# Computing the Types of the Relationships between Autonomous Systems

Giuseppe Di Battista, Thomas Erlebach, Alexander Hall, Maurizio Patrignani, Maurizio Pizzonia, and Thomas Schank

***Abstract*— We investigate the problem of computing the types of the relationships between Internet Autonomous Systems. We refer to the model introduced by Gao (IEEE/ACM Transactions on Networking, 9(6):733–645, 2001) and Subramanian et al. (IEEE Infocom, 2002) 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 approximation algorithms and heuristics based on a novel paradigm and show their effectiveness against publicly available data sets. The experiments provide evidence that our algorithms perform significantly better than state-of-the-art heuristics.**

## I. Introduction

An *Autonomous System* (AS) is a portion of the 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) [1].

Several authors (see, e.g., [2], [3]) 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 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.

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 or at the router level (see, for example, [4], [5], [6]).

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 [4] 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 [7] 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 [7] a heuristic is presented together with experimental results. An analysis on the properties of the labeled graphs obtained with such heuristics is provided in [8].

Subramanian et al. [9] formally define, as an optimization problem, a slightly simplified version of the problem addressed in [7] 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.

A different approach towards obtaining the AS relationships has recently been pursued by Siganos and Faloutsos [10]; they use data available from Internet Routing Registries about the import and export policies of individual ASes in order to determine the relationships between them.

This paper contributes to the line of research opened in [7], [9]. Our main results are the following.

- We solve a problem explicitly stated in [9]. Namely, we characterize the complexity of determining the relationships between ASes while maximizing the number of valid paths, i.e., the number of paths consistent with the

Preliminary extended abstracts describing different parts of this work were published by Di Battista, Patrignani, and Pizzonia in the Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communication Societies (INFOCOM 2003) and by Erlebach, Hall and Schank in the Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN 2002).

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, by "The Multichannel Adaptive Information Systems (MAIS) Project", MIUR Fondo per gli Investimenti della Ricerca di Base, by the Swiss National Science Foundation Project "Approximation algorithms for problems in communication networks," and by European Commission - Fet Open project DELIS IST-001907 Dynamically Evolving Large Scale Information Systems.

G. Di Battista, M. Patrignani and M. Pizzonia are with Dipartimento di Informatica e Automazione, Università di Roma Tre, Rome, Italy (e-mail: {gdb,patrigna,pizzonia}@dia.uniroma3.it).

T. Erlebach is with the Department of Computer Science at the University of Leicester, U.K. (e-mail: t.erlebach@mcs.le.ac.uk).

A. Hall is with Institute for Theoretical Computer Science, ETH Zürich, Switzerland (e-mail: alex.hall@gmail.com). He was supported by the joint Berlin/Zurich graduate program Combinatorics, Geometry, and Computation (CGC), financed by ETH Zurich and the German Science Foundation (DFG), and in DICS-Project No. 1838 by the Hasler Foundation.

T. Schank is with the Faculty of Informatics, University Karlsruhe, Germany (e-mail: schank@ira.uka.de).

edge labeling. In particular:

- We show that the problem is $NP$-hard in the general case and cannot even be approximated within a factor of $1/n^{1-\varepsilon}$ for any $\varepsilon > 0$ unless $NP = ZPP$,[1] 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 $\ell$, where $\ell$ is an arbitrary constant greater or equal to 2) — more specifically, we even show that there is a constant less than 1 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., consistent with the edge labeling); 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 in such a way that all paths are valid.

- 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, i.e., paths not consistent with the edge labeling). For some of the algorithms we can prove that their solution is guaranteed to be within a constant factor of the optimal solution if the given AS paths are short. Also, some of them are improvements over preliminary versions described in [12], [13].
- We experimentally show that the proposed approaches lead to algorithms that perform significantly better than the cutting edge heuristics of [9] with respect to the number of valid paths.

The paper is structured as follows. Section II describes the addressed problem. Section III shows an algorithm for testing if the problem admits a solution with no anomalous paths and how to find such a solution if it exists. In Section IV we prove the $NP$-completeness of the problem in the general case and present our inapproximability results. Section V presents our new algorithms. Their results are compared with the state of the art in Section VI. Section VII discusses the problem of discovering peer-to-peer relationships once customer-provider relationships are known. Finally, Section VIII 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., [2], [3], [14], [9]). 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.

[1] $ZPP$ is the class of problems that can be solved by a probabilistic Turing machine in expected polynomial time. According to [11], the faith in the hypothesis $NP \neq ZPP$ is almost as strong as in $NP \neq P$.

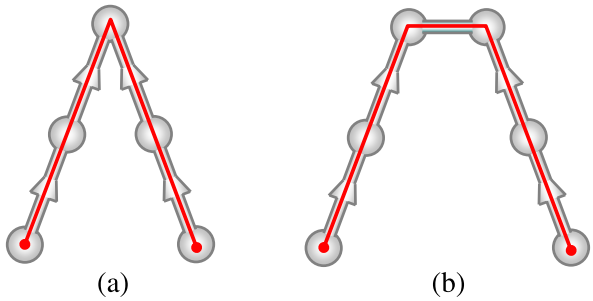


Fig. 1. An example of Type 1 (a) and of Type 2 (b) path.

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 [7], [9]. 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 pair 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 [7], that has been also re-casted in [9]. 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 $v$ (customer-provider 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 of the sequence of vertices $v_1, \ldots, v_n$. Then $p$ is *valid* if it is of one of two types:

Type 1: $p$ is composed of 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 $j = 1, \ldots, i-1$ the edge between $v_j$ and $v_{j+1}$ is directed from $v_j$ to $v_{j+1}$ and for $j = i, \ldots, n-1$ the edge between $v_j$ and $v_{j+1}$ is directed from $v_{j+1}$ to $v_j$. (See Figure 1.a.)

Type 2: $p$ is composed of a (possibly empty) sequence of forward edges, followed by an undirected edge, followed by a (possibly empty) sequence of backward edges. (See Figure 1.b.)

An *invalid path* is a path that is not valid. We refer to invalid paths also as *anomalous paths*. If two consecutive edges of a path violate the conditions of Type 1 and Type 2 above (i.e., if the edges are a backward edge followed by a forward edge, a

backward edge followed by an undirected edge, an undirected edge followed by a forward edge, or two undirected edges), we refer to these two edges as an *anomaly*. If we view an undirected edge as pointing in both directions, two consecutive edges on a path are an anomaly if and only if both edges point away from their common vertex. Observe that every invalid path must contain at least one anomaly.

At this point the above mentioned theorem [9] 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 on 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 [9] toward identifying the following problem.

> *Type-of-Relationship (ToR) Problem* [9]: 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 $P$.

