OPERATIONS RESEARCH
Vol. 57, No. 4, July–August 2009, pp. 1006–1025
ISSN 0030-364X | EISSN 1526-5463 | 09 | 5704 | 1006

DOI 10.1287/opre.1080.0677

Connectivity Measures for Internet Topologies on the Level of Autonomous Systems

Thomas Erlebach
Department of Computer Science, University of Leicester, Leicester LE1 7RH, United Kingdom,
t.erlebach@mcs.le.ac.uk

Linda S. Moonen, Frits C. R. Spieksma
Operations Research Group, Katholieke Universiteit Leuven, 3000 Leuven, Belgium
{linda.moonen@econ.kuleuven.be, frits.spieksma@econ.kuleuven.be}

Danica Vukadinović
Computer Engineering and Networks Laboratory (TIK), ETH Zürich, Switzerland,
vukadin@gmail.com

Classical measures of network connectivity are the number of disjoint paths between a pair of nodes and the size of a minimum cut. For standard graphs, these measures can be computed efficiently using network flow techniques. However, in the Internet on the level of autonomous systems (ASs), referred to as AS-level Internet, routing policies impose restrictions on the paths that traffic can take in the network. These restrictions can be captured by the valley-free path model, which assumes a special directed graph model in which edge types represent relationships between ASs. We consider the adaptation of the classical connectivity measures to the valley-free path model, where it is $\mathcal{NP}$-hard to compute them. Our first main contribution consists of presenting algorithms for the computation of disjoint paths, and minimum cuts, in the valley-free path model. These algorithms are useful for ASs that want to evaluate different options for selecting upstream providers to improve the robustness of their connection to the Internet. Our second main contribution is an experimental evaluation of our algorithms on four types of directed graph models of the AS-level Internet produced by different inference algorithms. Most importantly, the evaluation shows that our algorithms are able to compute optimal solutions to instances of realistic size of the connectivity problems in the valley-free path model in reasonable time. Furthermore, our experimental results provide information about the characteristics of the directed graph models of the AS-level Internet produced by different inference algorithms. It turns out that (i) we can quantify the difference between the undirected AS-level topology and the directed graph models with respect to fundamental connectivity measures, and (ii) the different inference algorithms yield topologies that are similar with respect to connectivity and are different with respect to the types of paths that exist between pairs of ASs.

Subject classifications: networks: computer network modeling; integer programming; analysis of algorithms.
Area of review: Telecommunications and Networking.
History: Received July 2005; revisions received May 2007, June 2008; accepted September 2008. Published online in *Articles in Advance* June 3, 2009.

1. Introduction

It is a cliché to state that stability and robustness of the Internet are fundamental for ensuring today's efficient communication. Maintaining the speed and the reliability of Internet-based communication is a prime challenge for service providers, their clients, and other involved institutions. To understand the potential vulnerability of Internet-based communication, we need to get an idea of the routes that are being used for sending traffic, of the routes that could be used for sending traffic, and how different ways of sending traffic vary with respect to their susceptibility to failing systems and/or failing connections. Recent incidents such as the Middle East fiber cut in January 2008 and the Taiwan earthquake in December 2006, each upsetting Internet traffic, emphasize this point; see also Wu et al. (2007).

A first step is, then, to discover how traffic is being sent over the Internet. This is, however, not easy to find out (Chang et al. 2004). To explain this, let us view the Internet as a set of autonomous systems (ASs; an AS is a subnetwork under separate administrative control) that are connected by physical links. ASs exchange routing information using the border gateway protocol (BGP); this is a protocol that governs the communication between a pair of ASs. More specifically, each AS uses a local routing policy that determines which routes are announced to which neighboring ASs. For commercial reasons, details about these local policies of individual ASs are not publicly available. Obviously, this makes it difficult to create an accurate model that can be used in the analysis of the robustness of the Internet.

The routing policies impose restrictions on the paths that traffic can take in the network. These restrictions can be captured by the so-called *valley-free path model*, explained below in more detail. The valley-free path model assumes a special directed graph model of the Internet on the level of

autonomous systems, referred to as AS-level Internet, that contain information about the AS relationships. Algorithms that infer such directed graph models from available routing data have been proposed in the literature.

