Opt + y' + g_4$ derived from an optimal solution of the corresponding MAXTOR instance. By [7] we know that $S/S' \geq r$. From this we can bound the approximation ratio A_4/Opt. In the worst case $y = g_4$ and $y' = Opt$. This gives

$$\frac{A_4 + 2 \cdot g_4}{2 \cdot Opt + g_4} \geq r$$
$$A_4 \geq 2r \cdot Opt + (r-2) \cdot g_4 \geq 2r \cdot Opt + 4(r-2)A_4,$$

because there are $3 \cdot g_4$ clauses of which at least 3/4 are satisfied in the approximate solution, i.e. $A_4 + 2 \cdot g_4 \geq 9/4 \cdot g_4$ or $g_4 \leq 4 \cdot A_4$. Solving for A_4 we get

$$A_4 \geq \frac{2r}{9-4r} \cdot Opt.$$

This gives an approximation ratio of $2r/(9-4r) \geq 0.358$, which is a slight improvement compared to $5/16 = 0.3125$.

Note that this approach cannot be carried over to paths of length greater than 4. It uses the fact that at least 3/4 of the clauses can be satisfied, which does not help if each path is represented by 4 or more clauses and the path is valid if and only if all of them are satisfied.

Summarizing the results discussed above, we get the following theorem.

Theorem 4 *The* MAXTOR *problem with paths no longer than k edges can be approximated within a factor c_k of the optimum in polynomial time, where c_k has the following form:*

- $c_2 := 0.940$
- $c_3 := 0.839$
- $c_4 := 0.358$
- $c_k := \frac{k+1}{2^k}$ *for* $k > 4$

5.3 $\mathcal{APX}$-hardness

So far we have given constant-factor approximation algorithms for instances of MaxToR with paths of bounded length. Now we show that even instances containing only paths of length 2 cannot be approximated better than some constant unless $\mathcal{P} = \mathcal{NP}$. Both results together give that MaxToR with constant maximal path length is $\mathcal{APX}$-complete.

To this aim, we reduce Max2SAT and apply an $\mathcal{APX}$-hardness result by Håstad [11]. We start by reducing a Max2SAT instance to a MaxToR instance (G, P) with paths of length 3. Then we explain how this instance can be modified such that it contains only paths of length 2.

Assume that we are given a Max2SAT instance with variables x_i, $i \in \{1 \ldots n'\}$ and clauses c_j, $j \in \{1 \ldots m'\}$. For each variable x_i we add two nodes $\overline{x_i}$, x_i to G and an edge $e_i = \{\overline{x_i}, x_i\}$ between them. If in a solution to MaxToR this edge is directed towards x_i this corresponds to $x_i = \mathit{true}$. Otherwise, if it is directed towards $\overline{x_i}$, this means $x_i = \mathit{false}$. For each clause $c_k = l_i \vee l_j$, with literals $l_i \in \{x_i, \overline{x_i}\}$ and $l_j \in \{x_j, \overline{x_j}\}$, one edge $\{l_i, l_j\}$ is added. Additionally a path of length 3 is added to P along the nodes $\overline{l_i}, l_i, l_j, \overline{l_j}$. Figure 6 shows a simple example with one clause, and the graph on the left-hand side of Figure 7 shows an example with two clauses.

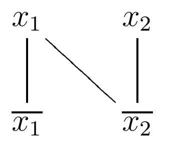
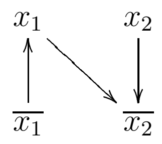
Max2SATclause	path	graph	directed path	truth value
$(x_1 \vee \overline{x_2})$	$\overline{x_1}, x_1, \overline{x_2}, x_2$			$x_1 = 1$, $x_2 = 0$

Figure 6: Example of how an instance of MaxToR is constructed from a Max2SAT instance. On the right-hand side, a possible assignment of directions to the edges and the corresponding truth values are shown.

The paths and edges are defined such that a path is valid in a solution to the MaxToR instance if and only if the corresponding clause is satisfied. This is clear because a path is valid if and only if either e_i is directed towards l_i or e_j towards l_j, which corresponds to a truth assignment where $l_i \vee l_j$ is satisfied. In particular, note that given a satisfied clause the edge $\{l_i, l_j\}$ can always be directed such that the whole path is valid. Conversely, for an unsatisfied clause no such direction of $\{l_i, l_j\}$ exists.

Thus, maximizing the number of valid paths also maximizes the number of satisfied clauses. With [11] we get that MaxToR is not approximable within ratio $q = 0.955$ for instances with paths of length at most k for any constant $k \geq 3$.

To obtain a similar result for paths of length 2, we modify the instance as follows: each path $\overline{l_i}, l_i, l_j, \overline{l_j}$ is replaced by two overlapping paths $\overline{l_i}, l_i, l_j$ and $l_i, l_j, \overline{l_j}$. See the graph in the middle of Figure 7 for an example. Clearly, in a MaxToR solution one of the two paths is always valid. The corresponding clause is satisfied if and only if both paths are valid. So an optimal solution to the MaxToR instance with *Opt* valid paths gives an optimal assignment to the variables such that $\mathit{Opt} - m'$ clauses are satisfied. An approximate solution to MaxToR giving A_2 valid paths leads to $A_2 - m'$ satisfied clauses. With [11] we know that

$$\frac{A_2 - m'}{\mathit{Opt} - m'} \leq q$$

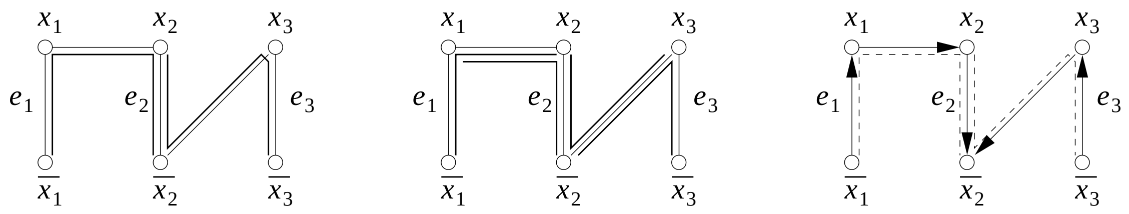


Figure 7: Network and paths resulting from the two clauses $x_1 \vee x_2$, $\overline{x_2} \vee x_3$: Constructed instance with length 3 paths (left), modified instance with length 2 paths (middle), and possible solution where both clauses are satisfied (right).

for at least one instance of MAX2SAT. With $Opt - m' \geq 3/4 \cdot m'$ (at least 3/4 of the clauses of any MAX2SAT instance can be satisfied) this yields

$$A_2 \leq \frac{4+3q}{7} \cdot Opt,$$

which gives the following theorem.

Theorem 5 *Unless $\mathcal{P} = \mathcal{NP}$, there is no approximation algorithm for* MAXTOR *with paths of length at most k that achieves ratio at least $\frac{4+3q}{7} \leq 0.981$ if $k = 2$ and ratio at least 0.955 if $k \geq 3$.*

In particular, with Theorem 4 we have that MAXTOR is $\mathcal{APX}$-complete for paths with constant maximum length.

6 Experimental results

In this section we discuss some experiments conducted with the approximation algorithm for MAXTOR presented in Section 5.2: the algorithm constructs a MAX2SAT instance from the given paths and uses an SDP-based algorithm to obtain an approximate solution. For our experiments we used the data that has been accumulated by Subramanian et al. [14] and is available on the WWW [1]. For three different dates (18 April 2001, 4 February 2002, and 6 April 2002) routing paths from 10, 9, respectively 14 autonomous systems were collected. From these 3.4 to 6.4 million paths in a network of about 10,000 nodes and 25,000 edges, the edge directions were deduced. It is notable that the path lengths are relatively short (see Table 1), i.e. the assumption in Section 5.2 that the path lengths are bounded by a small constant is quite realistic.

Table 1: Size of the network and number of paths at the three different dates. The maximal and average path lengths are given before ($\max_1$ and avg_1) and after ($\max_2$ and avg_2) the preprocessing described in the text.

Date	# of nodes	# of edges	# of paths	$\max_1$	avg_1	$\max_2$	avg_2
18 Apr 2001	10923	23935	3423422	12	3.52	10	1.70
4 Feb 2002	12788	27936	4988100	11	3.52	9	1.32
6 Apr 2002	13164	28510	6356435	11	3.55	8	1.39

In our implementation the paths are first preprocessed by directing edges that can be directed without conflicts. This shortens the paths considerably (cf. Table 1). Then it is checked with the AllToR algorithm described in Section 3 whether the graph can be oriented such that all paths are valid. This was not the case for any of the three dates. Finally, an approximate solution is calculated as described in Section 5.2: Max2SAT is relaxed as in [9] and the rounding is done as in [7]. In [7] the improved approximation ratio could be achieved by adding new constraints and a modified rounding strategy. We could not add the constraints because the instance would have become too large. The new rounding strategy was adopted though. The freely available solver DSDP 4.5 [3] was used to solve the semidefinite programs. The overall results are given in Table 2. The computed orientations make roughly 98% of the given paths valid. This should be contrasted with the edge classifications computed by the heuristic algorithm of [14] that are also available at [1]. In these edge classifications, only about 80% of the paths are valid. This demonstrates that it pays off to use SDP-based approximation algorithms for MaxToR in practice.

Table 2: Experimental results achieved with the approximation algorithm presented in Section 5.2.

Date	Total # of paths	# of valid paths
18 Apr 2001	3423422	3360926 (98.17%)
4 Feb 2002	4988100	4919310 (98.62%)
6 Apr 2002	6356435	6204849 (97.62%)

References

[1] S. Agarwal. Data used in [14]. *http://www.cs.berkeley.edu/∼sagarwal/research/BGP-hierarchy/*, 2002.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation. Combinatorial Optimization Problems and their Approximability Properties*. Springer, Berlin, 1999.

[3] S. Benson. Semidefinite programming solver DSDP 4.5. *http://www-unix.mcs.anl.gov/∼benson/*, 2002.

[4] T. Bu and D. Towsley. On distinguishing between Internet power law topology generators. In *Proceedings of INFOCOM'02*, June 2002.

[5] G. Di Battista, M. Patrignani, and M. Pizzonia. Computing the types of the relationships between autonomous systems. Technical Report RT-DIA-73-2002, Dipartimento di Informatica e Automazione, Università di Roma Tre, 2002.

[6] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *Proceedings of ACM SIGCOMM'99*, 1999.

[7] U. Feige and M. Goemans. Approximating the value of two proper proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proceedings of the Third Israel Symposium on Theory of Computing and Systems*, pages 182–189, 1995.

[8] L. Gao. On inferring autonomous system relationships in the internet. *IEEE/ACM Transactions on Networking*, 9:733–745, 2000.

[9] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[10] J. Håstad. Clique is hard to approximate within $n^{1-\varepsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[11] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[12] M. Lewin, D. Livnat, and U. Zwick. Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems. In *Integer Programming and Combinatorial Optimization (IPCO)*, LNCS 2337, pages 67–82, 2002.

[13] A. Medina, I. Matta, and J. Byers. On the origin of power laws in Internet topologies. *ACM Computer Communication Review*, 30(2):18–28, April 2000.

[14] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *Proceedings of INFOCOM'02*, June 2002.

[15] D. Vukadinović, P. Huang, and T. Erlebach. On the spectrum and structure of Internet topology graphs. In *Innovative Internet Computing Systems*, LNCS 2346, pages 83–95, June 2002.