An algorithm for the ToR problem is a *$\rho$-approximation algorithm* if it runs in polynomial time and always produces an orientation of some of the edges of the given graph such that the number of valid paths in $P$ is at least $\rho \cdot S$, where $S$ is the number of valid paths in the optimal solution. The value $\rho$ is always at most 1, and the goal is to find $\rho$-approximation algorithms for which $\rho$ is as close to 1 as possible.

Note that the objective of maximizing the number of valid paths is equivalent to the objective of minimizing the number of invalid paths in terms of optimal solutions. For approximation algorithms, the two objective functions are not equivalent. Following [9], we treat the ToR problem as a maximization problem with the number of valid paths as the objective.

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. Figure 3 shows an instance that admits an orientation without invalid paths with a possible orientation.

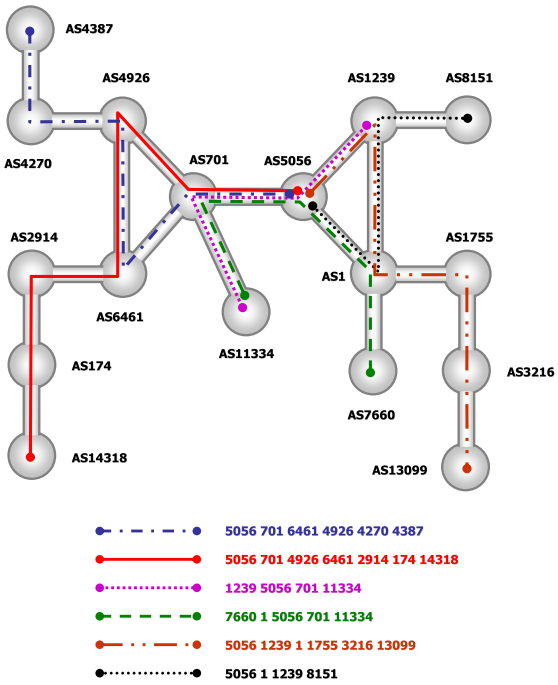


Fig. 2. An instance of the ToR problem that does not admit an orientation without invalid paths. The six paths of the instance are represented with different line styles.

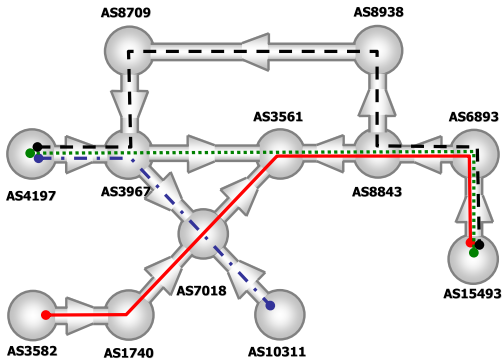


Fig. 3. An instance of the ToR problem that admits an orientation without invalid paths. The four paths of the instance are represented with different line styles.

## B. Simplifying the Problem

The Type-of-Relationship Problem is a maximization problem. In order to study its complexity, following a standard technique [15], we consider its corresponding decision version:

> *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 valid paths in $P$ is at least $k$.

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 losing 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 valid paths in $P$ is at least $k$.

Notice that ToR-D-simple 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 $(u, v)$. There are two cases: $p$ is valid or $p$ is invalid.

If $p$ is valid, 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 that 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.

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 $u$ undirected edges there exist $2^u$ 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 by looking for edges whose orientation can be removed without increasing the number of anomalous paths. A necessary and sufficient condition, which 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-step approach where in the first step a solution is found for ToR-D-simple and in the second step peering edges are discovered.

## III. Computing the Relationships between ASes 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 these problems have also been discussed. In this section we show that problem ToR-D-simple (and, consequently, ToR-D) can be solved efficiently when $k = |P|$, that is when we want to check if $G$ admits an orientation where all the paths in $P$ are valid (i.e., there are 0 invalid paths).

### A. Path Anomalies and Boolean Formulas

Observe that a path $p$ on $G$ composed of the sequence of vertices $v_1, \ldots, v_n$ is of Type 1 if and only if there 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 [15].

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 of 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 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.

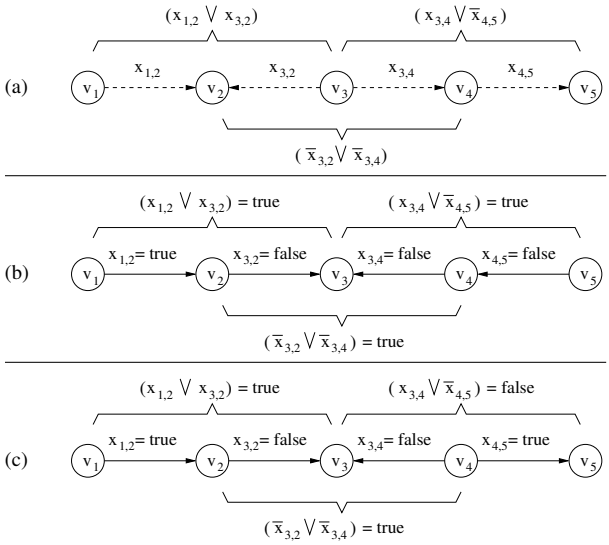


Fig. 4. (a) An initial orientation for a five vertices path and the boolean variables associated with its edges. The orientation shown in (b), which makes the path valid, corresponds to the truth assignment $x_{1,2} =$ true, $x_{3,2} =$ false, $x_{3,4} =$ false, and $x_{4,5} =$ false, which satisfies 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.

As an example consider a path composed of five vertices $v_1, \ldots, v_5$ and suppose that the initial orientation step has 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 4.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 $x_{1,2} = \mathsf{true}$, $x_{3,2} = \mathsf{false}$, $x_{3,4} = \mathsf{false}$, and $x_{4,5} = \mathsf{false}$. 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 4.b). On the other hand, consider the truth assignment $x_{1,2} = \mathsf{true}$, $x_{3,2} = \mathsf{false}$, $x_{3,4} = \mathsf{false}$, and $x_{4,5} = \mathsf{true}$. 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 4.c).

### B. Computational Aspects

Problem 2SAT may be efficiently solved by using the well known result in [16] that maps 2SAT into a problem on a suitable directed graph $G_{\mathsf{2SAT}}$. Observe that $G$ and $G_{\mathsf{2SAT}}$ are different graphs.

Although this result is clearly illustrated in the literature, we give a brief description to help the reader to better understand the algorithms described in Sections III-D and V.

Graph $G_{\mathsf{2SAT}}$ has two nodes for each boolean variable $x$ of 2SAT, corresponding to its two literals $x$ and $\overline{x}$. 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 $G_{\mathsf{2SAT}}$ 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 $G_{\mathsf{2SAT}}$ can be efficiently solved by computing the *strongly connected components* of $G_{\mathsf{2SAT}}$ 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 [17].

From a theoretical point of view, the outcome is that ToR-D-simple (and, as a consequence, ToR-D) with $k = |P|$, 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, the above algorithm can be implemented by exploiting a facility from the Leda [18] software library that efficiently computes the strongly connected components of a directed graph.

