Università degli Studi di Roma Tre
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79 – 00146 Roma, Italy

Computing the Types of the Relationships between Autonomous Systems

Giuseppe Di Battista, Maurizio Patrignani, and Maurizio Pizzonia

RT-DIA-73-2002 **Luglio 2002**

Dipartimento di Informatica e Automazione,
Università di Roma Tre,
Via della Vasca Navale, 79
00146 Roma, Italy.
{gdb,patrigna,pizzonia}@dia.uniroma3.it

Work partially supported by: European Commission - Fet Open project COSIN - COevolution and Self-organisation In dynamical Networks - IST-2001-33555.

ABSTRACT

We investigate the problem of computing the types of the relationships between Internet Autonomous Systems. We refer to the model introduced in [6, 17] 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.

1 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) [16].

Several authors (see, e.g. [3, 12]) 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, [10, 11, 18]).

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 [10] 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 [6] 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 [6] 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. [17] formally define, as a minimization problem, a slightly simplified version of the problem addressed in [6] 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 [6, 17]. Namely, its main results are the following.

- We solve a problem explicitly stated in [17]. 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 particular 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 [17].

The paper is structured as follows. Section 2 describes the addressed problem. Sections 3 and 4 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 5 we prove the NP-completeness of the problem in the general case. Section 6 shows new heuristics and compare the results with the state of the art. Finally, Section 7 contains conclusions and open problems.

2 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. [3, 12, 7, 17]). 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 [6, 17]. 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 [6], that has been also re-casted in [17]. A graph-theoretic formulation of the same theorem will be given in what follows.

2.1 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.

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.

An *invalid path* is a path that is not valid.

At this point the above mentioned theorem [17] 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 [17] toward identifying the following problem.

> *Type-of-Relationship (ToR) Problem* [17]: 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 1 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 (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 2 shows an instance of the ToR problem that admits an orientation without invalid paths. Figures 3 and 4 show a possible orientation.

2.2 Simplifying the problem

The Type-of-Relationship Problem is a minimization problem. In order to studying it, following a standard technique [8], we consider its corresponding decision version as follows.

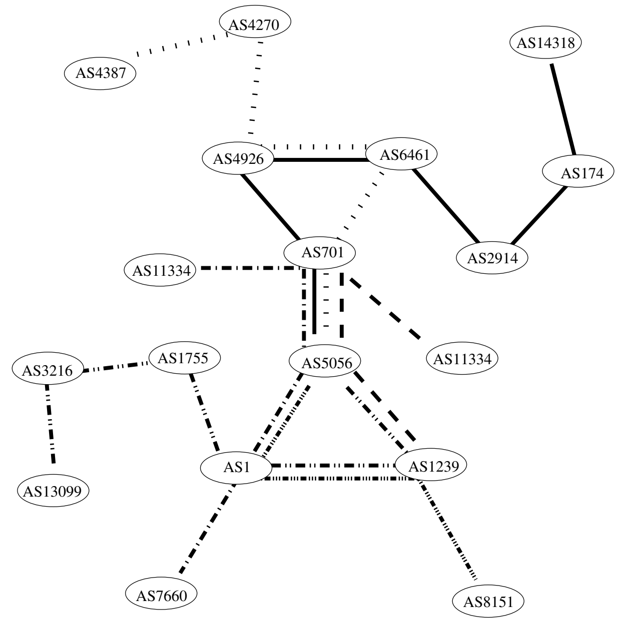

Figure 1: An instance of the ToR problem that does not admit an orientation without invalid paths. The six paths of the instance are represented with different line styles.