Computing the Types of the Relationships between Autonomous Systems

Giuseppe Di Battista, Maurizio Patrignani, and Maurizio Pizzonia
Dipartimento di Informatica e Automazione, Università di Roma Tre, Rome, Italy
Email: {gdb,patrigna,pizzonia}@dia.uniroma3.it

***Abstract*— We investigate the problem of computing the types of the relationships between Internet Autonomous Systems. We refer to the model introduced in [1], [2] that bases the discovery of such relationships on the analysis of the AS paths extracted from the BGP routing tables. We characterize the time complexity of the above problem, showing both NP-completeness results and efficient algorithms for solving specific cases. Motivated by the hardness of the general problem, we propose heuristics based on a novel paradigm and show their effectiveness against publicly available data sets. The experiments put in evidence that our heuristics performs significantly better than state of the art heuristics.**

I. Introduction

An *Autonomous System* (AS) is a portion of Internet under a single administrative authority. Currently, there are more than $10,000$ ASes and their number is rapidly growing. They interact to coordinate the IP traffic delivery, exchanging routing information with a protocol called Border Gateway Protocol (BGP) [3].

Several authors (see, e.g. [4], [5]) have pointed out that the relationships between ASes can be roughly classified into categories that have both a commercial and a technical flavor. A pair of ASes such that one sells/offers Internet connectivity to the other is said to have a *provider-customer* relationship. If two ASes simply provide connectivity between their respective customers are said to have a *peer-to-peer* relationship. Finally, if two ASes offer each other Internet connectivity are said to be *siblings*. Of course, this classification does not capture all the shades of the possible commercial agreements and technical details that govern the traffic exchanges between ASes but should be considered as an important attempt toward understanding the Internet structure.

Since many applications would benefit from the knowledge about the Internet structure, the research on the subject has recently produced many contributions. More specifically, there is a wide research area focusing on the discovery of the topology underlying the Internet structure, either at the AS and at the router level (see, for example, [6], [7], [8]).

Other researchers concentrate more directly on the above mentioned relationships and on the hierarchy that they induce on the set of ASes. Govindan and Reddy [6] study the interplay between the *degree* of the ASes and their rank in the hierarchy, where the degree of an AS is the number of ASes that have some kind of relationship with it. Gao [1] studies, for the first time, the following problem. ASes are the vertices of a graph (*AS graph*) where two ASes are adjacent if they exchange routing information; the edges of such a graph should be labeled in order to reflect the type of relationship they have. In order to infer the relationships between ASes, Gao uses the information on the degree of ASes together with the *AS paths* extracted from the BGP routing tables. An AS path is the sequence of the ASes traversed by a connectivity offer (*BGP announcement*). In [1] a heuristic is presented together with experimental results. An analysis on the properties of the labeled graphs obtained with such heuristics is provided in [9].

Subramanian et al. [2] formally define, as a minimization problem, a slightly simplified version of the problem addressed in [1] and conjecture its NP-completeness. They also propose a heuristic based on the observation of the Internet from multiple vantage points, which does not rely on the degree of the ASes. Further, they validate the results obtained by the heuristic against a rich collection of data sets.

This paper contributes to the line of research opened in [1], [2]. Namely, its main results are the following.

- We solve a problem explicitly stated in [2]. Namely, we characterize the complexity of determining the relationships between ASes while minimizing the number of "anomalies". In particular:
  - We show that such a problem is NP-complete in the general case;
  - We produce a linear time algorithm for determining the AS relationships in the case in which the problem admits a solution without anomalies; and
  - We use such a linear time algorithm to show that for large portions of the Internet (e.g., data obtained from single points of view) it is often possible to determine the relationships between ASes with no anomalies.
- We introduce heuristics, based on a novel approach, for determining the relationships between ASes with a small number of anomalies.
- We experimentally show that the proposed approach leads to heuristics that performs significantly better than the cutting edge heuristics of [2].

Work partially supported by European Commission - Fet Open project COSIN - COevolution and Self-organisation In dynamical Networks - IST-2001-33555, by "Progetto ALINWEB: Algoritmica per Internet e per il Web", MIUR Programmi di Ricerca Scientifica di Rilevante Interesse Nazionale, and by "The Multichannel Adaptive Information Systems (MAIS) Project", MIUR Fondo per gli Investimenti della Ricerca di Base.

The paper is structured as follows. Section II describes the addressed problem. Sections III and IV show an algorithm for testing if the problem admits a solution with no anomalies, and show how to find a solution if it exists. In Section V we prove the NP-completeness of the problem in the general case. Section VI shows new heuristics and compare the results with the state of the art. Finally, Section VII contains conclusions and open problems.

II. Problem Description

A *prefix* is a block of destination IP addresses. An Internet Autonomous System (AS) applies local policies to select the best *route* for each prefix and to decide whether to *export* this route to neighboring ASes.

Several authors have pointed out that ASes typically have *provider-customer* or *peer-to-peer* relationships (see, e.g. [4], [5], [10], [2]). A *customer* exports to a provider its routes and the routes learned from its own customers, but does not export routes learned from other providers or peers. A *provider* exports to a customer its routes, the routes learned from the other customers, its providers, and its peers. *Peers* export to each other their own routes and the routes learned from their customers but do not export the routes learned from their providers and other peers.

Consider the *AS paths* that are associated with the BGP announcements of the routes. If all the ASes adopted export policies according to the above model, then the AS paths would have a peculiar structure [1], [2]. Namely, (1) no AS path can contain more than one pair of ASes having a peer-to-peer relationship; and (2) once a provider-customer or a peer-to-peer pair of ASes is met in the AS path, no customer-provider can be found in the remaining part of it.

Further, the above mentioned peculiarities of the AS paths have been formally stated in a theorem of [1], that has been also re-casted in [2]. A graph-theoretic formulation of the same theorem will be given in what follows.

A. Type-of-Relationship problem

The relationships between ASes in the Internet may be represented as a graph G whose edges are either directed or undirected. Each vertex is an AS, a directed edge from vertex u to vertex v indicates that u is a customer of v (provider-customer relationship), and an undirected edge between vertex w and vertex z indicates that w and z are peers (peer-to-peer relationship). A BGP AS path corresponds to a path on G. Suppose path p is composed by the sequence of vertices $v_1, \ldots, v_n$, then p is *valid* if it is of one of the following two types.

Type 1: p is composed by a (possibly empty) sequence of forward edges followed by a (possibly empty) sequence of backward edges; more formally, there exists a vertex v_i of p such that for $j \in 1, \ldots, i-1$ edge (v_j, v_{j+1}) is directed from v_j to v_{j+1} and for $j \in i, \ldots, n-1$ edge (v_j, v_{j+1}) is directed from v_{j+1} to v_j. (See Figure 1.a).

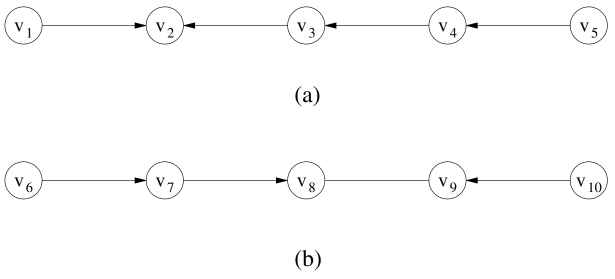


Fig. 1. An example of Type 1 (a) and of Type 2 (b) path.

Type 2: p is composed by a (possibly empty) sequence of forward edges, followed by an undirected edge, followed by a (possibly empty) sequence of backward edges; more formally, there exists a vertex v_i of p such that for $j \in 1, \ldots, i-1$ edge (v_j, v_{j+1}) is directed from v_j to v_{j+1}, edge (v_i, v_{i+1}) is undirected, and for $j \in i+1, \ldots, n-1$ edge (v_j, v_{j+1}) is directed from v_{j+1} to v_j. (See Figure 1.b).

See Figure 1 for examples of Type 1 and of Type 2 paths. An *invalid path* is a path that is not valid.

At this point the above mentioned theorem [2] can be restated as follows: if every AS obeys the customer, peer, and provider export policies, then every advertised path is either of Type 1 or of Type 2.

However, the Internet is more complex. To give a few examples: ASes operated by the same company can have a *sibling* relationship, where each AS exports all its routes to the other; two ASes may agree a *backup* relationship between them, to overcome possible failures; or ASes may have peering relationships through intermediate ASes. However, finding out which is the portion of Internet that obeys the customer, peer, and provider export policies can be considered as the first step toward a complete comprehension of the relationships between ASes. Such motivations have pushed the authors of [2] toward identifying the following problem.

Type-of-Relationship (ToR) Problem [2]: Given an undirected graph G and a set of paths P, give an orientation to some of the edges of G to minimize the number of invalid paths in P.

Figure 2 shows an instance of the ToR problem for which an orientation without invalid paths cannot be found. In particular, each orientation of edge (AS701, AS5056) yields at least one invalid path. Suppose, in fact, that edge (AS701, AS5056) was directed from AS701 to AS5056. Path AS5056, AS701, AS4926, AS6461, AS2914, AS174, AS14318 (drawn solid in the figure) would be valid only if edge (AS4926, AS6461) was directed from AS6461 to AS4926. Similarly, path AS5056, AS701, AS6461, AS4926, AS4270, AS4387 (drawn dotted in the figure) would be valid only if edge (AS4926, AS6461) was directed from AS4926 to AS6461. Hence, we have a contradiction, since edge (AS4926, AS6461) should have opposite orientations. Now, suppose that edge (AS701, AS5056) was undirected. The same arguments apply, leading to the same contradiction. Finally, suppose that edge

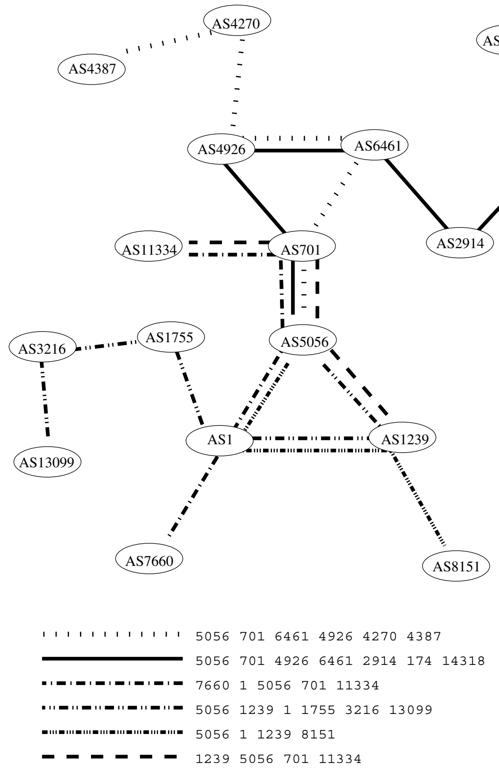


Fig. 2. An instance of the ToR problem that does not admit an orientation without invalid paths. The six paths of the instance are represented with different line styles.

(AS701, AS5056) was directed from AS5056 to AS701. It is easy to see that in this case we have a contradiction on the orientation of edge (AS1, AS1239).

Figure 3 shows an instance of the ToR problem that admits an orientation without invalid paths. Figures 4 and 5 show a possible orientation.