### 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 [9]) 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* [1] 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 by Subramanian et al. [9], who collect such test beds periodically and make them publicly available [19].

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 [9] and values computed from data available in [19] (and that are presented in Table I of this paper) appear to be slightly different.

TABLE I
TELNET LOOKING GLASS SERVERS AND CORRESPONDING AS GRAPHS.

| AS # AS Name | Apr 18, 2001 nodes / edges / paths | Apr 6, 2002 nodes / 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 / $2.5 \cdot 10^6$ | 13,055 / 27,277 / $4.6 \cdot 10^6$ |
| 3967 Exodus Comm. | 10,387 / 18,401 / 254,123 | 12,616 / 21,527 / 339,023 |
| 4197 Global Online J. | 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 |

Table II shows the results of the experiments. Observe that for all partial views but the one of the University of Oregon server [20], 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 the 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 5 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

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 |

of an instance of the ToR problem that does not admit an orientation without invalid paths.

| 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. 5. 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.

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.

### D. Finding an Orientation for ToR-D-simple

If a solution for ToR-D-simple with $k = |P|$ exists, that is, if $G$ admits an orientation where all the paths are valid, 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 [16] for computing such an assignment is the following. A function $f(v)$ can be computed for all the vertices of the graph $G_{\text{2SAT}}$ 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 $f(u) \leq f(v)$. Furthermore, two vertices $u$ and $v$ receive the same function value, $f(u) = f(v)$, if and only if they are in the same strongly connected component. 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* [18], 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 constrains some variables to have the same truth values in all 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 solution 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 that we 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.

## IV. THE DIFFICULTY OF MAXIMIZING VALID PATHS

The ToR problem was conjectured to be $NP$-complete in [9]. In Section III we have shown that finding a solution with zero invalid paths (provided that it exists) is a tractable problem. In this section we show that the ToR problem is indeed $NP$-complete in the general case, that is, when it does not admit an orientation without invalid paths. In addition, we derive inapproximability results showing that it is even hard to approximate the ToR problem within a small factor. First, we consider the case of instances where the given paths can have arbitrary length. Then, we show that the problem remains hard even if all given paths are short.

### A. Paths with Arbitrary Length

We give an approximation-preserving polynomial reduction from the $NP$-hard maximum independent set problem (denoted MAXIS) to the ToR problem. The goal of the MAXIS problem is to find, for a given undirected graph, a largest set of nodes such that no two nodes in the set are adjacent.

First, consider a graph $G$ with two paths as shown in Figure 6. Similarly to the example of Figure 2, it is easy to verify that there is no orientation of the edges such that both paths are valid. In particular, any orientation of edge $e_1$ yields a contradiction on the orientation of edges $e_2$ or $e_3$.

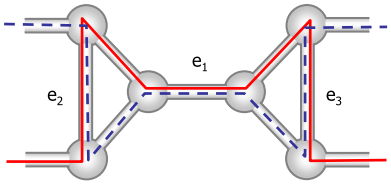


Fig. 6. A graph with two paths that cannot both be valid in any orientation.

*Lemma 1:* 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 the ToR problem. Let an instance of MAXIS be given by an undirected graph $H = (V_H, E_H)$. We create an instance $(G, P)$ of the ToR problem 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 6.

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 the ToR problem in polynomial time or approximate it in polynomial time with ratio $\rho$, we could also solve MAXIS in polynomial time or approximate it with ratio $\rho$, respectively. Since MAXIS is known to be $NP$-hard [15] and not even approximable with ratio $1/n^{1-\epsilon}$ on graphs with $n$ nodes for any $\varepsilon > 0$ unless $NP = ZPP$ [11], we obtain the following hardness result for the ToR problem.

*Theorem 1:* The ToR problem is $NP$-hard and cannot be approximated within a factor of $1/n^{1-\varepsilon}$ on instances with $n$ paths for any $\varepsilon > 0$ unless $NP = ZPP$.

The arguments leading to the theorem imply that the decision problems ToR-D and ToR-D-simple are $NP$-complete, as their membership in NP is easy to verify.

As it is not feasible to get a good approximation ratio for the general ToR problem, one might hope 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. One observation about the real data is that most of the AS paths are relatively short. Therefore, we investigate the complexity and approximability of the ToR problem if the length of the paths is bounded by a constant.

### B. Instances with Short Paths

In this section we show that even instances of the ToR problem that contain only paths of length 2 cannot be approximated better than some constant unless $P = NP$.

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

*Proof:* We reduce the MAX2SAT problem and apply an inapproximability result by Håstad [21]. An instance of the MAX2SAT problem is given in the same way as an instance of 2SAT, but the goal is to find an assignment to the boolean variables that maximizes the number of satisfied clauses. Håstad [21] has shown that MAX2SAT cannot be approximated within ratio $q = 0.955$ unless $P = NP$.

We start by reducing a MAX2SAT instance to a ToR 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 $x_i$, $i \in \{1 \dots n'\}$ and clauses $c_j$, $j \in \{1 \dots m'\}$. 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 the ToR problem this edge is directed towards $x_i$ this corresponds to $x_i = \mathsf{true}$. Otherwise, if it is directed towards $\overline{x_i}$, this means $x_i = \mathsf{false}$. 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 7 shows a simple example with one clause, and the graph of Figure 8.a shows an example with two clauses.

The paths and edges are defined such that a path is valid in a solution to the instance of the ToR problem if and only if the corresponding clause is satisfied. This is clear because a path is valid if and only if $e_i$ 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 in such a way that the whole path is valid. Conversely, for an unsatisfied clause no such direction of $\{l_i, l_j\}$ exists.

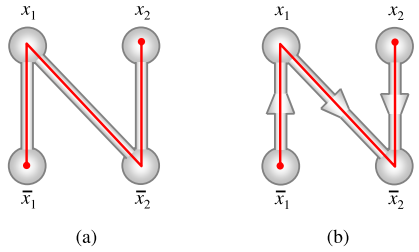


Fig. 7. Example of how an instance of the ToR problem is constructed from a MAX2SAT instance. The MAX2SAT instance composed of the single clause $(x_1 \vee \overline{x_2})$ corresponds to the ToR instance composed of the single path $\overline{x_1}, x_1, \overline{x_2}, x_2$ represented in (a). The solution of the ToR instance represented in (b) corresponds to the solution of the MAX2SAT instance $\{x_1 = \mathsf{true}, x_2 = \mathsf{false}\}$.