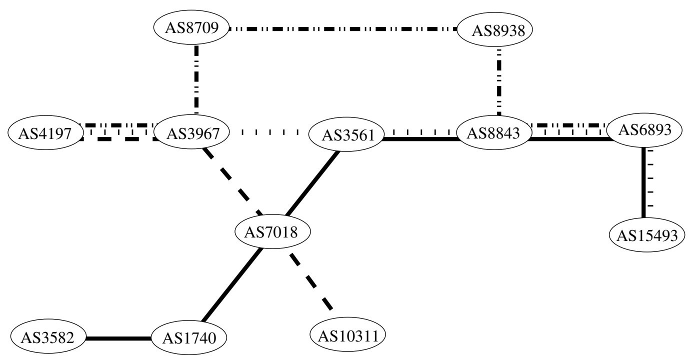

Figure 2: An instance of the ToR problem that admits an orientation without invalid paths. The four paths of the instance are represented with different line styles.

> *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.

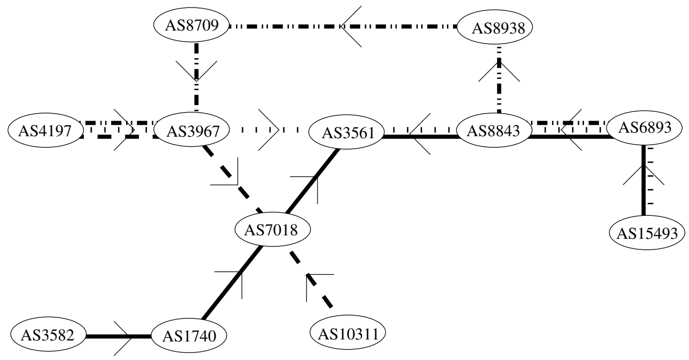

Figure 3: An orientation for the graph of Figure 2. Note that all the paths are valid.

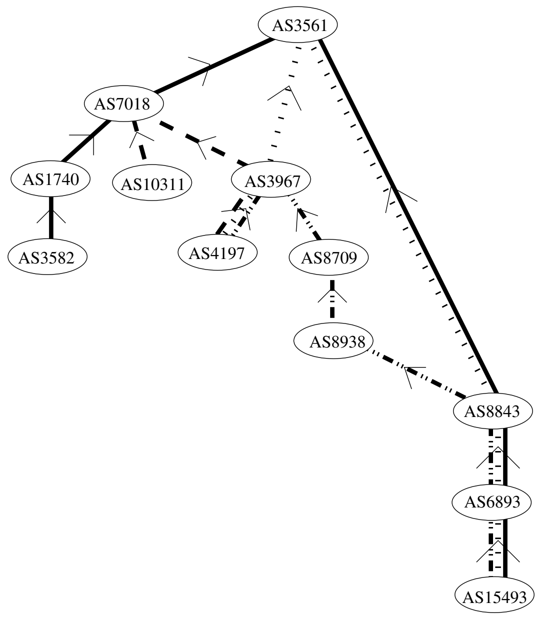

Figure 4: The directed graph of Figure 3 is drawn in such a way to emphasize the hierarchical relationships induced by the orientation.

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 5), 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.

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.

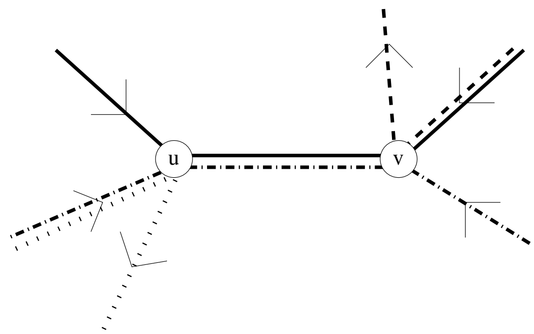

Figure 5: An undirected edge (u, v) of a solution of the ToR-D problem. The two paths traversing (u, v) are represented one with a solid line and the other with a dot-dashed line. Both paths are valid.

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.

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

In Section 2 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).

3.1 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 [8].

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 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 6.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 6.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 6.c).

3.2 Computational aspects

Problem 2SAT may be efficiently solved by using the well known result in [4] 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 4, and 6.