We consider the adaptation of two classical measures of network connectivity, the number of disjoint paths and the size of a minimum cut, to the valley-free path model. Our first main contribution is to present exact algorithms (i.e., algorithms that are guaranteed to find an optimal solution) for these classical connectivity problems adapted to the valley-free path model. The algorithms are based on integer programming formulations of the problems and use paradigms such as branch-and-price and branch-and-bound. Our second main contribution is an experimental evaluation of these algorithms applied to four types of directed graph models of the AS-level Internet produced by different inference algorithms. Most importantly, the evaluation confirms that our algorithms are able to compute optimal solutions to instances of realistic size of the connectivity problems in the valley-free path model in reasonable time. Furthermore, our experimental results provide some information about the characteristics of the directed graph models of the AS-level Internet produced by different inference algorithms. Our algorithms also make it possible to quantify the difference between the undirected AS-level topology and the directed graph models with respect to fundamental connectivity measures; we further motivate the relevance of the comparison of these different graph models in §1.2. Thus, our work focuses on connectivity; for other characteristics of topologies of the AS-level Internet (such as degree), we refer to Mahadevan et al. (2005).

Let us proceed by describing the background of this work in more detail. Routing policies depend mostly on the economic relationships between ASs. They represent an important aspect of Internet structure. The main trends in the diversity of commercial agreements between ASs are discussed in Huston (1999a, b). We will refer to routing policies governed by the BGP as BGP routing policies, or BGP policies for short. A good survey of BGP routing policies and interdomain traffic engineering can be found in Quoitin et al. (2003) and Caesar and Rexford (2005). The impact of economic relationships on the engineering level, more precisely on BGP policies, has been recognized as one of the reasons for BGP path inflation (i.e., the phenomenon that traffic uses paths that are sometimes longer than necessary; see Gao and Wang 2002) and one of the important factors in route convergence analysis (i.e., the fact that when a previously valid path to a destination D becomes invalid, it can take a long time until the network has obtained a new valid path to D; see Labovitz et al. 2001). Thus, the previously adopted undirected model of the Internet, which ignores BGP policies, is only a crude approximation of reality and might produce a distorted picture of the routes used in practice. On the other hand, incorporating all of the peculiarities of the manifold contracts between ASs in a new model would add too much complexity (assuming one would know these contracts). Therefore, a coarse classification of AS relationships into three categories—customer-provider, peer-to-peer, and siblings—has been proposed (Gao 2001). More recent work has focused attention mainly on customer-provider and peer-to-peer relationships (Subramanian et al. 2002).

If ASs A and B are in a customer-provider relationship, i.e., if A is a customer of B, then B announces all its routes to A, but A announces to B only its own routes and routes of its own customers. If they are peers, they exchange their own routes and routes of their customers, but not routes of their providers or other peers. If ASs A and B are siblings, then A announces all its routes to B and B announces all its routes to A. These policies arise because customers do not want to act as transit ASs for their providers, i.e., a provider cannot route traffic through a customer to a different provider of that customer. As a consequence, only *valley-free* paths are valid, i.e., paths that first go "up" in the hierarchy and then "down" toward the destination. (A formal definition will be given in §2. In this paper, we will use the terms "valley-free path" and "valid path" interchangeably.)

Thus, one arrives at the following model. The Internet is a graph containing the ASs as vertices. The graph can have directed and undirected edges. There are three different ways in which two vertices A and B can be connected:

(i) an undirected edge between A and B. This is interpreted as "A and B are peers."

(ii) a directed edge from A to B, and a directed edge from B to A. This is interpreted as "A and B are siblings."

(iii) a directed edge from A to B, and no directed edge from B to A. This is interpreted as "A is customer of B."

The resulting graph is called a ToR graph (see §2 for formal definitions). Two ASs with at least one physical link between them are connected by a single edge (or a single pair of edges, in the case of siblings) in this model, no matter how many physical links there are between these two ASs. For comparison, note that the previously adopted *undirected graph model* of the Internet consisted of an undirected graph with an undirected edge between two ASs if there is at least one physical link between them.

Because information about the economic relationships between ASs is not publicly available (such information is often treated like a business secret; see, e.g., Chang et al. 2005), four algorithms have been proposed for inferring these relationships from BGP-routing table information (see Gao 2001, Subramanian et al. 2002, Di Battista et al. 2003, and Erlebach et al. 2002). However, it is not known how good the topologies produced by these algorithms are and how these topologies differ from each other; we intend to (partially) answer this question.