Thus, maximizing the number of valid paths also maximizes the number of satisfied clauses. With [21] we get that the ToR problem is not approximable within ratio $q = 0.955$ for instances with paths of length at most $k$ for 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 Figures 8.a and 8.b for an example of such a replacement. Clearly, in any ToR 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 instance of the ToR problem with $S$ valid paths gives an optimal assignment to the variables such that $S - m'$ clauses are satisfied, where $m'$ is the number of clauses of the MAX2SAT instance. An approximate solution to the ToR problem giving $A_2$ valid paths leads to $A_2 - m'$ satisfied clauses. With [21] we know that $\frac{A_2 - m'}{S - m'} \leq q$ for at least one instance of MAX2SAT. With $S - m' \geq 3/4 \cdot m'$ (at least $3/4$ of the clauses of any MAX2SAT instance can be satisfied[2]) this yields $A_2 \leq \frac{4+3q}{7} \cdot S$, which concludes the proof. ∎

## V. Algorithms for Computing the AS Relationships

In Section IV we have seen that the ToR problem is computationally hard (even approximation-wise), and in Section III we have seen that, even if portions of the 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 efficient methods for discovering the AS relationships in a big chunk of the Internet with a large number of valid paths. First, we present approximation

[2] A random truth assignment satisfies each clause of a 2SAT formula with probability $3/4$, implying that the expected number of satisfied clauses in a random assignment is $3/4$ of the total number of clauses, see e.g. [22, pp. 104–105].

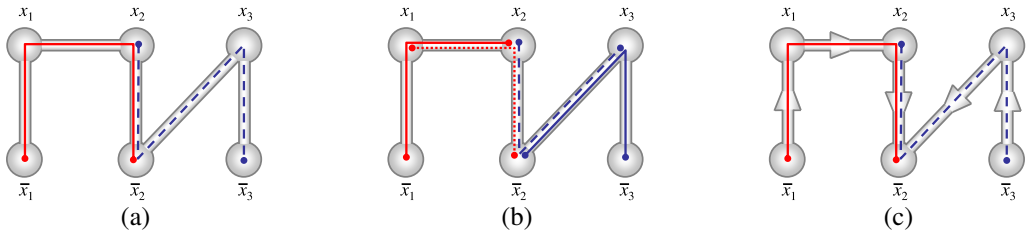


Fig. 8. Network and paths resulting from the two clauses $x_1 \vee x_2$, $\overline{x_2} \vee x_3$: Constructed instance with length 3 paths (a), modified instance with length 2 paths (b), and possible solution where both clauses are satisfied (c).

algorithms that are based on known algorithms for MAX2SAT. We can show that these algorithms achieve good approximation ratios for instances with short paths. After that, we will present a heuristic approach that also exploits the relationship to the MAX2SAT problem.

### A. Approximation Algorithms for Short Paths

In the following, we first present a very simple randomized approach achieving constant approximation ratio for the case of paths with bounded length. 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 $\ell$ for $\ell = 2, 3, 4$.

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

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

where $A_{rand}$ is the value of the approximate solution and $S$ 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 ToR problem with paths no longer than $\ell$ edges can be approximated within a factor of $\frac{\ell+1}{2^\ell}$ 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 1:* The ToR problem with average path length bounded by $\ell$ can be approximated within a factor of $\frac{2\ell+1}{2 \cdot 4^\ell}$ of the optimum in polynomial time.

*Proof:* Consider an instance of the ToR problem with $n$ paths. If the average path length is $\ell$, there are at least $n/2$ paths with length at most $2\ell$. Applying the simple randomized algorithm described above to these $n/2$ paths, we obtain a solution with at least $\frac{n}{2} \cdot \frac{2\ell+1}{4^\ell}$ valid paths. ∎

To obtain better ratios for paths of length at most 4 or less, we employ an approximation algorithm for MAX2SAT as a subroutine. The first approximation algorithm for MAX2SAT based on semidefinite programming (SDP) was presented by Goemans and Williamson in their seminal paper [23]. Their algorithm has approximation ratio $0.878$. Feige and Goemans improved the algorithm and obtained approximation ratio $0.931$. The currently best known approximation algorithm for MAX2SAT is due to Lewin, Livnat and Zwick [24]. It achieves approximation ratio $r = 0.940$.

Given an instance $(G, P)$ of the ToR problem, we construct an instance of MAX2SAT from the paths as described in Section III (by adding a clause for every pair of consecutive edges on each given 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. The resulting approximation guarantee for this algorithm on paths of length at most 4 or less as well as the ratio obtained by the simple randomized algorithm of Theorem 3 for instances with longer paths are stated in the following theorem.

*Theorem 4:* The ToR problem with paths no longer than $\ell$ edges can be approximated within a factor $c_\ell$ of the optimum in polynomial time, where $c_\ell$ has the following form: $c_2 := 0.940$, $c_3 := 0.839$, $c_4 := 0.358$, and $c_\ell := \frac{\ell+1}{2^\ell}$ for $\ell > 4$.

*Proof:* For the case of $\ell > 4$, the result follows from Theorem 3. Now consider an instance of the ToR problem with paths of length at most $\ell$ for some $\ell \leq 4$. As described above, our algorithm constructs a MAX2SAT instance as described in Section III, applies the $0.940$-approximation algorithm for MAX2SAT from [24] to it, and orients the edges according to the resulting truth assignment.

To analyze the algorithm, let us first introduce a bit of notation: Let $S$ denote the optimum value of the considered instance of the ToR problem and $A_\ell$ the value of our approximate solution for an instance with path length bounded by $\ell$. For $1 \leq i \leq \ell$, let $g_i$ be the number of paths in $P$ that have length exactly $i$. 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 $A_\ell/S$, it suffices to consider only instances in which all paths have length exactly $\ell$: 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 $\ell \cdot n$ 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 the ToR problem, because each path is represented by exactly one clause that is satisfied if and only if the path is valid.

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. 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 an instance of the ToR problem with $n = g_3$ paths of length 3. Note that an optimal solution of MAX2SAT has the value $S + g_3$ and directly gives an optimal solution to the ToR problem (with value $S$). The MAX2SAT approximation algorithm satisfies $A_3 + g_3$ clauses with

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

Because there is always an easily computable solution to MAX2SAT such that at least $3/4$ of the clauses are satisfied and there are $2 \cdot g_3$ clauses, we can assume $A_3 + g_3 \geq 3/2 \cdot g_3$ or $g_3 \leq 2 \cdot A_3$. Applying this to (1) leads to $A_3 \geq r \cdot (S + g_3) - g_3 \geq r \cdot S + 2(r-1) \cdot A_3$ and thus $A_3 \geq r \cdot S/(3-2r)$, 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 $T = A_4 + y + g_4$ computed by the MAX2SAT approximation algorithm with a solution $T^* = S + y' + g_4$ derived from an optimal solution of the corresponding instance of the ToR problem. By [25] we know that $T/T^* \geq r$. From this we can bound the approximation ratio $A_4/S$. In the worst case $y = g_4$ and $y' = S$. This gives

$$\begin{aligned} \frac{A_4 + 2 \cdot g_4}{2 \cdot S + g_4} &\geq r \\ A_4 \geq 2r \cdot S + (r-2) \cdot g_4 &\geq 2r \cdot S + 4(r-2)A_4, \end{aligned}$$

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 $A_4 \geq \frac{2r}{9-4r} \cdot S$. 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 analysis 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. ■

### B. A Heuristic Approach

In this section, we propose a heuristic algorithm based on the idea of computing a maximal set $P' \subseteq P$ of paths (subset of the given set $P$ of paths) such that ToR-D with $k = |P'|$ admits a solution that makes all paths in $P'$ valid. 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 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 discussion, we propose 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 as follows. Initialize $P'$ with the set of all paths in $P$.