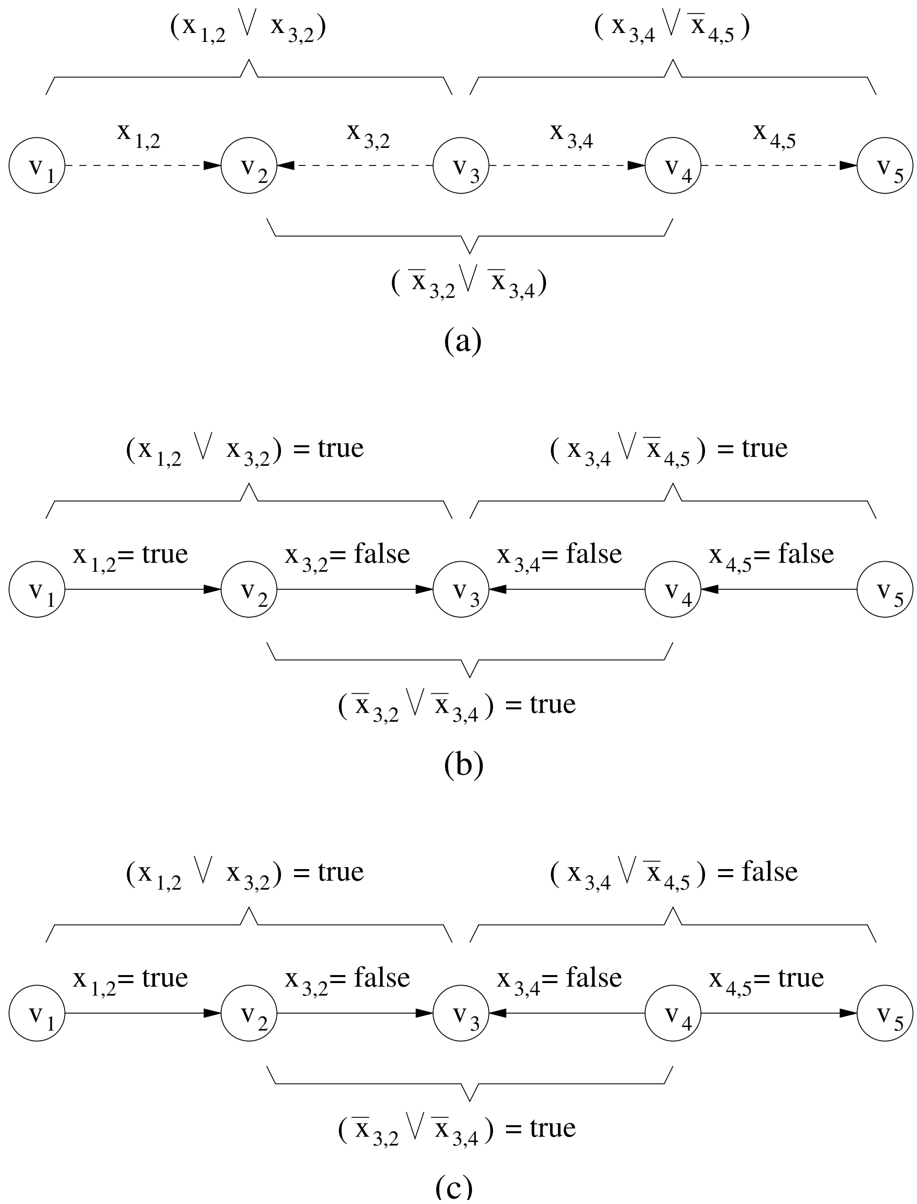

Figure 6: (a) An initial orientation for a five vertex path and the boolean variables associated with its edges. The orientation shown in (b), which makes the path valid, corresponds to the truth assignment $x_{1,2} = true$, $x_{3,2} = false$, $x_{3,4} = false$, and $x_{4,5} = false$, which satisfies the 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.