Classical measures of network connectivity are the number of disjoint paths between two nodes, and the minimum size of a cut separating these two nodes. In view of the large impact of BGP policies in the Internet, we are interested in an adaptation of these connectivity measures to the valley-free path model: We would like to compute

the maximum number of vertex-disjoint valley-free paths between two ASs and the minimum size of a valid cut between two ASs, i.e., the minimum number of vertices that must be removed from the graph so that no valley-free path between these two ASs remains. It is well known that in the standard graph models (in the standard model, a path consists of a sequence of forward arcs in the directed case and of a sequence of undirected edges in the undirected case) the maximum number of disjoint paths between s and t is equal to the size of a minimum s-t cut (provided that s and t are not adjacent); moreover, the corresponding solutions can be computed efficiently (see Ahuja et al. 1993). In a ToR graph this is not the case. It is $\mathcal{NP}$-hard to compute the maximum number of vertex-disjoint s-t paths; it is also $\mathcal{NP}$-hard to compute the minimum size of a valid s-t cut. The best known approximation ratio is two for both problems. Also, the minimum size of an s-t cut can be up to twice the maximum number of vertex-disjoint s-t paths. Thus, the max-flow min-cut equality holds only approximately for ToR graphs (Erlebach et al. 2005). Here, we present algorithms that are able to obtain optimal solutions with a moderate amount of computation time using exact approaches, one of which involves applying a branch-and-price algorithm. Furthermore, we apply our new exact algorithms for the computation of connectivity measures in the valley-free path model to the directed graph models produced by four different inference algorithms. Our experimental results confirm that it is feasible to compute the minimum size of an s-t cut as well as a maximum number of vertex-disjoint paths, in real AS-level Internet graphs in reasonable time. We also compare the results from the exact methods with those of the 2-approximation algorithms from Erlebach et al. (2005), thus providing some insight into the solution quality of these known approximation algorithms on realistic instances.

We also compute disjoint paths and minimum cuts in the undirected Internet graph and compare the results with those of the different directed models, thus allowing us to quantify the difference in connectivity between the undirected and directed graph models.

Additionally, our experimental results also provide information about the characteristics of the directed graph models obtained with different inference algorithms. We further investigate the differences between the four inference algorithms by also considering other aspects of the graph models they produce. Apart from investigating the connectivity of the different graph models, we report statistics concerning the number of ASs that are connected by a directed path in the different directed models, a quantity that is related to the depth of the provider hierarchy and to the *customer-preference* aspect of current interdomain routing (Feigenbaum et al. 2002). The latter means that paths through customers are preferred over paths through peers, and these to paths through providers. We also study directed customer-provider cycles, which are a somewhat unexpected structure, in the directed graph models. We claim that these cycles can help to detect misclassified relationships and thus improve the accuracy of the Internet topology; see §1.2 for a further motivation.

In summary, our investigations address the following research questions:

- In a graph on the scale of the Internet (containing up to 11,000 vertices and 30,000 edges, for the February 2004 snapshot, after pruning vertices of degree one), is it feasible to compute exact solutions to the $\mathcal{NP}$-hard problems of finding a maximum number of vertex-disjoint valley-free paths between two ASs or the size of a minimum cut? Such computations could prove useful for ASs that want to evaluate different options for selecting upstream ISPs to improve the robustness of their connection to the Internet or, more generally, in future investigations of robustness issues of the Internet.
- How does the performance of the 2-approximation algorithms proposed in Erlebach et al. (2005) compare to the performance of the exact algorithms, both in the quality of the solutions found and in the running times?
- How large are the differences between the undirected graph model and the directed graph models with respect to connectivity properties?
- How do the directed graph models produced by the four algorithms proposed in Gao (2001), Subramanian et al. (2002), Di Battista et al. (2003), and Erlebach et al. (2002) compare to each other? Here, we are mainly interested in (i) comparing connectivity measures, (ii) the depth of the provider hierarchy, and (iii) the occurrence of directed customer-provider cycles in the directed graph models.

1.1. Related Work

Our starting point for the interpretation of BGP policies is the work described in Gao (2001), which addressed the problem of unavailable information about the exact relationships between ASs. A heuristic algorithm was proposed for inferring AS relationships from BGP routing tables. In addition, it was observed that a path between a pair of ASs follows a particular structure: No path contains more than one peer-to-peer relationship, and once a provider-customer or peer-to-peer relationship is encountered in the path, no customer-provider relationship can follow. If we ignore sibling relationships for the moment and imagine that providers are at a higher level than their customers, and peers are at the same level, the valid paths are "only up," "only down," or "first up and then down." Valid paths can have only one "peak" (which can consist of a single AS or of two ASs connected by a peer-to-peer relationship) and they must not contain "valleys." Therefore, such paths are also called *valley-free* paths. We use the same characterization of valid paths in this paper.