1) Construct the $G_{\mathsf{2SAT}}$ 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 $G_{\mathsf{2SAT}}$ (e.g., with the algorithm in [17]).
4) Identify each variable $x$ such that $x$ and $\overline{x}$ are in the same strongly connected component of $G_{\mathsf{2SAT}}$.
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 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 $G_{\mathsf{2SAT}}$.

At the end of the first phase, we have a large set $P' \subseteq P$ 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 $\pi$ in $P \setminus P'$ and check whether there is an orientation that makes all paths in $P' \cup \{\pi\}$ valid. If this is the case, we add $\pi$ to $P'$, otherwise we discard $\pi$ 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 algorithm for ToR-D-simple with $k = |P'|$ of Section III. In this way, all edges used by at least one path in $P'$ are oriented. The remaining edges are not assigned an orientation; in our experiments (Section VI), we treat them as undirected (peer-to-peer) edges.

Alternatively, the second phase can be replaced by the following heuristic. Let $P'$ be the set of paths obtained at the end of the first phase. Now the idea is to let every path in $P \setminus P'$ that contains a so far unoriented edge "vote" in which way this edge should be directed. A path $\pi \in P \setminus P'$ votes for one of the two possible directions of an unoriented edge if $\pi$ can only be valid in case the edge is directed correspondingly. To give a simple example, imagine a path consisting of three edges: a forward edge, a backward edge, and finally a so far unoriented edge. This path can only be valid if the last edge is directed to be a backward edge. Thus the path would vote for this direction here. Another example would be a path that consists of a forward edge, an unoriented edge, and a backward edge. This path would vote for neither of the directions of its middle edge, since the path is valid no matter in which way the edge is directed. After all paths have voted, for each previously unoriented edge the direction is chosen according to the majority of the votes that the edge has received (ties are broken arbitrarily).

The hope is that this approach should work quite well if most edges are already oriented after the first phase and only few edges are left to be oriented in the second phase.

## VI. Experimental Results

In this section, we describe experimental results obtained with our new algorithms and compare them with previous approaches. First, in Section VI-A, we discuss the implementations of our own algorithms and of previous approaches that we have used in our experiments. Then, in Section VI-B, we describe the data sets on which we have tested the algorithms. In Section VI-C, we report the results of comparing the different algorithms with respect to the number of valid paths that they achieve on these data sets. In Section VI-D, we study the relationship between anomalies and anomalous paths in more detail, i.e., we investigate for the edge classifications obtained with the different algorithms whether there are a few anomalies that are responsible for most of the anomalous paths. Finally, in Section VI-E, we employ the methodology proposed by Subramanian et al. [9] to validate their approach to formalizing the ToR problem: We check whether the classifications computed by our algorithms also make many paths valid in additional sets of AS paths from different sources.

### A. Evaluated Algorithms

First, we have implemented an approximation algorithm for instances with short paths along the lines discussed in Section V-A. The algorithm constructs a MAX2SAT instance from the given paths and uses an SDP-based algorithm to obtain an approximate solution. In our implementation, the paths are first preprocessed by directing edges that can be directed without conflicts. This shortens the paths considerably (see Section VI-B). Then it is checked with the ToR-D-simple algorithm for $k = |P|$ described in Section III whether the graph can be oriented such that all paths are valid. If this is not the case, an approximate solution is calculated as follows: MAX2SAT is relaxed to a semidefinite program following [23]. The rounding is done as in [25], which improves the approximation ratio of [23] by adding new constraints and modifying the rounding strategy. We did not add the extra constraints because the instances would have become too large, but adopted the new rounding strategy.[3] The freely available solver DSDP 4.5 [26] was used to solve the semidefinite programs. We refer to the resulting implementation as algorithm EHS. For instances consisting of several million paths in a graph with 15,000 nodes, it produces edge classifications within a few minutes on a modern workstation.

Furthermore, we have implemented the heuristic approach described in Section V-B. Both alternatives of implementing the second phase were considered. The algorithm that executes the second phase by checking for each path $\pi$ in $P \setminus P'$ whether the set $P' \cup \{\pi\}$ can be valid is referred to as DPP. In the setting with multiple paths (see Section VI-B), we process the paths in $P \setminus P'$ in order of non-increasing multiplicity. The alternative algorithm that executes the second phase by letting the paths in $P' \setminus P$ vote for a direction of the unoriented edges is referred to as DPP*. On a modern workstation, DPP needs a calculation time of several hours for one data set (most of the time is spent in the second phase), while a run of DPP* usually takes only a few minutes.

In order to compare our algorithms with Subramanian et al.'s algorithm from [9] we downloaded the edge classifications that they have obtained with their algorithm from the webpage [19]. We refer to their algorithm as SARK.

For Gao's algorithm [7], which we refer to as GAO, we used the implementation available from [27].

### B. Description of Data Sets

TABLE III
CHARACTERISTICS OF AS DATA SETS AT FIVE DIFFERENT DATES.

| Date | nodes / edges | paths / unique p. | $m_1$ / $m_2$ | $a_1$ / $a_2$ |
|---|---|---|---|---|
| 2001/04/18 | 10,916 / 23,761 | 3,423,422 / 502,515 | 12 / 10 | 3.5 / 1.7 |
| 2002/02/04 | 12,766 / 27,759 | 4,988,100 / 768,688 | 11 / 9 | 3.5 / 1.3 |
| 2002/04/06 | 13,124 / 28,326 | 6,356,435 / 982,320 | 11 / 8 | 3.5 / 1.4 |
| 2003/01/09 | 14,674 / 30,800 | 5,993,411 / 906,285 | 11 / 8 | 3.6 / 1.3 |
| 2004/02/10 | 16,911 / 37,369 | 7,525,967 / 1,151,245 | 12 / 9 | 3.7 / 1.5 |

For our experiments we used the data that has been accumulated by Subramanian et al. [9] and made available on the WWW [19]. For five different dates (18 April 2001, 4 February 2002, 6 April 2002, 9 January 2003, and 10 February 2004), routing paths from 10, 9, 14, 10 and 10 autonomous systems, respectively, were collected. This data yields ToR instances with several million paths (between 3.4 and 7.5 million) in graphs of about 11,000–17,000 nodes and 23,000–37,000 edges. Note that the data set of 18 April 2001, which consists of the union of all the paths of the Telnet Looking Glasses of Table I, was used by Subramanian et al. [9] as the

[3] We did not implement the best known MAX2SAT approximation algorithm from [24], because we already obtained very good results with our implementation of the algorithm based on [23], [25].

basis for evaluating their classification algorithm. Therefore, it is important to compare our new algorithms from Section V with previous algorithms on this reference data set. In addition, we use the four other, more recent data sets in order to obtain richer experimental results.