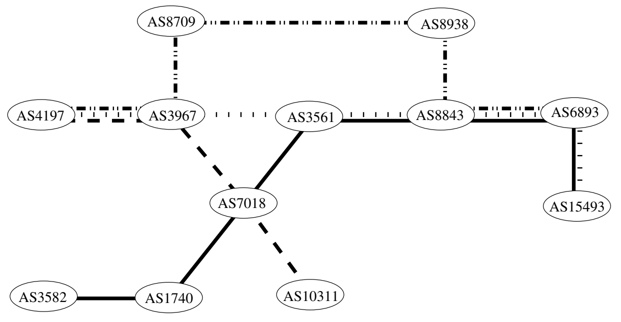


Fig. 3. An instance of the ToR problem that admits an orientation without invalid paths. The four paths of the instance are represented with different line styles.

B. Simplifying the problem

The Type-of-Relationship Problem is a minimization problem. In order to studying it, following a standard technique [11], we consider its corresponding decision version as follows.

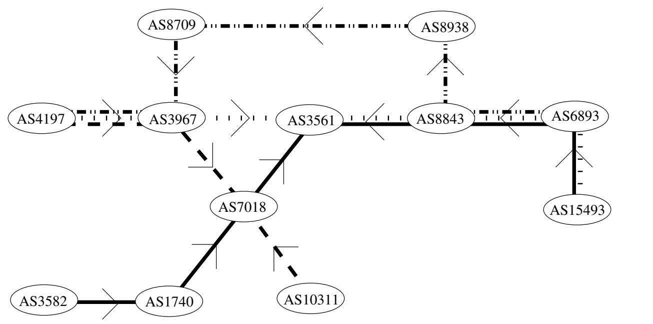


Fig. 4. An orientation for the graph of Figure 3. Note that all the paths are valid.

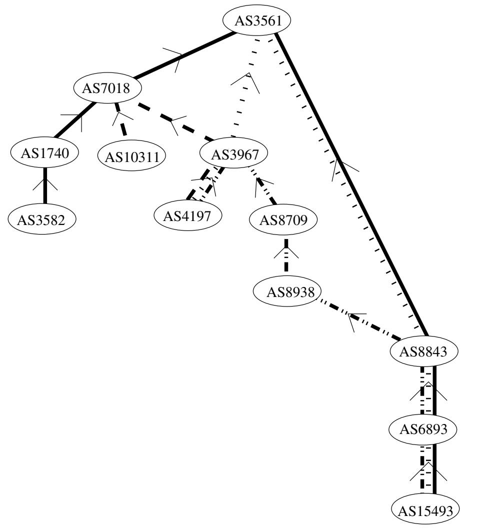


Fig. 5. The directed graph of Figure 4 is drawn in such a way to emphasize the hierarchical relationships induced by the orientation.

ToR-D Problem: Given an undirected graph G, a set of paths P, and an integer k, test if it is possible to give an orientation to some of the edges of G so that the number of invalid paths in P is at most k.

One of the ingredients that make the ToR-D problem difficult is the presence of both directed and undirected edges. Fortunately, the problem can be simplified by "ignoring" the undirected edges, without loosing its generality. Namely, the ToR-D problem admits a solution if and only if the following simpler problem admits one.

ToR-D-simple Problem: Given an undirected graph G, a set of paths P, and an integer k, test if it is possible to give an orientation to *all* the edges of G so that the number of invalid paths in P is at most k.

Notice that the ToR-D-simple problem considers Type 1 paths only.

In fact, consider an orientation of the edges of G that is a solution for the ToR-D-simple problem. It is clear that the same orientation is also a solution for the ToR-D problem. Conversely, consider an orientation of some of the edges of G that is a solution for the ToR-D problem and let (u, v) be an edge of G that is undirected. Consider any path p of P

through (u, v). Two cases are possible: either p is valid or p is invalid.

If p is valid (see Figure 6), then it is a Type 2 path and all the edges of p preceding u are forward edges, while all the edges of p following v are backward edges. If (u, v) is arbitrarily oriented, then the only effect on p is of transforming it from Type 2 to Type 1. Hence, the number of invalid paths does not increase. If p is invalid and (u, v) is arbitrarily oriented either it becomes valid or it remains invalid. In this case the number of invalid paths does not increase. The same process can be repeated on all the undirected edges, until an orientation of G that is a solution for ToR-D-simple is found.

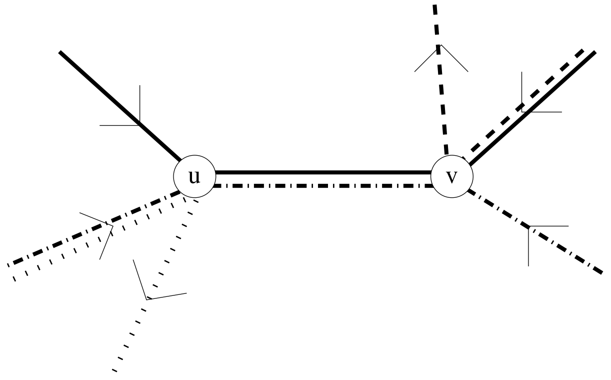


Fig. 6. An undirected edge (u, v) of a solution of the ToR-D problem. The two paths traversing (u, v) are represented one with a solid line and the other with a dot-dashed line. Both paths are valid.

To better understand the relation between the two problems, observe that the above consideration suggests that for each partial orientation of G that is a solution of ToR-D with n undirected edges there exist 2^n orientations that are a solution for ToR-D-simple.

Further, we can pick an orientation that is a solution for ToR-D-simple and consider it as a solution for ToR-D. Then, we can refine such a solution looking for edges whose orientation can be removed without increasing the number of anomaly paths. A necessary and sufficient condition, that is also easy to test, for removing the orientation of a single directed edge (u, v) is the following. Consider all the paths through (u, v) and all the edges following (u, v) in such paths. Edge (u, v) can be made undirected if such edges are all directed toward v.

The above discussion justifies a two steps approach where in the first step a solution is found for ToR-D-simple and in the second step peering edges are discovered.

III. Testing whether an AS Graph Admits a Hierarchical Structure without Path Anomalies

In Section II we have seen that the problem of detecting the types of relationships between ASes can be tackled by studying the ToR problem, its decision version ToR-D, and a simpler problem called ToR-D-simple. The relations among such problems have also been discussed. In this section we show that problem ToR-D-simple (and, consequently, ToR-D) can be solved efficiently when $k = 0$, that is when we want to check if G admits an orientation where all the paths are valid (i.e., there are 0 invalid paths).

A. Path anomalies and boolean formulas

Observe that a path p on G composed by the sequence of vertices $v_1, \ldots, v_n$ is of Type 1 if and only if it does not exist a vertex v_i ($i = 2, \ldots, n-1$) of p such that the two edges of p incident on v_i are directed away from v_i. Hence, to impose that p is valid it suffices to rule out such a configuration. Based on this observation ToR-D-simple can be mapped to a 2SAT problem [11].

In the 2SAT problem you are given a set X of boolean variables and a formula in conjunctive normal form. Such a formula is composed by clauses of two literals, where a literal is a variable or a negated variable. You are asked to find a truth assignment for the boolean variables in X so that the formula is satisfied.

The mapping of ToR-D-simple to 2SAT is a two step process. First, all the edges of G are arbitrarily (for example randomly) oriented. Second, a boolean formula is constructed so to represent the constraints that each path imposes on the orientation of G in order to be a path of Type 1. The construction is performed as follows.

- For each directed edge (v_i, v_j) of G a variable $x_{i,j}$ is introduced. A true value for $x_{i,j}$ means that, in the final orientation, (v_i, v_j) will be directed from v_i to v_j (that is, the direction of the initial arbitrary orientation will be preserved), while a false value means that (v_i, v_j) will be directed from v_j to v_i (that is, the direction of the initial arbitrary orientation will be reversed).
- Consider a path $p \in P$ and three consecutive vertices v_{i-1}, v_i, v_{i+1} of p. Four cases are possible, according to the arbitrary orientations that we have given to the edges between v_{i-1}, v_i, and v_{i+1}.
  - Both edges are directed toward v_i, i.e. such directed edges are (v_{i-1}, v_i) and (v_{i+1}, v_i). We introduce clause $x_{i-1,i} \vee x_{i+1,i}$.
  - Both edges are directed away from v_i, i.e. such directed edges are (v_i, v_{i-1}) and (v_i, v_{i+1}). We introduce clause $\overline{x}_{i,i-1} \vee \overline{x}_{i,i+1}$.
  - One edge is directed toward v_i and the other toward v_{i+1}, i.e. such directed edges are (v_{i-1}, v_i) and (v_i, v_{i+1}). We introduce clause $x_{i-1,i} \vee \overline{x}_{i,i+1}$.
  - One edge is directed toward v_{i-1} and the other toward v_i, i.e. such directed edges are (v_i, v_{i-1}) and (v_{i+1}, v_i). We introduce clause $\overline{x}_{i,i-1} \vee x_{i+1,i}$.

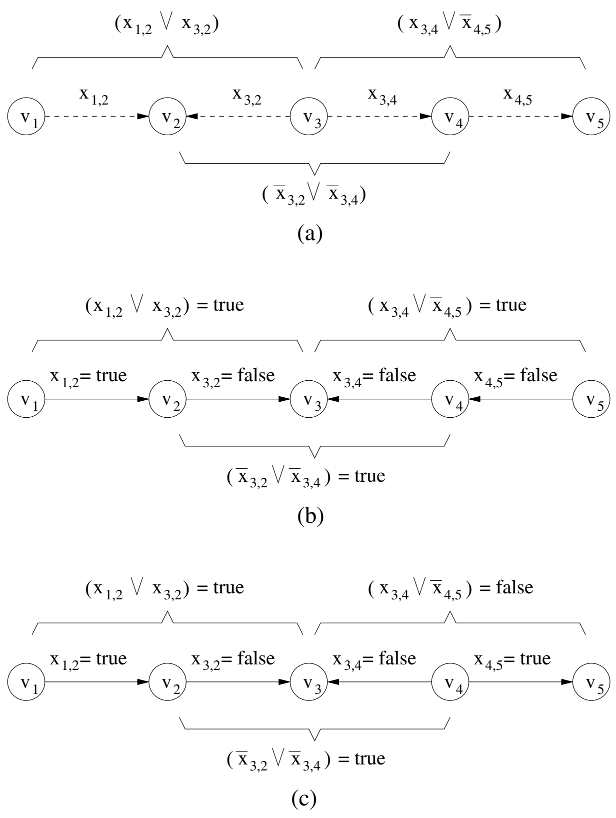
In this way we introduce $n-2$ clauses for each path of P with n vertices. We impose that all the constraints are simultaneously satisfied by considering the boolean "and" of all the clauses. Since each clause has two literals, we have mapped the ToR-D problem to a 2SAT formula.

As an example consider a path composed by five vertices $v_1, \ldots, v_5$ and suppose that the initial orientation step has

TABLE I

TELNET LOOKING GLASS SERVERS AND CORRESPONDING AS GRAPHS.