Further work trying to infer AS relationships is presented in Subramanian et al. (2002). They formalize the problem by posing it as the optimization problem of giving an orientation to the edges of an undirected AS graph with the objective of maximizing the number of paths in the given

BGP tables that become valid for this orientation. They pose the complexity of this problem as an open question. They also give a heuristic algorithm that infers relationships by first ranking all ASs and then applying certain rules to decide about the relationships between pairs of ASs using the rank values. Independently obtained results in Di Battista et al. (2003) and Erlebach et al. (2002) resolve the open question of Subramanian et al. (2002) and prove this inference problem to be $\mathcal{NP}$-hard. Two heuristic algorithms for calculating approximately optimal orientations with respect to the number of valid paths are also presented in Di Battista et al. (2003) and Erlebach et al. (2002), respectively.

In Rimondini et al. (2004), the algorithms from Subramanian et al. (2002) and Di Battista et al. (2003) are compared with respect to two measurements. First, the AS relationships that are found by a certain algorithm on data sets from different moments in time are considered (called the *stability* in the paper). Second, the AS relationships found by the two algorithms on the same data set are taken into account (this is referred to as *algorithm independence*). They conclude that both algorithms produce highly stable results, and that the AS relationships found by both algorithms are very similar. This leads the authors to the conclusion that the valley-free path approach leads to reliable results.

In another paper (Xia and Gao 2004), a comparison of the algorithms from Gao (2001) and Subramanian et al. (2002) is performed. Furthermore, in this paper, a new algorithm for inferring AS relationships is proposed, which is also taken into account in the comparison. The authors evaluate the accuracy of the three algorithms using partial AS relationships obtained from BGP community attribute and IRR (Internet Routing Registry) databases. They conclude that the new algorithm proposed in the paper outperforms the algorithms from Gao (2001) and Subramanian et al. (2002).

In Teixeira et al. (2003), the number of vertex- and edge-disjoint paths is computed for the undirected model of the Internet AS topology, as well as for the topology of one Internet service provider (ISP). They did not take routing policies into account.

1.2. Motivation

In this paper, we investigate the number of vertex-disjoint valley-free paths and the size of a minimum valid cut. These criteria serve as a measure of the connectivity between a pair of ASs, and they are natural generalizations of the traditional max-flow and min-cut criteria for ordinary graphs. We show how these two measures can be computed exactly and in reasonable time for AS-level Internet graphs that are constrained by BGP policies. Both connectivity measures are important in practice. The size of a minimum cut represents the minimum number of other ASs that must fail to disconnect a given pair of ASs; assuming that the routing protocol automatically finds a path as long as a valid path exists in the network, the size of a minimum cut is thus the natural metric for the robustness of the Internet connection between two ASs. The number of disjoint valley-free paths between two ASs, on the other hand, is more relevant in a setting where important data are sent simultaneously along different paths to make sure that at least one copy of the data reaches the receiver. This might allow us to investigate potential gains of multipath extensions to BGP. If data are sent along k disjoint paths between two ASs, the data can be received even if up to $k-1$ arbitrary ASs fail. Interestingly, because there can be a gap of factor two between the minimum cut size and the maximum number of disjoint paths in the valley-free path model, these two views of robustness are not equivalent in this model: Even if the minimum cut has size k, it is not always possible to find k static paths that make the data transmission resilient against $k-1$ failures.

Our algorithms can be used to evaluate adjustments to the topology at the AS level, both from the viewpoint of an individual AS as well as from a more global point of view. As discussed in Chang et al. (2003a, 2006), ASs periodically revise their (peering) relationships with other ASs based on different criteria, among which reachability of other parts of the Internet is a prime one. More specifically, an existing AS might consider improving the robustness of its connection to the Internet by changing or adding peering relationships with ISPs. Our algorithms could serve as a tool to evaluate the gains in connectivity arising from different choices of peering partners. Similarly, consider the situation of a newly created AS that wishes to connect to several ISPs to achieve a very robust connection to the Internet. Using our algorithms, the new AS could evaluate, for different choices of ISPs, the resulting connectivity to other important ASs in the Internet. More precisely, for each potential choice of providers, the new AS could use our algorithms to determine the robustness of its connectivity to the most important ASs to/from which it expects to send/receive a significant amount of traffic. In this way, the new AS could identify an optimal choice of providers (given its budget and the potential providers and their costs); see also Wang and Loguinov (2006).

It is also conceivable that organizations (or governments) that have an interest in helping to maintain a high level of robustness in a larger part of the Internet would use our algorithms to evaluate the connectivity and identify potentially problematic bottlenecks, indicated by small valid cuts between important ASs. In this way, our results may be helpful to achieve more resilient and efficient interdomain routing.