In order to get the data sets, we downloaded the processed path files from the website [19]. In a first cleaning step, we replaced consecutive identical ASes on each path by a single occurrence of that AS and removed AS paths consisting of a single AS only. Some characteristics of the five resulting data sets are shown in Table III. Note that many paths occur more than once in the path files. In the table, we report the number of paths counting multiple occurrences as well as the number of unique paths. Whenever we deal with a data set, we use the expressions "multiple paths" and "unique paths" to indicate whether we use the original path files (with multiple occurrences of many paths) or a reduced data set with a single occurrence of every distinct path, respectively. We ran the algorithms EHS, DPP, DPP*, and GAO on both unique paths and multiple paths[4] as input, thus producing one edge classification for the case of unique paths and a possibly different edge classification for the case of multiple paths with each algorithm. The edge classifications produced by the SARK algorithm were downloaded from [19] and thus we use the same SARK classification for the case of unique paths and of multiple paths.

From each set of path files (for one date), we then created an AS graph by taking all nodes and edges that occur in the path files. The number of nodes and edges of the resulting graphs are also given in the table.

It is notable that the path lengths in these real data sets are relatively short. The last two columns of Table III give the maximal and average path lengths before ($m_1$ and $a_1$) and after ($m_2$ and $a_2$) the preprocessing done by algorithm EHS as mentioned in Section VI-A. (The numbers refer to the data with multiple paths.) This shows that the assumption in Section V-A that the path lengths are bounded by a small constant is quite realistic.

To illustrate how DPP works in practice, we describe in more detail how the computation of DPP progresses for the data set of 18 April 2001 (multiple paths). For this data set with 3,423,422 paths, the starting $G_{\mathsf{2SAT}}$ 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 phase contains 3,236,823 paths. During the second phase, 171,756 paths have been re-inserted without causing anomalies. The final maximal set of valid paths contains 3,408,579 paths.

After computing a maximal set of paths, an orientation for the edges of the AS graph obtained from those paths was computed using the technique illustrated in Section III-D. A fragment of the computed orientation has in fact been used for the example of Figure 3.

[4]In the case of multiple paths, we ran algorithm GAO on the original data, i.e. we did not perform the cleaning step that removes prepending and empty paths. This yielded slightly better results.

Using the condition discussed in Section II-B we have also checked for edges that can be made undirected while preserving the quality of the solution and found 3,270 such edges that can be considered as candidates for being peer-to-peer edges. We found that about 40% of the edges in the AS graph that connect different top-level providers (the 20 ASes classified as 'inner core' according to the AS hierarchy for 18 April 2001 from [19]) are among these peer-to-peer candidates.

On the same platform as the one described in Section III, this run of DPP, involving 3,423,422 paths, required a computation time of about 10 hours.

### C. Comparison of Edge Classification Algorithms

We used the algorithm EHS, DPP, DPP*, GAO and SARK to infer the AS relationships for the five data sets described in the previous section (in the case of SARK, we downloaded the resulting classification from [19]). First, we checked with our ToR-D-simple algorithm for $k = |P|$ (cf. Section III) whether the graph can be oriented in such a way that all paths are valid. This turned out not to be the case for all five data sets.

For the relationship classifications obtained with each of the five different algorithms, we computed the number (and percentage) of valid and invalid paths. The results are shown in Table IV. For each algorithm and each date, we report the percentage of invalid paths. The results to the left refer to the data with multiple paths, i.e. duplicates were not removed and the reported numbers count the invalid paths including multiple occurrences. The right side of the table refers to the data with unique paths, i.e., we removed duplicate paths in the input before we ran the algorithms and the reported numbers refer to the number of unique paths that are invalid. Each of the tables has two columns for algorithm GAO: in column "sib", the edges classified by the algorithm as sibling edges are indeed interpreted as siblings (i.e., they can occur anywhere on a path without making the path invalid); in column "peer", these edges are treated like undirected (peer-to-peer) edges. The edge classifications produced by the SARK algorithm, which we downloaded from [19], leave the classification of some edges open. We treated these unlabeled edges as undirected edges (peer-to-peer edges) for our evaluation.

On almost all of the instances, the orientations computed by DPP, DPP* and EHS make less than 0.5%, 0.3%, and 0.3%, respectively, of the given paths invalid. This should be contrasted with the edge classifications computed by SARK and GAO (peer). In these edge classifications, about 15–30% of the paths are invalid. This demonstrates clearly that our new heuristics DPP and DPP* as well as our SDP-based approximation algorithm EHS outperform previous approaches for the ToR problem if edge classifications without sibling edges are desired. DPP* and EHS produce the best results and even fare well in comparison to GAO (sib), which classifies roughly 1.5% of the edges as sibling edges; DPP*, EHS and GAO (sib) achieve comparable numbers of invalid paths, although DPP* and EHS do not use any sibling edges (note that sibling edges simplify the task of making paths valid).

The improvement shown by DPP and EHS with respect to

TABLE IV

PERCENTAGE OF INVALID PATHS FOR THE DIFFERENT ALGORITHMS (LEFT: MULTIPLE OCCURRENCES OF PATHS COUNTED, RIGHT: UNIQUE PATHS).

| Date | DPP | DPP* | EHS | SARK | GAO (sib) | GAO (peer) |
|---|---|---|---|---|---|---|
| 2001/04/18 | 0.43% | 0.27% | 0.18% | 27.87% | 0.26% | 29.90% |
| 2002/02/04 | 0.29% | 0.13% | 0.14% | 29.54% | 0.29% | 27.64% |
| 2002/04/06 | 0.37% | 0.16% | 0.12% | 28.41% | 0.37% | 25.57% |
| 2003/01/09 | 0.26% | 0.13% | 0.12% | 30.68% | 0.14% | 16.05% |
| 2004/02/10 | 0.19% | 0.12% | 0.10% | 22.93% | 0.05% | 14.98% |

| DPP | DPP* | EHS | SARK | GAO (sib) | GAO (peer) |
|---|---|---|---|---|---|
| 0.57% | 0.37% | 0.30% | 28.80% | 0.43% | 28.19% |
| 0.36% | 0.24% | 0.22% | 32.85% | 0.40% | 26.92% |
| 0.45% | 0.21% | 0.20% | 31.33% | 0.38% | 24.38% |
| 0.36% | 0.26% | 0.21% | 33.83% | 0.09% | 17.62% |
| 0.28% | 0.22% | 0.20% | 25.04% | 0.07% | 17.05% |

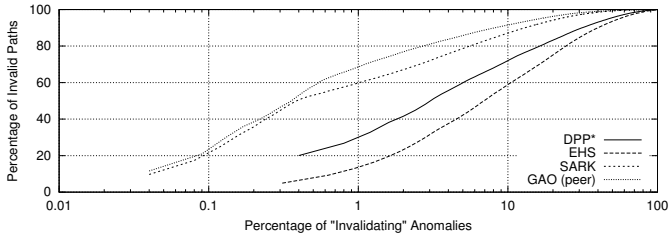


Fig. 9. Plot for 18 April 2001 showing how many anomalies are responsible for invalidating the majority of the invalid paths, considering the individual solutions of the algorithms DPP*, EHS, SARK and GAO (peer).

their preliminary implementations described in [12], [13] are due to code debugging and re-engineering.