AS #	AS Name	Apr 18, 2001			Apr 6, 2002		
		# Vertices	# Edges	# Paths	# Vertices	# Edges	# Paths
1	Genuity	10,203	13,001	58,156	12,700	15,946	63,744
1740	CERFnet	10,007	13,416	70,830	not available		
3549	Globalcrossing	10,288	13,039	60,409	12,533	16,025	76,572
3582	U. of Oregon	10,826	22,440	2,584,230	13,055	27,277	4,600,981
3967	Exodus Comm.	10,387	18,401	254,123	12,616	21,527	339,023
4197	Global Online Japan	10,288	13,004	55,060	12,518	15,628	59,745
5388	Energis Squared	10,411	13,259	58,832	12,659	16,822	117,003
7018	AT&T	9,252	12,117	120,283	11,706	15,429	170,325
8220	COLT Internet	8,376	10,932	46,606	12,660	18,421	154,855
8709	Exodus, Europe	10,333	15,006	114,931	12,555	18,175	126,370



$(x_{1,2} \vee x_{3,2})$
$(x_{3,4} \vee \overline{x}_{4,5})$
$x_{1,2}$
$x_{3,2}$
$x_{3,4}$
$x_{4,5}$
v_1
v_2
v_3
v_4
v_5
$(\overline{x}_{3,2} \vee \overline{x}_{3,4})$
(a)
$(x_{1,2} \vee x_{3,2})$ = true
$(x_{3,4} \vee \overline{x}_{4,5})$ = true
$x_{1,2}$= true
$x_{3,2}$= false
$x_{3,4}$= false
$x_{4,5}$= false
$(\overline{x}_{3,2} \vee \overline{x}_{3,4})$ = true
(b)
$(x_{1,2} \vee x_{3,2})$ = true
$(x_{3,4} \vee \overline{x}_{4,5})$ = false
$x_{1,2}$= true
$x_{3,2}$= false
$x_{3,4}$= false
$x_{4,5}$= true
$(\overline{x}_{3,2} \vee \overline{x}_{3,4})$ = true
(c)

Fig. 7. (a) An initial orientation for a five vertices path and the boolean variables associated with its edges. The orientation shown in (b), which makes the path valid, corresponds to the truth assignment $x_{1,2} = true$, $x_{3,2} = false$, $x_{3,4} = false$, and $x_{4,5} = false$, which satisfies formula $(x_{1,2} \vee x_{3,2}) \wedge (\overline{x}_{3,2} \vee \overline{x}_{3,4}) \wedge (x_{3,4} \vee \overline{x}_{4,5})$ associated with the path. Conversely, the orientation shown in (c), which makes the path invalid, corresponds to the truth assignment $x_{1,2} = true$, $x_{3,2} = false$, $x_{3,4} = false$, and $x_{4,5} = true$, which does not satisfy the formula.

given to the edges of the path a direction as follows: (v_1, v_2), (v_3, v_2), (v_3, v_4), and (v_4, v_5). We have variables $x_{1,2}$, $x_{3,2}$, $x_{3,4}$, and $x_{4,5}$ (see Figure 7.a). Applying the above procedure we obtain the following 2SAT formula: $(x_{1,2} \vee x_{3,2}) \wedge (\overline{x}_{3,2} \vee \overline{x}_{3,4}) \wedge (x_{3,4} \vee \overline{x}_{4,5})$. Consider the truth assignment $x_{1,2} = true$, $x_{3,2} = false$, $x_{3,4} = false$, and $x_{4,5} = false$. It is easy to see that it satisfies the formula and that it corresponds to an orientation of the edges of the path toward vertex v_3 (see Figure 7.b). On the other hand, consider the truth assignment $x_{1,2} = true$, $x_{3,2} = false$, $x_{3,4} = false$, and $x_{4,5} = true$. It is easy to see that it does not satisfy the formula and that it corresponds to an orientation of the edges of the path that is not consistent with Type 1 (see Figure 7.c).

B. Computational aspects

Problem 2SAT may be efficiently solved by using the well known result in [12] that maps 2SAT into a problem on a suitable directed graph G_{2SAT}. Observe that G and G_{2SAT} are different graphs.

Although this result is clearly illustrated in the literature, we give here a brief description of it, that will help the reader to better understand the algorithms described in Sections IV, and VI.

Graph G_{2SAT} has two nodes for each boolean variable x of 2SAT, corresponding to its two literals x and $\overline{x}$. Further, for each clause of the form $l_1 \vee l_2$, where l_1 and l_2 are literals, the two directed edges $(\overline{l}_1, l_2)$ and $(\overline{l}_2, l_1)$ are introduced. Intuitively, edge $(\overline{l}_1, l_2)$ represents the logical implication $\overline{l}_1 \rightarrow l_2$, while edge $(\overline{l}_2, l_1)$ represents $\overline{l}_2 \rightarrow l_1$. Problem 2SAT admits a solution if and only if for no variable x there is a directed cycle in G_{2SAT} containing both x and $\overline{x}$ (i.e. a logical contradiction).

Testing, for each variable, if there exists a cycle containing its two literals can be quite time consuming. However, fortunately, the problem of testing for all the variables in 2SAT whether such a cycle exists in G_{2SAT} can be efficiently solved by computing [13] the *strongly connected components* of G_{2SAT} and by testing for each variable if x and $\overline{x}$ are in the same strongly connected component. We recall that a strongly connected component of a directed graph is a maximal set of vertices such that for each pair u, v of vertices of the set there exists a directed path from u to v and vice versa. Computing the strongly connected components of a directed graph can be done in time linear in the size of the graph [13].

From a theoretical point of view, it comes out that ToR-D-simple (and, as a consequence, ToR-D) with $k = 0$, i.e. the problem of deciding if a graph G of n vertices and m edges admits an orientation so that all the paths of a set P are valid, can be solved in $O(n + m + q)$ time, where q is the sum of the lengths of the paths of P.

More practically, we have implemented the above algorithm by exploiting a facility from the Leda [14] software library that efficiently computes the strongly connected component of a directed graph and that labels each vertex of the graph with an integer that identifies the component it belongs to. Hence, testing for a variable x if x and $\overline{x}$ are in the same strongly connected component is performed by testing whether they have the same label.

C. Experiments

This section illustrates the first group of experiments of this paper. Such experiments have the purpose of understanding if at least for partial views of the Internet graph the ToR problem admits a solution without invalid paths. This is important, in our opinion, at least for the following reason. Even if it is unlikely that the entire Internet AS graph could be classified in terms of customer-provider and peer-to-peer relationships without exceptions (and we will see evidence of this in the remainder of this paper), it is unclear if this is possible for what is visible from a specific observation point ("vantage point" in [2]) of the network.

The test bed consists of BGP data sets obtained as follows. Each data set is extracted from the BGP routing table of a Looking Glass server. First, the output of the "show ip bgp" command is collected. Second, a file of AS paths is computed by discarding the prefix column and all the BGP attributes different from the AS path. Duplicate ASes arising from *prepending* [3] are removed in each path. Note that duplicated paths may be present in the set.

There are many Looking Glass servers on the Internet and it is very difficult to say which are the most representative. In order to compare our work with previous results, we have chosen to use the collection of ten BGP data sets obtained from Telnet Looking Glass servers already adopted as a test bed in [2]. Such test beds are periodically collected and publicly distributed by the authors [15].

For each data set we have constructed a different AS graph (a partial view of the global AS graph) by using only the adjacencies contained in the AS paths of the specific data set. Table I shows the main features of the graphs constructed from the ten data sets. Note that values of Tables I and IV of [2] and values computed from data available in [15] (and that are presented in the aforementioned Table I of this paper) appear to be slightly different.

Table II shows the results of the experiments. Observe that for all the partial views, but the one of the University of Oregon server [16], the ToR problem admits a solution without invalid paths. In fact, the server of the University of Oregon is not just a Looking Glass that gives a view of Internet from a specific point of observation, but it offers an integrated view obtained from 52 peering sessions with routers spread on 39 different ASes. This clearly indicates that integrating information from different points of view makes the problem much more difficult.

Figure 8 shows six rows extracted from the routing table of the U. of Oregon dated Apr 18, 2001. Observe that the six paths are exactly those used in Figure 2 to give an example of an instance of the ToR problem that does not admit an orientation without invalid paths.

TABLE II

TESTING IF THE ToR PROBLEM HAS A SOLUTION WITHOUT INVALID PATHS FOR SEVERAL BGP ROUTING TABLES.

AS #	AS Name	Orientable w/o anomalies	
		Apr 18, 2001	Apr 6, 2002
1	Genuity	yes	yes
1740	CERFnet	yes	not available
3549	Globalcrossing	yes	yes
3582	U. of Oregon	no	no
3967	Exodus Comm.	yes	yes
4197	Global Online J.	yes	yes
5388	Energis Squared	yes	yes
7018	AT&T	yes	yes
8220	COLT Internet	yes	yes
8709	Exodus, Europe	yes	yes

It is worth noting that we have conducted all the experiments on a PC Pentium III with 1 GB of RAM. Each of the above experiments required a few seconds of computation time.

IV. COMPUTING THE AS RELATIONSHIPS

In Section III we have seen how problem ToR-D can be solved efficiently when $k = 0$, that is when we want to check if G admits an orientation where all the paths are valid (i.e., there are 0 invalid paths). We can do that solving a simpler problem, called ToR-D-simple.

In this section we deal with the problem of determining the relationships between ASes in the assumption that ToR-D admits a solution without anomalies. Essentially, this is a two steps process. In the first step an orientation that solves ToR-D-simple is computed. In the second step peering relationships are discovered by examining the solution computed for ToR-D-simple.

A. Finding an orientation for ToR-D-simple

If a solution for ToR-D-simple exists, computing it is an easy task. Since we mapped ToR-D-simple to 2SAT, we can find a solution to ToR-D-simple by computing a truth assignment for the boolean variables of the corresponding 2SAT instance. A standard method [12] for computing such assignment is the following. A function $f(v)$ can be computed for all the vertices of the graph G_{2SAT} associated with 2SAT (see Section III-B) such that, for any two vertices u and v, if there exists a directed path from u to v, then $f(u) \leq f(v)$. A true value is assigned to variable x if $f(x) > f(\overline{x})$, a false value otherwise. The satisfiability of 2SAT guarantees that $f(x) \neq f(\overline{x})$.

Function f can be efficiently computed by exploiting the decomposition of the graph into strongly connected components and by computing a special ordering, called *topological sorting* [14], on the directed acyclic graph of the components.

Of course, an instance of the problem ToR-D-simple may admit several different solutions. The structure of the problem constraints some variables to have the same truth values in all

Network	Next Hop	Path
200.1.225.0	167.142.3.6	5056 701 6461 4926 4270 4387 i
200.10.112.0/23	167.142.3.6	5056 701 4926 4926 4926 6461 2914 174 174 174 174 14318 i
204.71.2.0	203.181.248.233	7660 1 5056 701 11334 i
213.172.64.0/19	167.142.3.6	5056 1239 1 1755 1755 1755 1755 3216 13099 i
200.33.121.0	167.142.3.6	5056 1 1239 8151 i
204.71.2.0	144.228.241.81	1239 5056 701 11334 i

Fig. 8. Six rows extracted from the BGP routing table of the U. of Oregon dated Apr 18, 2001. Each orientation of the edges of the corresponding graph yields at least one invalid path.

the solutions, while other variables may assume any true/false assignment. Coming back to problem ToR-D-simple, this means that some edges have a constrained customer-provider orientation, while others may assume different orientations.

Interestingly, the proposed approach permits to "explore" the solutions space. Namely, if some knowledge is available on the customer-provider relationships between ASes, it is easy to force the solution to respect such constraints. For example, suppose to know in advance that AS v_i is a customer of AS v_j and suppose that in the initial arbitrary orientation edge (v_i, v_j) is directed from v_i to v_j. We can impose that the solution respects the constraint by adding to the 2SAT formula associated with Problem ToR-D-simple the clause $(x_{i,j} \vee x_{i,j})$. Of course, adding constraints to the problem decreases the size of the solution space and may lead to unsatisfiable instances.