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 $(\bar{l}_1, l_2)$ and $(\bar{l}_2, l_1)$ are introduced. Intuitively, edge $(\bar{l}_1, l_2)$ represents the logical implication $\bar{l}_1 \rightarrow l_2$, while edge $(\bar{l}_2, l_1)$ represents $\bar{l}_2 \rightarrow l_1$. 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 [14] 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 [14].

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 [15] 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.

3.3 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 [17]) 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* [16] 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 [17]. Such test beds are periodically collected and publicly distributed by the authors [1].

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

Table 1: Telnet Looking Glass servers and corresponding AS graphs.

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 1 shows the main features of the graphs constructed from the ten data sets. Note that values of Tables I and IV of [17] and values computed from data available in [1] (and that are presented in the aforementioned Table 1 of this paper) appear to be slightly different.

Table 2 shows the results of the experiments. Observe that for all the partial views, but the one of the University of Oregon server [2], 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.

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

Table 2: Testing if the ToR problem has a solution without invalid paths for several BGP routing tables.

Figure 7 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 1 to give an example of an instance of the ToR problem that does not admit an orientation without invalid paths.

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.

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

Figure 7: 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.

4 Computing the AS Relationships

In Section 3 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.

4.1 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 [4] for computing such assignment is the following. A function $f(v)$ can be computed for all the vertices of the graph $G_{2\mathsf{SAT}}$ associated with 2SAT (see Section 3.2) 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* [15], 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 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.

4.2 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 2.2, it is possible to refine the obtained solution reintroducing peering relationships. In the same section a sufficient condition has been given for making undireced a directed 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 G 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 $(n_i, n_j) \in A$ (see Figure 8 for an example). 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.

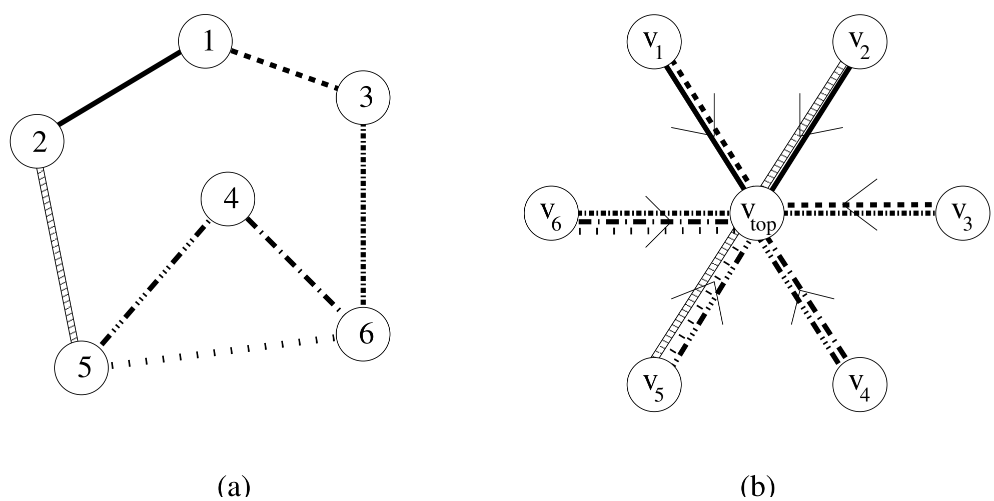

Figure 8: (a) A graph G that is an instance of the INDEPENDENT-SET problem and (b) the corresponding instance of the ToR-D-simple problem. Each arc of G (and the corresponding path of the ToR-D-simple instance) is drawn with a different line style. Finding k independent nodes of G is equivalent to finding k directed edges of the ToR-D-simple instance that can be made undirected without introducing invalid paths.