It is interesting to know how many edges of the AS graph are classified in the same way by different classification algorithms. We have checked this for any pair of algorithms among DPP*, EHS, SARK and GAO (peer) for 18 April 2001. The fractions of identically classified edges are 95.34% for DPP* and EHS, 89.61% for DPP* and SARK, 89.68% for DPP* and GAO(peer), 91.15% for EHS and SARK, 90.63% for EHS and GAO(peer), and 89.99% for SARK and GAO(peer).

### D. Relating Anomalies to Anomalous Paths

Recall that an anomaly is a situation where two consecutive edges on a path point away from an internal node of that path. Thus each anomaly in a solution invalidates at least one of the given paths. An interesting question is how the invalid paths are distributed over the anomalies. In other words: How many anomalies are responsible for invalidating the majority of the invalid paths? It turns out that a few anomalies invalidate most of the invalid paths. Figure 9 shows this relationship for the solutions computed by the four algorithms DPP*, EHS, SARK, and GAO (peer) for the data set of 18 April 2001 (multiple paths). For the solution computed by the SARK algorithm, we again treat the unlabeled edges as peer-to-peer edges. For example, one can see in the figure that 1% to 20% (depending on the algorithm) of the anomalies are already responsible for 70% of the invalid paths. The plots for the other dates are qualitatively very similar and thus not included here.

The data points for the plot were computed by first assessing for each anomaly how many paths it invalidates. This is done by traversing each invalid path until the first anomaly is found and then increasing a counter for this anomaly. The total number of anomalies with positive counters in the solutions of DPP*, EHS, SARK and GAO (peer) was 250, 318, 2,564 and 2,316, respectively. Subsequently, the anomalies are sorted by descending counters, i.e., the first anomaly invalidates the most paths. Then we compute, for each anomaly, the cumulative number of invalidated paths, i.e., the number of paths invalidated by the anomalies up to this one in sorted order. In the plot, the horizontal axis (in logarithmic scale) corresponds to the anomalies in sorted order, and the vertical axis gives the cumulative number of invalidated paths.

### E. Validation with Additional Path Sets

Following the experimental guideline of [9], we checked for the orientations computed by the five algorithms DPP, DPP*, EHS, SARK and GAO how many valid paths they achieve on the 10 individual data sets of 18 April 2001 as well as on four additional data sets that were not input of the algorithms. We consider the data with multiple paths. The extra group of data sets is, again, available from [19] and contains data from AS1755, AS2516, AS2548, and AS6893. Among the paths in these four data sets, there are 1.2%, 12.6%, 0.5% and 9.0%, respectively, that contain edges that are not present in the 10 data sets we used as input of our algorithms. We discarded these paths in our computations.

TABLE V

DETAILED VIEW OF THE RESULTS OBTAINED WITH DIFFERENT CLASSIFICATION ALGORITHMS FOR 18 APRIL 2001.

| AS # AS Name | DPP | DPP* | EHS | SARK | GAO (sib) / (peer) |
|---|---|---|---|---|---|
| 1 Genuity | 0.34% | 0.13% | 0.10% | 16.85% | 0.03% / 8.65% |
| 1740 CERFnet | 0.32% | 0.09% | 0.11% | 5.90% | 0.00% / 7.14% |
| 3549 Globalcr. | 0.12% | 0.20% | 0.25% | 17.28% | 0.01% / 12.20% |
| 3582 U. of Oregon | 0.39% | 0.25% | 0.17% | 29.75% | 0.14% / 31.86% |
| 3967 Exodus C. | 0.74% | 0.34% | 0.28% | 29.65% | 0.04% / 14.33% |
| 4197 Global O. J. | 0.34% | 1.58% | 0.13% | 13.39% | 0.05% / 81.42% |
| 5388 Energis Squ. | 0.33% | 0.19% | 0.10% | 55.35% | 0.08% / 54.51% |
| 7018 AT&T | 0.12% | 0.09% | 0.11% | 23.30% | 0.00% / 7.72% |
| 8220 COLT I. | 0.19% | 0.15% | 0.18% | 11.38% | 1.24% / 8.49% |
| 8709 Exodus, Eur. | 1.61% | 0.25% | 0.31% | 10.60% | 4.15% / 48.87% |
| 1755 Ebone | 1.07% | 0.19% | 0.22% | 5.42% | 0.02% / 4.86% |
| 2516 KDDI | 5.31% | 4.25% | 5.79% | 72.87% | 5.44% / 18.11% |
| 2548 MaeWest | 0.20% | 0.18% | 0.12% | 2.36% | 0.02% / 4.26% |
| 6893 CW | 2.13% | 1.79% | 1.22% | 16.28% | 0.12% / 13.51% |

Table V shows that algorithms DPP, DPP* and EHS leave a very small percentage of invalid paths. In particular, they perform significantly better, in terms of invalid paths, than the cutting edge heuristic SARK of [9]. Note that the numbers we give for algorithm SARK differ from the numbers provided in [9]. This seems to result from different ways of checking the validity of paths. For example, Subramanian et al. did not count as invalid Type 2 paths containing two consecutive undirected edges instead of one [28]; their motivation for relaxing the model in this way is that two ASes may have an "indirect peering", i.e., a peer-to-peer relationship through an intermediate AS. Algorithm GAO (peer) performs similarly to SARK on three of the data sets, but significantly better than SARK on the data from AS 2516.

## VII. 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 by reintroducing peering relationships. In that section a sufficient condition has been given for modifying a directed edge into an undirected edge while 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 peerings 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 peering edges is a hard one. The problem can be formally stated as follows:

*PEERING-DISCOVERY Problem*: Given a directed graph $G$, a set of paths $P$, and an integer $k$, test if it is possible to remove the orientation of $k$ edges of $G$ without increasing the number of invalid paths in $P$.

We prove its hardness by using a reduction from the following $NP$-complete problem:

*INDEPENDENT-SET Problem*: Given an undirected graph with node set $N$ and edge set $A$ and an integer $k$, find a subset of the nodes of size $k$ such that no two nodes of the subset are adjacent.