B. Discovering the peering relationships

A solution for the ToR-D-simple problem provides an orientation for all the edges of the AS graph (customer-provider relationships). However, as described in Section II-B, it is possible to refine the obtained solution reintroducing peering relationships. In such a section a sufficient condition has been given for modifying a directed edge into an undirected edge still having a solution for ToR-D.

Several different criteria can be adopted to measure the quality of a solution once peerings are reintroduced. For example, one could say that a solution is especially interesting if many peering have been discovered. Unfortunately, it can be shown that, given a solution for a ToR-D-simple instance, i.e., with no peerings, the problem of producing a solution for the corresponding ToR-D instance that maximizes the number of the peering edges is a hard one.

We prove it using a reduction from the INDEPENDENT-SET problem, in which you are given a graph with nodes in N and arcs in A and you are asked to find a subset of the nodes of size k such that no two nodes of the subset are adjacent. To build the instance of the ToR-D-simple problem corresponding to the instance of the INDEPENDENT-SET problem we introduce an edge (v_i, v_{top}) for each node $n_i \in N$ and we introduce a path v_i, v_{top}, v_j for each arc $(n_i, n_j) \in A$. The edges of the ToR-D-simple instance can be directed toward vertex v_{top} in order to have a solution with no invalid path. It can be easily shown that the problem of reintroducing k peering edges without increasing the number of invalid paths is equivalent to the problem of finding an independent set of size k. We omit the details of the proof for the sake of brevity.

V. The Difficulty of Minimizing Path Anomalies

The ToR problem was conjectured to be NP-complete in [2]. In Section III we have shown that finding a solution with zero invalid path (provided that it exists) is a tractable problem. In this section we show that the ToR problem is NP-complete in the general case, that is, when it does not admit an orientation without invalid paths. In order to prove that the ToR problem is NP-hard we reduce the NP-complete problem MAX2SAT to it.

In the remaining part of this section, following a standard technique when dealing with optimization problems [11], we refer to their decision versions. For ToR we already defined in Section II the ToR-D and ToR-D-simple problems (we will use the latter one). As for MAX2SAT, its decision version of MAX2SAT-D can be defined as follows. You are given a set X of boolean variables and a collection C of disjunctive clauses, each one of 2 literals, where a literal is a variable or a negated variable. You are asked to find a truth assignment for the boolean variables in X so that the number of unsatisfied clauses of C is at most k, where k is a positive integer.

Given an instance of the MAX2SAT-D problem, we will produce an instance of the ToR-D-simple problem, such that an orientation with k invalid paths exists iff an assignment with k unsatisfied clauses of MAX2SAT-D can be found. For each variable $x_i \in X$ we introduce two vertices x'_i and x''_i. For each clause $l_1 \vee l_2$ we introduce a path of four vertices as follows. If l_1 is the negated literal of variable x_i, then the first two vertices of the path will be x''_i and x'_i, otherwise they will be x'_i and x''_i. Similarly, if l_2 is the negated literal of variable x_j, then the last two vertices of the path will be x'_j and x''_j, otherwise they will be x''_j and x'_j.

Figure 9 shows an example of an instance of the MAX2SAT-D problem and the corresponding instance of the ToR-D-simple problem.

Given an orientation for the edges of the graph, if edge (x'_i, x''_i) is directed from x'_i to x''_i, then we associate a true value with the corresponding boolean variable x_i, otherwise we associate a false value. Observe that, given an orientation for the edges of the graph, if the first edge of the four-vertex path is directed toward the first vertex of the path, then the first literal of the corresponding clause is false. Analogously, if the last edge of the path is directed toward the last vertex of the path, then the second literal of the clause is false.

If a path is valid, then its first and its last edges are not simultaneously directed toward the first vertex and the last vertex, respectively. It follows that, if a path is valid, the corresponding clause is satisfied. Thus, an orientation for the

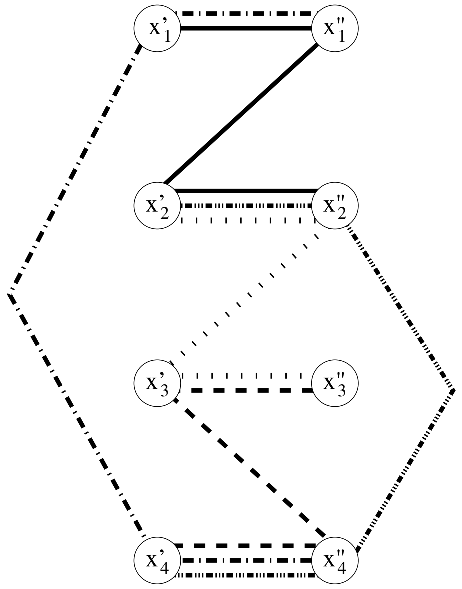


Fig. 9. The instance of the ToR-D-simple problem corresponding to the instance $(x_1 \vee \overline{x}_2) \wedge (x_2 \vee \overline{x}_3) \wedge (x_2 \vee x_4) \wedge (\overline{x}_1 \vee \overline{x}_4) \wedge (\overline{x}_3 \vee x_4)$ of the MAX2SAT-D problem.

edges of the graph with k invalid paths corresponds to an assignment of the boolean variables with k unsatisfied clauses.

Conversely, suppose to have an assignment for the boolean variables in X that leaves k clauses of the MAX2SAT-D instance unsatisfied. If variable x_i is positive, direct edge (x'_i, x''_i) toward x''_i, otherwise direct it toward x'_i. Each satisfied clause corresponds to a four vertex path whose first and last edge are not simultaneously directed toward its first and last vertices, respectively, and an orientation for the intermediate edge of the path can be easily found so that the path is valid. Thus, an assignment for the boolean variables that leaves k clauses of the MAX2SAT-D instance unsatisfied corresponds to an orientation of the ToR-D-simple problem with k invalid paths.

Since it can be shown that the ToR-D-simple problem belongs to the class NP (it is easy to count the invalid paths yielded by a given orientation), it follows that the ToR-D-simple problem is NP-complete.

Observe that, although we used a reduction of the MAX2SAT-D problem to show the NP-hardness of the ToR-D-simple problem, an instance of the ToR-D-simple problem can not be always mapped to an instance of the MAX2SAT-D problem (for example, when two paths have one internal edge in common).

VI. Heuristics for Computing the AS Relationships

In Section V we have seen that the ToR-D problem is computationally hard and in Section III we have seen that, even if portions of Internet admit a hierarchical structure without anomalies, when the data set becomes large, such a "strong" structure does not exist (see, e.g., the AS 3582 in Table II).

This section aims at giving a method for discovering the AS relationships in a big chunk of Internet with a small number of invalid paths. Observe that, even if heuristics are known for solving the MAX2SAT problem (see, e.g., [17]), they cannot be straightforwardly applied to ToR-D. In fact, maximizing the number of satisfied clauses of the 2SAT formula does not necessarily imply maximizing the number of valid paths. Another approach would be to reduce ToR-D to a problem called Maximum Number of Satisfiable Formulas, where a collection of formulas in conjunctive normal form is given, and the target is to maximize the number of satisfied formulas. However, that problem has been shown to be not approximable in [18] and we were not able to find in the literature effective and efficient heuristics for that problem.

As a reference data set, with the purpose of comparing our results with previous contributions, we consider the same portion of Internet taken into account in [2]. Namely, we consider the union of all the paths of the Telnet Looking Glasses of Table I (version of Apr 18, 2001). The total number of paths of the data set is $3,423,460$, involving $10,916$ ASes. The graph of the adjacencies between ASes contains $23,761$ edges. We measure the effectiveness of our heuristics against such quite large data set.

The target of the proposed heuristics is the computation of a maximal set of paths (subset of the given set of paths) such that ToR-D with $k = 0$ admits a solution. A set of paths is *maximal* if no path can be added to the set without introducing anomalies.

A simple strategy for computing a maximal set of paths is the following. Starting from the empty set, add all the paths one-by-one, each time testing if the set admits an orientation without anomalies. The test can be performed in linear time by exploiting the algorithm presented in Section III. If the insertion of a path makes the set not orientable, then it is discarded, otherwise it is added to the set. At the end of the process we have a maximal set of paths. However, this simple strategy is unfeasible. In fact we would have to run the testing algorithm millions of times. Even if each run takes one second, we could wait a couple of weeks to have the maximal set.

Motivated by the above discussion, we propose a two steps approach. First, we compute a very large (albeit not maximal) set of valid paths with an ad-hoc technique. Second, we check if the discarded paths can be reinserted with the method described above.

The computation of the initial very large set of valid paths is performed as follows. Initialize P with the set of all the paths.

1) Construct the $G_{2\mathsf{SAT}}$ graph considering all the adjacencies of P.
2) Set-up the following data structure: for each undirected edge (v_i, v_j) of the AS adjacency graph keep the number of paths traversing (v_i, v_j); call it *covering* of (v_i, v_j).
3) Compute the strongly connected components of $G_{2\mathsf{SAT}}$ (e.g., with the algorithm in [13]).
4) Identify each variable x such that x and $\overline{x}$ are in the same strongly connected component of $G_{2\mathsf{SAT}}$.

5) Select among those variables the variable $x_{i,j}$ whose corresponding edge (v_i, v_j) has the smallest covering and remove all the paths that cover such an edge from P.

Execute steps (1) through (5) until no strongly connected component contains both the literals of the same variable.

Observe that at each iteration, since we remove all the paths traversing a specific edge of the AS graph, the literals associated with such an edge disappear from $G_{\textsf{2SAT}}$.

In our data set the starting $G_{\textsf{2SAT}}$ graph contains $47,522$ nodes and $375,100$ edges. It contains one strongly connected component with $2,156$ literals and $12,570$ edges. The other components contain just one literal. The set of valid paths computed during the first step contains $3,423,460$ paths. During the second step $222,764$ paths have been re-inserted without causing anomalies. The final maximal set of paths contains $3,399,389$ paths.

After computing a maximal set of paths we have computed an orientation for the edges of the AS graph obtained from those paths, using the technique illustrated in Section IV. A fragment of the computed orientation has been used already in this paper for the example of Figures 4 and 5. Further, following the experimental guideline of [2] we have done two types of checks of the quality of such orientation:

1) We checked how many paths of the original ten data sets are valid and the percentage of invalid paths.
2) We checked how good is the computed orientation against four additional data sets that were not input of the heuristic algorithm. Such extra group of data sets is, again, available from [15] and contains data from AS1755, AS2516, AS2548, and AS6893.

TABLE III
COMPARISON BETWEEN THE HEURISTICS PRESENTED IN THIS PAPER AND THE STATE OF THE ART.

AS #	AS Name	Anomalies % ([2])	Anomalies % (this paper)
1	Genuity	0.65	0.45
1740	CERFnet	n.a.	0.36
3549	Globalcrossing	n.a.	0.13
3582	U. of Oregon	n.a.	0.57
3967	Exodus Comm.	n.a.	0.42
4197	Global Online J.	n.a.	0.46
5388	Energis Squared	n.a.	0.46
7018	AT&T	0.63	0.21
8220	COLT Internet	n.a.	0.22
8709	Exodus, Europe	n.a.	0.21
1755	Ebone	2.89	1.52
2516	KDDI	8.97	4.95
2548	MaeWest	1.49	0.19
6893	CW	2.92	0.64

Table III shows that the heuristics described above leaves a very small percentage of invalid paths. In particular, it performs significantly better, in terms of invalid paths, than the cutting edge heuristics of [2]. The invalid paths are about halved for ASes 1, 1755, and 2516, are about one third for AS 7018, and are one fourth or less for ASes 2548 and 6893. These results are, in our opinion, even stronger if we consider that the percentages of anomalies provided by [2] do not count as invalid Type 2 paths containing two consecutive undirected edges instead of one [19]. The basis of such relaxation of the model is that two ASes may have an "indirect peering", that is a peer-to-peer relationship through an intermediate one.