By computing connectivity measures on the directed graph models as well as the underlying undirected graph, we can also *quantify* the impact of BGP routing policies on connectivity measures. The significant difference that we observe in the connectivity of the directed and undirected models confirms the large impact of BGP policies on routing in the Internet, which has already been observed in a

number of other contexts (Gao and Wang 2002; Labovitz et al. 2001; Tangmunarunkit et al. 2001a, b, 2003). In addition, the observed gap between the undirected and directed model might be used to explore to what extent backup routes (routes not used because they violate business relationships) can improve robustness (see also Wang et al. 2007).

Furthermore, we perform extensive computational experiments with our new exact algorithms on four types of topologies of the AS-level Internet. Apart from demonstrating the usability of our algorithms on realistic instances, our investigations also provide useful information on the similarities and differences between the topologies produced by the four different inference algorithms from Gao (2001), Subramanian et al. (2002), Di Battista et al. (2003), and Erlebach et al. (2002). We believe that it is important to learn more about the characteristics of the topologies produced by different inference algorithms because recent results about measurements on the AS level of the Internet have shown that there is a need for a simple and accurate algorithm to infer relationships; see Spring et al. (2003) about path inflation in inter- and intradomain routing, Akella et al. (2003b) about multihoming (i.e., the phenomenon that customers tend to have more than one external link to different providers to guarantee the reliability of their network), and Akella et al. (2003a) about scaling properties of the Internet regarding link congestion. Thus, in addition to the two connectivity measures, we study aspects such as the depth of the provider-hierarchy in the different topologies and the presence of so-called customer-provider cycles. When two ASs are connected via different paths, there might be an incentive to prefer one path over the other. As described in Spring et al. (2003), routing through a customer brings profit, through a peer is neutral, and through a provider incurs costs for the sender. Obviously, this may have implications for the particular path chosen, and therefore we report statistics about the depth of hierarchy and so-called *customer chains* in the directed graph models. Finally, we investigate the occurrence of directed customer-provider cycles. Our analysis indicates that this concept can be useful for detecting misclassifications.

1.3. Outline

Section 2 gives formal definitions of the concepts that we require in this paper, and we discuss a primal-dual formulation of the problem. Section 3 deals with the computation of vertex-disjoint valid paths; §4 describes how we compute minimum cuts with respect to valid paths. Section 5 reviews the known 2-approximation algorithms for solving both problems. In §6, we present our experimental results concerning the number of vertex-disjoint valid paths and the sizes of minimum cuts in the four different models with inferred relationships and the undirected model. We discuss their implications and also the differences that we observe in the depth of the provider hierarchy in the different models. Statistics about directed cycles in the graphs are given, and some examples where they can be used to detect misclassifications are shown. We conclude in §7 by summarizing our results and drawing conclusions.

2. Problem Description

To formulate the problem, we first state some preliminaries in §2.1. Then, in §2.2, we give a mathematical formulation for the problems of finding the maximum number of vertex-disjoint paths and minimum cut sizes.

2.1. Preliminaries

In Subramanian et al. (2002), Di Battista et al. (2003), and Erlebach et al. (2002), the problem of inferring the AS relationships in the Internet is referred to as the type of relationship (ToR) problem. Following this terminology, we construct a graph $G = (V, E)$, called a *ToR graph*, as follows: The vertices of G are the ASs. As mentioned before, a directed edge from u to v, where $u, v \in V$, together with a directed edge from v to u, means that u and v are siblings. A directed edge from u to v means that u is a customer of v, and an undirected edge means that u and v are in a peer-to-peer relationship. In a ToR graph, a directed edge from u to v is denoted by (u, v), and an undirected edge between u and v by $\{u, v\}$.

We define a path $p = (v_1, v_2, \ldots, v_r)$ from v_1 to v_r in a ToR graph $G = (V, E)$ to be *valid* if it satisfies one of the two following conditions:

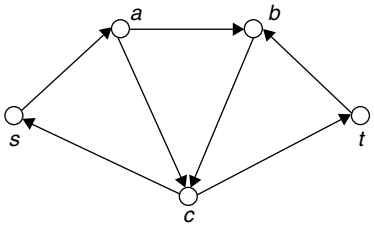
1. There exists some j, $1 \leqslant j \leqslant r$, such that $(v_i, v_{i+1}) \in E$ for $1 \leqslant i \leqslant j-1$ and $(v_i, v_{i-1}) \in E$ for $j+1 \leqslant i \leqslant r$.
2. There exists some j, $1 \leqslant j \leqslant r$, such that $(v_i, v_{i+1}) \in E$ for $1 \leqslant i \leqslant j-1$, $\{v_j, v_{j+1}\} \in E$, and $(v_i, v_{i-1}) \in E$ for $j+2 \leqslant i \leqslant r$.