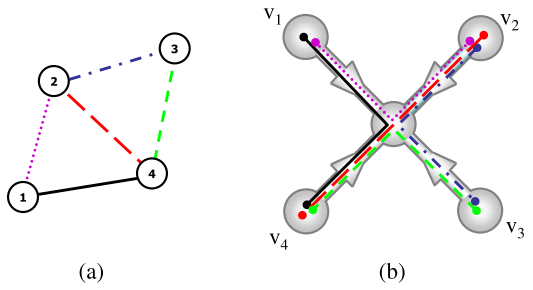


Fig. 10. An instance of the INDEPENDENT-SET problem (a) and the corresponding instance of the PEERING-DISCOVERY problem (b).

To build the instance of the PEERING-DISCOVERY 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 edge $\{n_i, n_j\} \in A$. The edges of the PEERING-DISCOVERY 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 $k$. Therefore, and since PEERING-DISCOVERY is easily seen to lie in $NP$, we obtain the following theorem.

*Theorem 5:* PEERING-DISCOVERY is $NP$-complete.

We remark that our reduction from the INDEPENDENT-SET problem to the PEERING-DISCOVERY problem provides also an approximation-preserving reduction between the maximization versions of the problems. Following arguments similar to Section IV-A, we therefore obtain that the problem of maximizing the number of reintroduced peer-to-peer edges in a graph with $n$ vertices does not admit an approximation algorithm with ratio $1/n^{1-\varepsilon}$ for any $\varepsilon > 0$ unless $NP = ZPP$.

## VIII. 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 valid paths is maximized. Also, we proved that the corresponding maximization problem is $NP$-hard in the general case (as conjectured in [9]) and cannot even be approximated well in the worst case.

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 approximation algorithms and heuristics for the general case. The approximation algorithms are guaranteed to give good solutions for instances with short AS paths. For all our algorithms we demonstrate experimentally that they are more effective than previously presented approaches with respect to the resulting number of valid paths.

The website http://www.dia.uniroma3.it/∼compunet/ contains further details on the experiments and the data sets.

Several problems remain open. While our approach produces good relationship classifications with respect to the number of valid paths, further work is necessary in order to address the problem of classifying peer-to-peer edges, which make up a significant portion of the real AS relationships. In particular, an alternative formulation of the ToR problem in which the objective function also reflects the quality of the peer-to-peer classifications would be highly desirable. 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., [29]) in order to provide information about the relationships between ASes.

## Acknowledgments

We are grateful to the authors of [9] for their help and to Fontas Dimitropoulos and Dmitri Krioukov for discovering a bug in an earlier version of our EHS code. Also, we would like to thank Massimo Rimondini for re-engineering and debugging the code for the DPP algorithm and Andrea Vitaletti and Debora Donato for interesting conversations.

## References

[1] J. W. Stewart, *BGP4: Inter-Domain Routing in the Internet*. Reading, MA: Addison-Wesley, 1999.
[2] C. Alaettinoglu, "Scalable router configuration for the internet," in *Proc. IEEE IC3N*, October 1996.
[3] G. Huston, "Interconnection, peering, and settlements," in *Proc. INET*, June 1999.

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

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

[6] W. Theilmann and K. Rothermel, "Dynamic distance maps of the internet," in *Proc. IEEE INFOCOM 2000*, March 2000.

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

[8] 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.

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

[10] G. Siganos and M. Faloutsos, "Analyzing BGP policies: Methodology and tool," in *Proc. IEEE INFOCOM 2003*, March 2003.

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

[12] T. Erlebach, A. Hall, and T. Schank, "Classifying customer-provider relationships in the internet," in *Proc. IASTED International Conference on Communications and Computer Networks (CCN)*, 2002, pp. 538–545.

[13] G. Di Battista, M. Patrignani, and M. Pizzonia, "Computing the types of the relationships between autonomous systems," in *Proc. IEEE INFOCOM 2003*, 2003.

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

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

[16] 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.

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

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

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

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

[21] J. Håstad, "Some optimal inapproximability results," *J. ACM*, vol. 48, no. 4, pp. 798–859, 2001.

[22] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[23] M. Goemans and D. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *J. ACM*, vol. 42, no. 6, pp. 1115–1145, 1995.

[24] 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)*, ser. LNCS 2337, 2002, pp. 67–82.

[25] 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*, 1995, pp. 182–189.

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

[27] D. R. Figueiredo, Z. Ge, and S. Jaiswal, "Implementation of algorithm given in [7]," *http://www-net.cs.umass.edu/∼ratton/AS/*, 2002.

[28] L. Subramanian, personal communication.

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


**Giuseppe Di Battista** received a Ph.D. in Computer Science from the University of Rome "La Sapienza" and is currently a professor in the Department of Computer Science and Automation at the Third University of Rome. His research interests include Graph Drawing, Computational Geometry, and Networking. He is one of the authors of a 1999 book by Prentice Hall on Graph Drawing and is a founding member of the steering committee for the Graph Drawing Symposium.

**Thomas Erlebach** received a Ph.D. in Computer Science from Technische Universität München in 1999. From 2000 to 2004, he was an assistant professor in Theory of Communication Networks at ETH Zürich. In September 2004 he joined the Department of Computer Science at the University of Leicester as a Reader in Algorithms. His research interests lie in the design and analysis of efficient algorithms for optimization problems arising in communication networks and other application areas.

**Alexander Hall** received a Master's degree ("Diplom") in Computer Science at the Technische Universität München in 1998. In December 2003 he completed his doctoral studies in the group of Thomas Erlebach at the ETH Zürich and received a Ph.D. for the thesis "Scheduling and Flow-Related Problems in Networks". He currently is a post-doc at ETH Zürich.

**Maurizio Patrignani** received the "Laurea" degree in Electronic Engineering at the University of Rome "La Sapienza" in 1996 and a Ph.D. in Computer Science from the same institution in 2001. He is currently a faculty member in the Department of Computer Science and Automation of the Third University of Rome. In addition to Networking his research interests include two- and three-dimensional Graph Drawing, Information Visualization, and Computational Geometry.

**Maurizio Pizzonia** received the Master's degree ("Laurea") in Software Engineering at the University of Rome "La Sapienza" in 1997 and a Ph.D. in Computer Science at the same institution in 2001 defending the Thesis "Engineering of Graph Drawing Algorithms for Applications". He is currently a faculty member at the Third University of Rome. His research interests lie in design of algorithms and software systems for Internet analysis, Graph Drawing and Information Visualization.

**Thomas Schank** studied mathematics and physics from 1994 to 2001 at the University of Konstanz, Germany. After his studies he took part in the predoctoral program in "Combinatorics, Geometry and Computation" in the winter-term 2001/2002 at the ETH Zürich, Switzerland. Thomas Schank is currently a Ph.D. student with Prof. Dorothea Wagner at the University of Karlsruhe, Germany.