Using the condition discussed in Section II-B we have also found edges that can be made undirected still preserving the quality of the solution and have found $3,936$ edges that can be considered as candidates for being peering edges.

It is worth noting that we have conducted all the experiments with the same platform described in Section III. The above experiment, involving $3,423,460$ paths, required a computation time of about 10 hours.

VII. Conclusions and Open Problems

In this paper we introduced a novel approach for computing the relationships between Autonomous Systems starting from a set of AS paths, so that the number of invalid paths is kept small. Also, we proved that the corresponding minimization problem is NP-complete in the general case (as conjectured in [2]).

Our approach consists of mapping the problem into a 2SAT formulation, which can be exploited in several ways. For example, a solution for the 2SAT formulation can be found in linear time, if it exists, determining a solution to the original problem without invalid paths. Also, we take advantage of the theoretical insight gained with the 2SAT formulation to conceive new heuristics for the general case which we experimentally prove to be more effective than previously presented approaches.

Further details on the experiments, the used source code, and the data sets are available from the Website `http://www.dia.uniroma3.it/~compunet` and in [20].

The classification of AS relationships in the Internet is a hot research topic. Its relevance is confirmed by the interest of other research groups on the same subject. In [21] analogous and independently discovered results concerning the time complexity of the general problem and the linearity in the case of all valid paths are shown. However, while such work puts more emphasys on the approximability of the problem, we focus more on the engineering and the experimentation of an effective heuristic approach.

Several problems remain open. We think it is interesting to understand why the portion of the Internet seen from a single observation point is very often orientable without invalid paths. Is that a matter of size or there is a more subtle property that can be formally studied? Also, the recognition of AS relationships can probably take advantage of further information provided by the BGP routing tables, for example, the size of the prefixes. Can this lead to a prefix-driven formulation of the problem instead of the as-path driven formulation adopted until now? Further, it could be interesting to improve existing tools for the visualization of the AS graph (see, e.g., [22]) in order to provide information about the relationships between ASes.

Acknowledgments

We are grateful to the authors of [2] for their help. Also, we would like to thank Andrea Vitaletti and Debora Donato for interesting conversations.

References

[1] L. Gao, "On inferring autonomous system relationships in the internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 733–745, Dec 2001.

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

[3] J. W. Stewart, *BGP4: Inter-Domain Routing in the Internet*. Reading, MA: Addison-Wesley, 1999.

[4] C. Alaettinoglu, "Scalable router configuration for the internet," in *Proc. IEEE IC3N*, October 1996.

[5] G. Huston, "Interconnection, peering, and settlements," in *Proc. INET*, June 1999.

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

[7] R. Govindan and H. Tangmunarunkit, "Heuristics for internet map discovery," in *Proc. IEEE INFOCOM 2000*, March 2000.

[8] W. Theilmann and K. Rothermel, "Dynamic distance maps of the internet," in *IEEE INFOCOM 2000*. Tel-Aviv, Israel: IEEE, March 2000. [Online]. Available: citeseer.nj.nec.com/theilmann00dynamic.html

[9] Z. Ge, D. R. Figueiredo, S. Jaiswal, and L. Gao, "On the hierarchical structure of the logical internet graph," in *Proc. SPIE ITCom 2001*, 2001.

[10] L. Gao, T. G. Griffin, and J. Rexford, "Inherently safe backup routing with BGP," in *IEEE INFOCOM 2001*, Apr 2001, pp. 547–556.

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

[12] B. Aspvall, M. F. Plass, and R. E. Tarjan, "A linear-time algorithm for testing the truth of certain quantified boolean formulas," *Information Processing Letters*, vol. 8, no. 3, pp. 121–123, 1979.

[13] K. Mehlhorn, *Data Structures and Algorithms*. Springer Publishing Company, 1984, vol. 1-3.

[14] K. Mehlhorn and S. Näher, *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[15] "Characterizing the internet hierarchy from multiple vantage points," `http://www.cs.berkeley.edu/~sagarwal/research/BGP-hierarchy/`.

[16] "University of Oregon RouteViews project," `http://www.routeviews.org`.

[17] U. Feige and M. Goemans, "Approximating the value of two prover proof systems, with applications to max 2sat and max dicut," in *Proc. of 3rd Israel Symposium on the Theory of Computing and Systems*, 1995, pp. 182–189.

[18] V. Kann, "On the approximability of NP-complete optimization problems," Ph.D. dissertation, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, 1992.

[19] L. Subramanian, personal communication.

[20] G. Di Battista, M. Patrignani, and M. Pizzonia, "Computing the types of the relationships between autonomous systems," Dipartimento di Informatica e Automazione, Università di Roma Tre, Technical Report RT-DIA-73-2002, July 2002.

[21] T. Erlebach, A. Hall, and T. Schank, "Classifying customer-provider relationships in the internet," ETH Zurich, Technical Report TIK-145, July 2002.

[22] A. Carmignani, G. Di Battista, W. Didimo, F. Matera, and M. Pizzonia, "Visualization of the high level structure of the internet with hermes," *J. of Graph Algorithms and Applications*, vol. 6, no. 3, pp. 281–311, 2002.

Appendix 7.5

Archives of BGP Updates: Integration and Visualization[*]

Giuseppe Di Battista, Federico Mariani,
Maurizio Patrignani, and Maurizio Pizzonia
Dip. Informatica e Automazione
Università di Roma Tre
Via della Vasca Navale 79, 00146 Roma, Italy
{gdb,mariani,patrigna,pizzonia}@dia.uniroma3.it

Abstract

The possibility of analyzing updates exchanged between BGP talkers is crucial for several operational and research purposes. To give a few examples, they can be used to investigate the stability of specific routes, to monitor the effects of faults, and to analyze the behavior of the entire network in the presence of particular events. Several archives of BGP conversations, such as the Routing Information System of the RIPE and the Oregon Route Views database, give an answer to this need for information. We describe a work that aims at integrating the data from different BGP-update sources and at presenting such data with graph-based visualization techniques. We address both technological issues related to the data integration and user-interaction issues originated from the necessity of visualizing data that change over time.

1 Introduction

The exploration and visualization of the Internet attracts an increasing research interest, motivated by the growing size of the network and the significant impact that connectivity has on social and economic activities.

Contributions in this field can be roughly classified with respect to the granularity of the data they consider.

For example, Hermes [6, 1] visualizes the information provided by the Internet Registries about the interconnections between Autonomous Systems (ASes), showing their peerings and mutual policies.

At a more granular level, Mercator [9], Skitter [11], and Rocketfuel [15] compute maps from data collected by means of network discovery, showing interconnections among routers. Another system using traceroutes for probing the network is described in [5].

In this paper we present BGPlay, a system that shows the routing at the interdomain level acquired from communications between BPG-speaking peers [16]. BGPlay integrates the data from different archives of BGP updates and presents such data with graph-based visualization techniques.

Currently, BGPlay integrates the BGP updates collected by the Routing Information Service [3] of the RIPE and those collected by the Oregon Route Views project [4]. BGPlay aims at giving a highly intuitive visual representation of the status of the routing at a specific time, and of its evolution in a given time interval. In order to show such an evolution the system relies on an animation which shows how the BGP-paths evolve over time while preserving the user mental map [14, 7].

BGPlay has been designed to satisfy both operational and research needs. To give a few examples, it can be used to investigate the stability of specific routes, to monitor the effects of faults, and to analyze the behavior of the entire network in the presence of particular events. Instabilities and faults of interdomain routing have been the subject of recent research (see for example [13, 10, 8, 12]).

The paper is organized as follows. In Section 2 the data sources used by BGPlay are described. The overall architecture of the system is discussed in Section 3, while Section 4 and Section 5 illustrate how data are collected and visualized. Finally, Section 6 contains our conclusions and future work.

2 Archives of BGP Data

The Oregon Route Views (ORV) project (see Fig. 1.a) provides a service for Internet operators to obtain real-time

[*] Work partially supported by European Commission - Fet Open project COSIN - COevolution and Self-organisation In dynamical Networks - IST-2001-33555; by ”Progetto ALINWEB: Algoritmica per Internet e per il Web”, MIUR Programmi di Ricerca Scientifica di Rilevante Interesse Nazionale; and by “The Multichannel Adaptive Information Systems (MAIS) Project”, MIUR Fondo per gli Investimenti della Ricerca di Base.

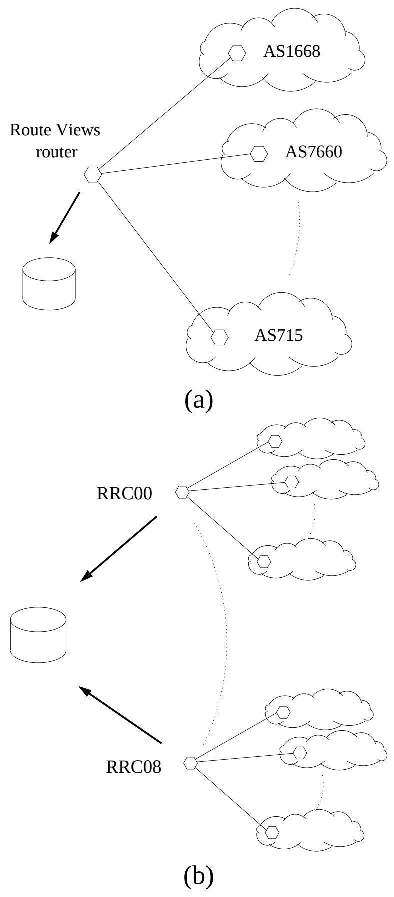


Figure 1. The architecture of ORV (a) and of RIS (b).

information about the global routing system from the perspectives of several different locations around the Internet. Currently, the BGP Route Views router has more than 60 multi-hop eBGP peering sessions with routers of Internet service providers.

ORV collects, without providing any transit service, the BGP updates coming from its peers. Such updates allow to reconstruct the evolution of the best routes (at the AS level) adopted by each peer. ORV data archives contain both snapshots of the global routing information base of ORV at different instants of time and sequences of BGP updates. The data are made available in the MRT [2] format.

The Routing Information Service (RIS) of the RIPE (see Fig. 1.b) collects historical information about Internet routing by using Remote Route Collectors (RRC) at different locations around the world. Such information is integrated into a comprehensive view. An RRC of the RIS is a BGP speaking router that only collects BGP routing information. The collected raw data is regularly transferred to a central storage area at the RIPE NCC in Amsterdam. Each RRC is denoted by a string with format RRCxx, where xx is a two digit number. Essentially, each RRC behaves analogously to the ORV router.

The RIS offers to the users several query facilities. For example, the BGP Routing Hot Spot Utility generates lists of prefixes, originating from a specified AS, for which high BGP announcement activity has been observed by some RRC. Also, the ASInuse facility determines when an AS-number last appeared in the global routing table collected by the RIS.