Otherwise, a path is called *invalid*. This definition of valid paths captures the notion of "valley-free" paths arising from BGP routing policies. From now on, whenever we talk about paths in a ToR graph, we refer to valid paths. A path from s to t is also called an *s-t path*. Note that the reverse of an s-t path is a t-s path; hence, the direction of a valid path is not important. Two s-t paths are called *vertex-disjoint* if they do not share any vertices except s and t.

Let $p = (v_1, v_2, \ldots, v_r)$ be a valid path from s to t. We can divide p into a forward part and a backward part at some node v_j, such that $(v_i, v_{i+1}) \in E$, $i = 1, 2, \ldots, j-1$ (we know by definition that such a j exists; if j is not unique, we simply choose the maximal value for j). If p contains only directed edges, we say that a node v_l is on the forward part of p if $l < j$, v_l is on the backward part of p if $l > j$, and v_l is the node where p changes direction if $l = j$. If p contains an undirected edge, we say that a node v_l is on the forward part of p if $l \leqslant j$, and v_l is on the backward part if $l > j$.

Let $G = (V, E)$ be a ToR graph. For two nonadjacent vertices s and t in G, a *minimum valid s-t cut* in G is a

Figure 1. Gap between number of disjoint paths and minimum cut size.



set of vertices $C \subseteq V\backslash\{s,t\}$ of minimum cardinality such that there is no s-t path in the ToR graph $G\backslash C$ (i.e., in the graph that is obtained from G by deleting the vertices in C and their incident edges). Note that a minimum valid s-t cut is a smallest set of ASs whose failure disconnects s and t if only valid paths are allowed.

A *directed cycle* $v = (v_1, v_2, \ldots, v_r)$, $r > 2$, in a ToR graph $G = (V, E)$ is defined in the usual sense, i.e., the vertices $v_1, v_2, \ldots, v_r$ are distinct and we have $(v_i, v_{i+1}) \in E$ for $i = 1, \ldots, r-1$ and $(v_r, v_1) \in E$.

As mentioned before, the maximum number of vertex-disjoint paths can be strictly less than the number of nodes in a minimum cut; we give an example from Erlebach et al. (2005) to illustrate this. In Figure 1, we see that the maximum number of vertex-disjoint paths is equal to one, whereas the size of a minimum cut equals two. Indeed, one can verify that the set of valid s-t paths equals $\{(s,a,b,t), (s,a,b,c,t), (s,a,c,t), (s,a,c,b,t), (s,c,b,t)\}$, and thus the maximum number of vertex-disjoint paths is equal to one. Furthermore, one can easily verify that a minimum cut has at least size 2 because after removing one of the nodes a, b, or c, there is still a valid path connecting s and t.

2.2. Problem Formulation

Let us now give two integer programming formulations: the first formulation (denoted by P) models the problem of finding a maximum number of vertex-disjoint paths between s and t, the second formulation (denoted by D) models the problem of finding a minimum-sized set of nodes such that each path between s and t contains at least one node from this set.

Let $G = (V, E)$ be a ToR graph and s, t two distinct vertices of G. Assume that there is no direct edge between s and t (otherwise, we remove the direct edge, compute the maximum number of vertex-disjoint s-t paths, and add one to the result). Denote by $\mathcal{P}$ the set of all valid s-t paths in G, and let $\mathcal{V}_p$ be the set of all vertices contained in path $p \in \mathcal{P}$, except for s and t. Further, we define a decision variable x_p for each valid path p, as follows:

$$x_p = \begin{cases} 1 & \text{if valid path } p \text{ is in the solution,} \\ 0 & \text{otherwise.} \end{cases}$$

Using a set-packing formulation, we get the following integer programming formulation:

$$\text{(P)} \quad \max \sum_{p \in \mathcal{P}} x_p \tag{1}$$

$$\text{s.t.} \sum_{p:\, v \in \mathcal{V}_p} x_p \leqslant 1 \quad \forall\, v \in V\backslash\{s,t\}, \tag{2}$$

$$x_p \in \{0,1\} \quad \forall\, p \in \mathcal{P}. \tag{3}$$

The objective (1) is to maximize the number of paths between s and t. Constraints (2) state that each vertex (except for s and t) can belong to at most one path, and constraints (3) are the zero-one constraints on the x_p variables.