5 The Difficulty of Minimizing Path Anomalies

The ToR problem was conjectured to be NP-complete in [17]. In Section 3 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 [8], we refer to their decision versions. For ToR we already defined in Section 2 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 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

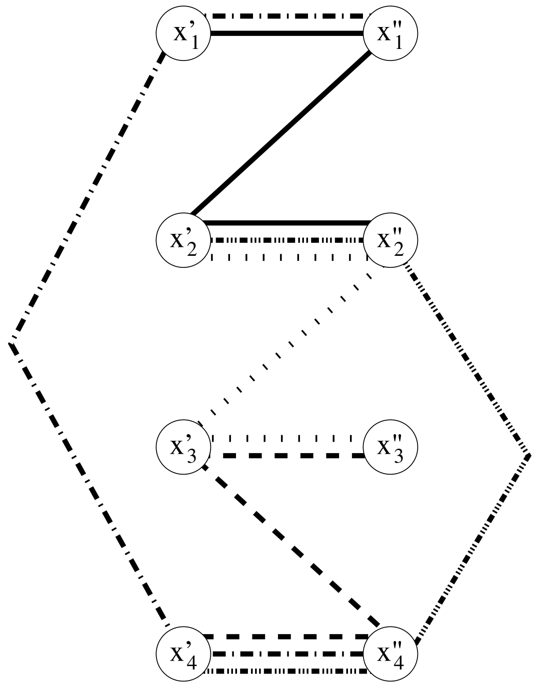

Figure 9: The instance of the ToR-D-simple problem corresponding to the instance $(x_1 \vee \overline{x}_2) \wedge (x_2 \vee \overline{x}_3) \wedge (x_2 \vee x_4) \wedge (\overline{x}_1 \vee \overline{x}_4) \wedge (\overline{x}_3 \vee x_4)$ of the MAX2SAT-D problem.

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).

6 Heuristics for Computing the AS Relationships

In Section 5 we have seen that the ToR-D problem is computationally hard and in Section 3 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 2).

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., [5]), 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 [13] 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 [17]. Namely, we consider the union of all the paths of the Telnet Looking Glasses of Table 1 (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 3. 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_{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_{2SAT} (e.g., with the algorithm in [14]).

4. Identify each variable x such that x and $\overline{x}$ are in the same strongly connected component of G_{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 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_{2SAT}.

In our data set the starting G_{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 reinserted 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 4. A fragment of the computed orientation has been used already in this paper for the example of Figures 3 and 4. Further, following the experimental guideline of [17] 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 [1] and contains data from AS1755, AS2516, AS2548, and AS6893.

AS #	AS Name	Anomalies % ([17])	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 3: Comparison between the heuristics presented in this paper and the state of the art.

Table 3 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 [17]. 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. Using the condition discussed in Section 2.2 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 3. The above experiment, involving $3,423,460$ paths, required a computation time of about 10 hours.

7 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 [17]).

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 Web-site `http://www.dia.uniroma3.it/~compunet`.

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?

Acknowledgments

We are grateful to the authors of [17] for their help. Also, we would like to thank Andrea Vitaletti and Debora Donato for interesting conversations.

References

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

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

[3] C. Alaettinoglu. Scalable router configuration for the internet. In *Proc. IEEE IC3N*, October 1996.

[4] 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*, 8(3):121–123, 1979.

[5] Uriel Feige and Michel 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*, pages 182–189, 1995.

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

[7] Lixin Gao, Timothy G. Griffin, and Jennifer Rexford. Inherently safe backup routing with BGP. In *IEEE INFOCOM 2001*, pages 547–556, Apr 2001.

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

[9] Zihui Ge, Daniel Ratton Figueiredo, Sharad Jaiswal, and Lixin Gao. On the hierarchical structure of the logical internet graph. In *Proc. SPIE ITCom 2001*, 2001.

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

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

[12] Geoff Huston. Interconnection, peering, and settlements. In *Proc. INET*, June 1999.

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

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

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

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

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

[18] Wolfgang Theilmann and Kurt Rothermel. Dynamic distance maps of the internet. In *IEEE INFOCOM 2000*, Tel-Aviv, Israel, March 2000. IEEE.