In this paper, we are interested in a simpler RIS query, that can be performed using the Search by Prefix utility. It allows to specify a prefix, a time interval, and a set of RRCs in order to search the RIS database. It outputs a list of BGP updates recorded by the selected RRCs in the prescribed time interval. It also outputs the status of the Local Routing Information Base (Loc-Rib) for the selected prefix in an instant of time that is "related" to the prescribed time interval.

3 System Architecture

The architecture of BGPlay is based on the following main choices.

- The user interacts with the system by means of a browser. A query identifies a time interval and a prefix to be monitored. The data sources to be used can be selected in a set of possible alternatives.

- The results of the query, i.e. the changes in the BGP routing observed during the time interval, are visualized by an "animation." Such animation relies on Graph Drawing techniques [7]. When the animation is started, a graph-like representation shows the routing at the beginning of the interval. During the animation the graph changes according to the observed BGP updates.

- The BGP data exploited to answer the user queries are partly fetched on-line at the moment they are needed and are partly locally stored. Namely, since the RIS provides a Web query facility we access those data on-line. Also, since the Route Views Project does not have a Web query facility for historical data, we periodically copy part of them locally. Currently, because of limitations on the available storage space, we maintain a copy of a limited (the most recent) period of time.

- The service is based on a client-server architecture, where the server computes the result of queries and the client is an applet running on the user's browser. We decided of using an applet instead of producing on the server standard jpeg or gif figures because of the graphical complexity of the animation.

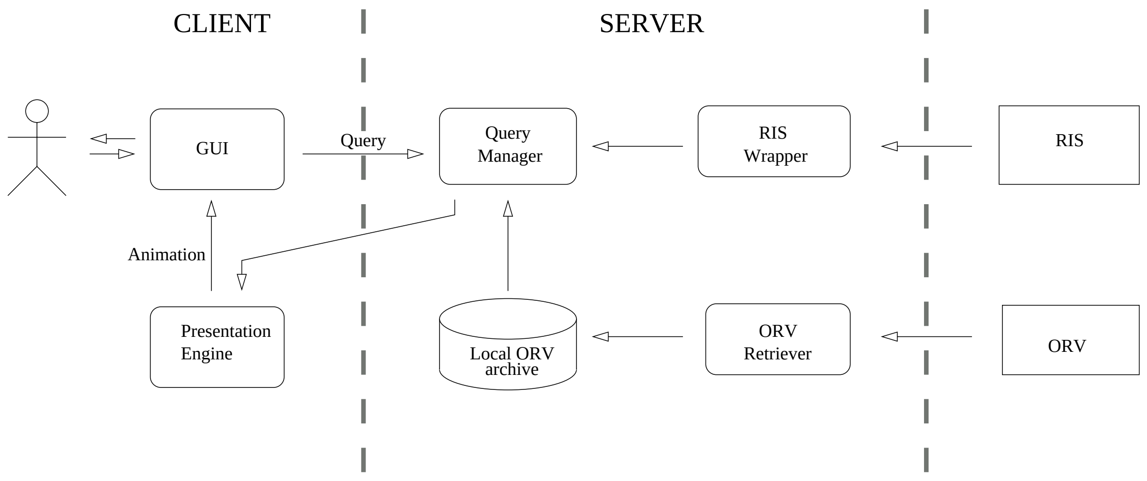


Figure 2. The system architecture.

Figure 2 illustrates the main components of the architecture:

Graphic User Interface. Provides the user with several tools for interacting with the visual representation of the routing information. For example: a *time panel* shows the time location of the most important events in the selected time interval, *animation buttons* allow to step over the events, and an *event display* gives all the available details about the currently visualized event.

Presentation Engine. Computes the layouts of the graphs representing the evolution of the routing. Further, is able to change such graph layout according to the routing events, in such a way that the user always perceives "smooth" variations. It exploits Graph Drawing methodologies and techniques.

RIS Wrapper. Queries the RIS Search by Prefix utility and retrieves the corresponding results in terms of BGP routing tables and updates.

ORV Retriever. Periodically retrieves routing tables and updates available at the ORV raw data archives in the MRT format. Updates the Local ORV Archive removing the oldest data.

Local ORV Archive. A relational database (currently, MySQL technology) storing the ORV data. Observe that, while all the RIS data refer to the UTC (Coordinated Universal Time), the ORV data refer to its local time. In BGPlay we refer to the UTC.

Query Manager. Gets a query description from the User Interface, drives the RIS Wrapper, accesses the Local ORV Archive, computes the result of the query, and delivers the result to the Presentation Engine.

An analysis of the workload and of the generated traffic lead us to a decomposition of the system in which the Presentation Engine is located on the client side, together with the Graphical User Interface.

4 Query Processing

In order to answer a query of the client, the Query Manager has to retrieve the BGP updates falling in the specified time interval. As mentioned in Section 3 two types of retrieval are involved: retrieval from the local database of ORV updates and on-line retrieval, performed by the RIS Wrapper, from the RIS Service.

The scenario is more precisely depicted in Fig. 3, where the starting and ending instants of the time interval are called t_1 and t_2, respectively. The purpose of the Query Manager is to compute:

- the status of the ORV RIB in t_1;
- the status of the RIS RIB in t_1; and
- the sequence of updates collected by ORV and RIS between t_1 and t_2.

Unfortunately, it is unlikely to have at disposal the two aforementioned RIBs at time t_1. Also, the possibilities of retrieving such information from the two data archives are quite different.

On one side, ORV offers a snapshot of its RIB every (about) two hours. Hence, we have to determine which is the last available ORV RIB before t_1. Once such RIB has been obtained (suppose it corresponds to a snapshot taken at time $t_0 \leq t_1$), we perform the following operations. First, the rows of the RIB corresponding to the given prefix p are

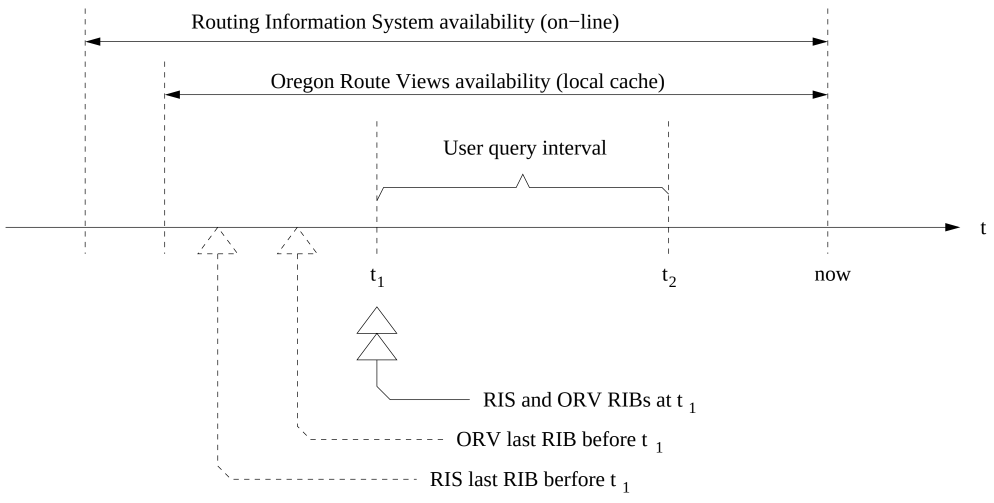


Figure 3. Available and computed RIBs.

extracted. Second, the updates occurring in the time interval t_0, t_1 are taken into account in order to compute the portion of the RIB corresponding to p at time t_1.

On the other side, the situation for RIS is more complex. Namely, as shown in Fig. 4, a query sent to the on-line RIS interface involving the time interval t'_1, t'_2 yields two types of data:

- the updates collected between t'_1 and t'_2 and
- the RIS RIB at time 23:59 of the day in which t'_1 falls.

Hence, it is not possible, for the Query Manager, to simply activate the RIS Wrapper with a query interval such that $t'_1 = t_1$ and $t'_2 = t_2$.

Because of the above discussion, the Query Manager asks the RIS Wrapper to perform a query with time interval t'_1, t'_2, where $t'_2 = t_2$ and t'_1 is the time 23:59 of the day preceding the one in which t_1 falls. Afterwords, analogously to what happens for ORV, the RIB is filtered saving only the portion concerning p. Then, the updates occurring in the time interval t'_1, t_1 are taken into account in order to compute the portion of the RIB corresponding to p at time t_1.

5 Updates Visualization

One of the purposes of BGPlay is to visualize BGP updates. However, the semantics of an update is essentially in the changes it induces in the routing. Hence, our approach relies on two types of visualization techniques. First, we exploit methods to visualize the status of the routing at a given instant of time. Second, we exploit techniques that allow the user to perceive how an update brings the routing from an initial status to a consequent one.

5.1 The Routing Graph

If we focus the attention on a prefix, the status of the routing at a given time for that prefix consists of a set of AS-paths, each representing the best route at that time to reach the prefix from a certain AS. Such a status is effectively represented with a *routing graph*. A routing graph is a decorated graph in which each vertex is an AS and edges are the pairs of ASes that appear consecutively in at least one of the paths. Each edge is labeled with the set of paths traversing it.

The AS-paths representing the status of the routing are provided to the Presentation Engine by the Query Manager. In general, part of them come from ORV and represent the routes selected by the ORV peerers to reach the prefix. Part of them come from RIS and represent the routes selected by the RRC peerers to reach the prefix. The overall information conveyed by the representation is the routing of the Internet traffic flowing toward the specified prefix at the AS level.

Visualizing a routing graph is not an easy task. In fact, even if large portions of it look like a tree, it may contain cycles and dense subgraphs. Further, some vertices may have many incident edges and a single edge may be traversed by several paths.

In designing the Presentation Engine we have identified the following requirements:

- The attention of the user should mainly be focused on the AS originating the prefix (target AS).
- An AS should appear in the drawing at a geometric distance from the target AS that is roughly proportional to the number of (AS-)hops separating them.
- Each single AS-path must be fully identified, even if traversing edges that are traversed by other paths. Ob-

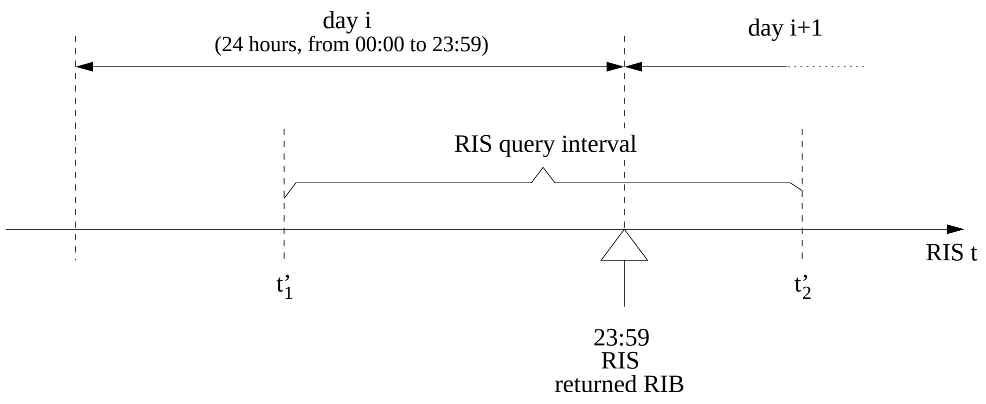


Figure 4. RIBs returned by the RIS.

serve that this requirement would be easy to meet if the graph was a tree; in fact in this case there is just one tree path from each AS to the target AS. The presence of cycles makes the problem more complex.