The second formulation has a variable y_v for every $v \in V\backslash\{s,t\}$:

$$y_v = \begin{cases} 1 & \text{if vertex } v \text{ is in the } s\text{-}t \text{ cut,} \\ 0 & \text{otherwise.} \end{cases}$$

The second formulation can now be given as follows:

$$\text{(D)} \quad \min \sum_{v \in V\backslash\{s,t\}} y_v \tag{4}$$

$$\text{s.t.} \sum_{v \in \mathcal{V}_p} y_v \geqslant 1 \quad \forall\, p \in \mathcal{P}, \tag{5}$$

$$y_v \in \{0,1\} \quad \forall\, v \in V\backslash\{s,t\}. \tag{6}$$

A property of formulations (P) and (D) is that the LP-relaxation of (P) and the LP-relaxation of (D) constitute a primal-dual pair of linear programs. Further, notice that formulation (P) has exponentially many variables (because the number of valid s-t paths can be exponential in the number of vertices), and, equivalently, formulation (D) has exponentially many constraints.

3. Problem P: Vertex-Disjoint Paths in ToR Graphs

In this section, we present two exact algorithms for solving problem (P); i.e., for finding the maximum number of vertex-disjoint paths in ToR graphs. The first one is a branch-and-price algorithm based on the integer programming formulation (1)–(3) (§3.1), and the second algorithm is a branch-and-bound method in which a max-flow computation has to be performed in each node of the search tree (§3.2).

3.1. A Branch-and-Price Algorithm

Branch-and-price is a technique for solving integer programs with a huge number of variables. We refer to Barnhart et al. (1998) or Vanderbeck and Wolsey (1996) for a thorough description of this technique. Here we apply it to solving instances of formulation (P). There are (at least) two important issues to be considered when developing a branch-and-price algorithm: (i) how to solve the pricing

problem (this enables us to conclude that either we have solved the LP-relaxation of (P), or we have identified a new variable (column) to be added to the restricted master (see §3.1.1)); (ii) how to branch. We need to develop a partition of the solution space in such a way that the efficient solvability of the pricing problem is preserved (see §3.1.2).

3.1.1. Column Generation. We start by generating a feasible solution (consisting of a set of vertex-disjoint paths) as follows. We apply a simple breadth-first search to find a valid path between s and t, we add this path to the solution, and remove all nodes in this path (except s and t) from our graph. Then, a new iteration starts, and we repeat the breadth-first search until no more valid paths can be found. The resulting set of paths found by this *iterated breadth-first search* is denoted by $\mathcal{P}'$, and its value (number of disjoint paths) is referred to as $V_{\mathcal{P}'}$. We consider the restriction of the LP-relaxation of (P) to the variables x_p for $p \in \mathcal{P}'$ (the *restricted master problem*). We solve the restricted master using an LP-solver and obtain a solution to the restricted primal program and its corresponding dual. Let us call the dual solution y^*. Now we need to check whether y^* is also a feasible solution to the dual program that includes constraints for all paths $p \in \mathcal{P}$. In other words, we need to check whether there exists a valid s-t path p in the graph such that $\sum_{v \in p} y^*_v < 1$. This problem is known as the *pricing problem* (Vanderbeck and Wolsey 1996). We can solve the pricing problem in polynomial time, thereby implying that the LP-relaxation of formulation (P) (as well as the LP-relaxation of (D)) can be solved in polynomial time.

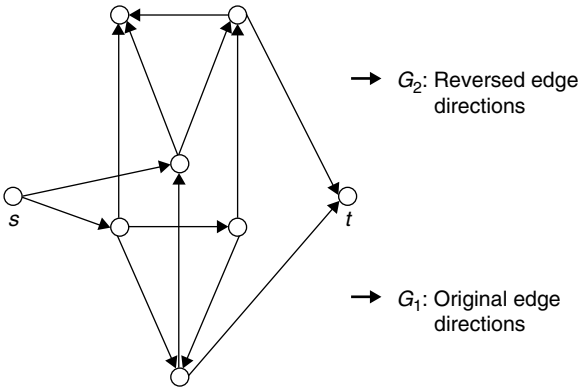
Claim 1. *The LP-relaxation of* (*P*) *can be solved in polynomial time.*

Proof. We prove the claim by showing that the pricing problem can be reduced to a shortest-path problem. The result then follows from the "separation = optimization" result in Grötschel et al. (1988).