In order to compute drawings of the routing graphs satisfying the above requirements we used a "spring embedder." It consider the graph as a system of bodies (vertices) and forces acting between the bodies. Such forces can either attract or repel the bodies. The system is left free to oscillate until an equilibrium is reached. We made the choice to use the following types of forces:

- A repelling force is set between each pair of ASes.
- An attractive force is set between each pair of ASes connected by an edge.
- To damp the oscillations, the effect of the forces is decreased with the time. Essentially, this is obtained by progressively augmenting the "viscosity" in the system.

We also fixed the position of the target AS at the center of the area. Paths are represented with different colors and edges traversed by several paths are displayed using as many lines as the number of paths traversing that edge, where each line is colored with the color of the corresponding path.

Fig. 5 shows a drawing produced by BGPlay. It is about a prefix announced by the GARR AS (AS137). It clearly shows that the route collectors are reached by the announcements of AS137 through three main directions. Namely, part of the paths flow through AS3549 (Globalcrossing), other paths through AS1299 (TeliaNet), while others through AS20965 (GEANT). Fig. 6 shows the routing for AS7018.

5.2 Sequences of Updates

Obviously, the visualization of the routing graph before and after a BGP update occurred is sufficient to convey the information of the update. However, it is essential for the two consecutive visualizations to be "similar", while the routing change should be apparent to the user.

Consider a BGP announcement and the corresponding AS-path p'. Let AS_T be the target AS and suppose that the ending ASes of p' are AS_T and AS_S. Two cases are possible: either another AS-path p with the same pair of ending ASes was already in the routing graph or not. In the first case the user should perceive that the traffic flow from AS_S to AS_T is changing its route. In the second case it is important to make clear that a new source of traffic is becoming visible from AS_T.

A route change from p to p' involves several possible changes in the routing graph. Let us compare $p = (AS_T = AS_1, AS_2, \dots, AS_m = AS_S)$ and $p' = (AS_T = AS'_1, AS'_2, \dots, AS'_n = AS_S)$. We can split p and p' into sub-paths as follows. Let AS_i be the first AS of p that is equal to some AS (say AS'_j) of p'. We split p and p' into the sub-paths $(AS_1, \dots, AS_i)$, $(AS_i, \dots, AS_m)$ and $(AS'_1, \dots, AS'_j)$, $(AS'_j, \dots, AS'_n)$, respectively. We can repeat the above split process on the two sub-paths $(AS_i, \dots, AS_m)$ and $(AS'_j, \dots, AS'_n)$, until they are no longer decomposable. Such a process yields a decomposition of the two original paths into an equal number of sub-paths pairwise starting and ending on the same vertices.

The Presentation Engine performs a graphic "morphing" where each sub-path in which p is decomposed is mapped to the corresponding sub-path of p'. Three cases are possible:

1. the two sub-paths have equal length
2. the sub-path of p is longer than the sub-path of p', and
3. the sub-path of p is shorter than the sub-path of p'.

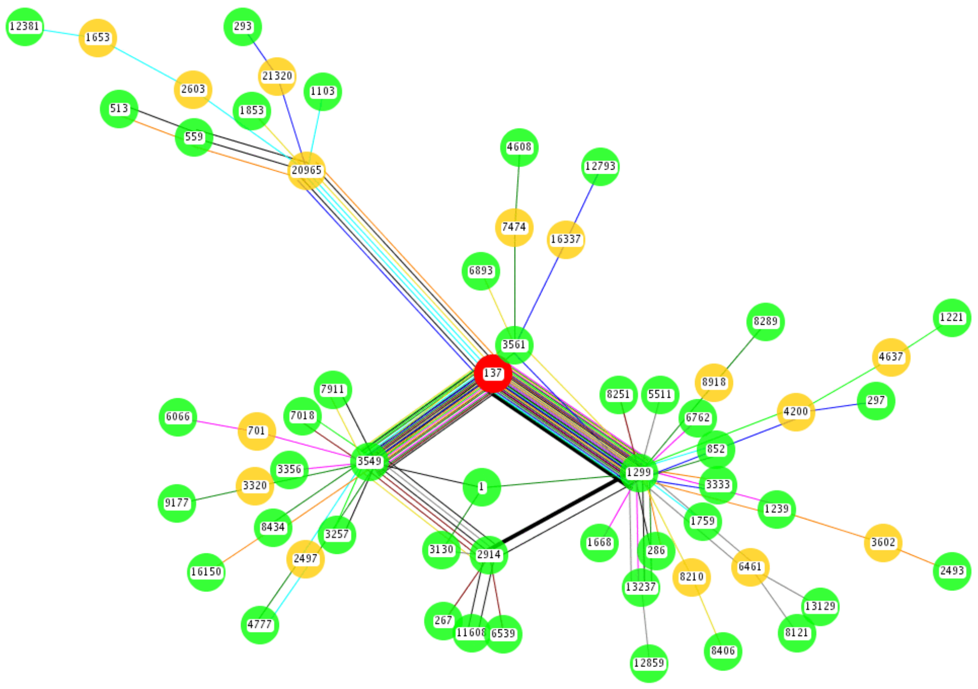


Figure 5. A drawing produced by BGPlay rapresenting the routing for prefix 193.204.0.0/15 (AS137).

In all the above cases we introduce in the routing graph a chain of additional “dummy” vertices and move them from the original sub-path to the position of the new sub-path. However, in case 2 some dummy vertices are “absorbed” into the same vertex, while in case 3 some dummy vertices are “created” from the same vertex. Of course, some ASes of the original graph may disappear since they are no longer traversed by any path and some ASes are created because they are in p' and were not in the graph.

Consider now a BGP withdrawal. It is managed by the Presentation Engine as follows. First, the involved path is highlighted, to attract the attention of the user, then the edges traversed by the path change their labels an, in case, removed.

6 Conclusions and Future Work

We have presented BGPlay, a system designed to visualize the evolution of the routing involving a given prefix at the Autonomous Systems level (BGP level). BGPlay is able to show the BGP events occurred in a given time interval through an animation illustrating the corresponding changes in the routing graph. BGPlay is available at `http://www.dia.uniroma3.it/~compunet`.

Future work will mainly focus on the following issues.

- We plan to test the effectiveness of new visualization methods, alternative to those based on the display of a routing graph.
- We would like to provide the user with more large-scale visualization facilities, that allow to track the evolution of the routing paths of several prefixes (not necessarily originated by the same AS) at the same time.
- We are interested in integrating new data sources, both publicly available and provided by private organizations.

Acknowledgements

We would like to thank Fabrizio Lombardozzi for implementing a preliminary version of the system, called Flapviewer.

References

[1] Hermes.
`http://www.dia.uniroma3.it/~hermes/`.

[2] Multithreaded Routing Toolkit (MRT) Project [Online].
`http://www.mrtd.net`.

[3] Routing information service of the ripe.
`http://www.ripe.net/ripencc/pub-services/np/ris/`.

[4] University of Oregon RouteViews project.
`http://www.routeviews.org`.

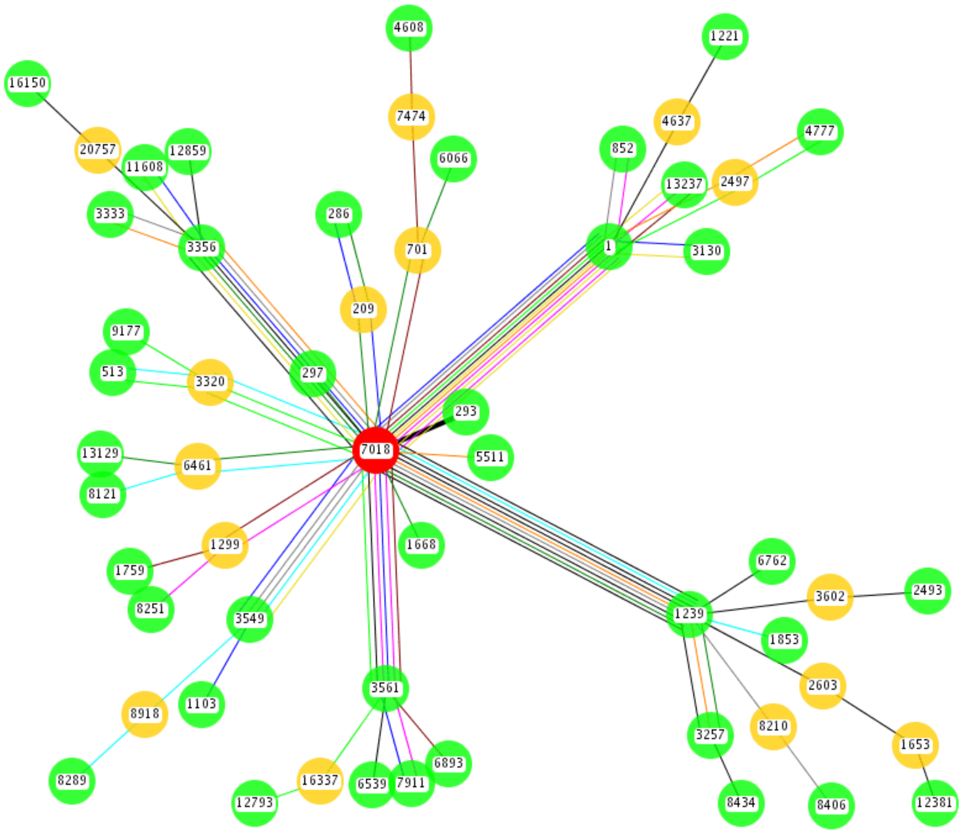


Figure 6. Flows of traffic for AS7018 represented by BGPlay.

[5] S. Branigan, H. Burch, B. Cheswick, and F. Wojcik. What can you do with traceroute? *IEEE Internet Computing*, Sept./Oct. 2001. `http://computer.org/internet/v5n5/index.htm`.

[6] A. Carmignani, G. Di Battista, W. Didimo, F. Matera, and M. Pizzonia. Visualization of the high level structure of the internet with hermes. *J. of Graph Algorithms and Applications*, 6(3):281–311, 2002.

[7] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999.

[8] L. Gao and J. Rexford. Stable internet routing without global coordination. In *Measurement and Modeling of Computer Systems*, pages 307–317, 2000.

[9] R. Govindan and H. Tangmunarunkit. Heuristics for internet map discovery. In *IEEE INFOCOM 2000*, pages 1371–1380, Tel Aviv, Israel, March 2000.

[10] T. Griffin and G. T. Wilfong. An analysis of BGP convergence properties. In *SIGCOMM*, pages 277–288, 1999.

[11] B. Huffaker, D. Plummer, D. Moore, and k claffy. Topology discovery by active probing. Technical report, Cooperative Association for Internet Data Analysis - CAIDA, San Diego Supercomputer Center, University of California, San Diego, 2002.

[12] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. In *SIGCOMM*, pages 175–187, 2000.

[13] C. Labovitz, G. R. Malan, and F. Jahanian. Internet routing instability. *IEEE/ACM Transactions on Networking*, 6(5):515–528, 1998.

[14] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *J. Visual Lang. Comput.*, 6(2):183–210, 1995.

[15] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. In *Proceedings of ACM/SIGCOMM '02*, Aug. 2002.

[16] J. W. Stewart. *BGP4: Inter-Domain Routing in the Internet*. Addison-Wesley, Reading, MA, 1999.