Consider the so-called 2-layer graph that has been proposed in Erlebach et al. (2005) (we first assume that there are no undirected edges in G): Two copies G_1 and G_2 of graph G are created, but in G_2 all edge-directions are reversed. Then, so called "vertical edges" are added, i.e., directed edges from the vertices in G_1 to their copies in G_2. Finally, s in G_1 is identified with its copy in G_2 and all edges that end in s are removed; and t in G_1 is identified with its copy in G_2 and all edges that start in t are removed. In this way, all valid s-t paths in G correspond to directed paths from s to t in the 2-layer model: If a valid s-t path in G first uses some forward edges and then some backward edges, its forward part in the 2-layer model lies in G_1, then it switches to the second layer using a vertical edge, and then it again goes forward to t in G_2 because of the inverted edge-directions. (See Figure 2 for an illustration.)

We can deal with undirected edges in the following way. For an undirected edge $\{a, b\}$, we do not add corresponding edges to G_1 or G_2, but instead add directed edges (a_1, b_2) and (b_1, a_2) to the 2-layer model, where a_1, a_2 (b_1, b_2) are the copies of a (b) in G_1 and G_2, respectively. This ensures that valid s-t paths in G that include an undirected edge also have a corresponding path in the 2-layer model.

Figure 2. The 2-layer graph of the ToR graph depicted in Figure 1.



Next, we define the edge weights of the 2-layer graph as follows: Edges entering a copy of vertex v get weight y^*_v, except for vertical edges, which get weight 0. Observe that a shortest directed path from s to t in the 2-layer model gives us a valid s-t path in G that minimizes the sum of the y^*_v values of the vertices on the path. Because the shortest-path problem can be solved in polynomial time (using for instance Dijkstra's algorithm), we can solve the pricing problem in polynomial time. Q.E.D.

If the solution of the pricing problem produces a valid s-t path p such that the sum of y^*_v values on this path is less than one, we add path p to the restricted master and repeat the procedure. If there is no such path, we are done and have obtained an optimal solution to the LP-relaxation of (P). If the obtained solution is fractional, i.e., contains variables whose values are strictly between zero and one, we use a branching strategy to arrive at an integral solution.

3.1.2. Branching. If the optimal solution to the linear programming relaxation is fractional, a natural approach is to try different ways of fixing these variables to integers and solving the problem recursively for each of these alternatives (branching). Here it is important to preserve the form of the pricing problem and its efficient solvability in the branching procedure. We achieve this as follows.

Given a feasible, optimal solution x^* to the LP-relaxation of (P), we call a vertex *fractional* if it has at least three incident edges that lie on different valid paths with value $x^*_p > 0$. Notice that if a solution is fractional, it has at least one fractional vertex. Our branching strategy is as follows: For a fractional vertex w, we delete all edges incident to w except two that could lie consecutively on some valid path. Each possible way of doing this forms a branch. Thus, for example, if w has k incoming edges and l outgoing edges, the number of branches is $kl + \binom{k}{2}$. If there are many fractional

vertices, we choose one for branching that has a maximum number of incident edges lying on fractional paths.

In this way, we exclude the current fractional solution, but do not exclude any integral solution, and the problem structure is preserved: In each branch, we solve a problem of the same type on a graph with fewer edges.

For each branch, if the value of the fractional solution is not larger than the value of the best integral solution found so far, we do not enter that branch. Otherwise, we explore all branches in a depth-first traversal. In this way, we are sure to arrive at an optimal integral solution to (P).

We remark that our approach can be adapted easily to a version of the problem where each vertex v has an integral capacity c_v, and we allow up to c_v valid paths passing through it. (Here, valid paths could occur more than once in the solution.) To solve this version of the problem, we simply replace each vertex v by c_v copies and then apply our algorithm as described above.

The branch-and-price algorithm for a given a ToR graph G and two distinct vertices s and t is summarized by the pseudocode given in Figure 3.

3.1.3. Valid Inequalities. To strengthen the LP-relaxation, a natural strategy is to add valid inequalities. In this section, we discuss a class of inequalities that is valid for formulation (P) of the vertex-disjoint paths problem. We refer to these inequalities as *triangle inequalities*.

Figure 3. Pseudocode of the branch-and-price algorithm.

BRANCH-AND-PRICE ALGORITHM VERTEXDISJOINTPATHS

1. Calculate an initial solution consisting of a set of paths $\mathcal{P}'$ with value $V_{\mathcal{P}'}$ using the iterated breadth-first search, and let $V^* = V_{\mathcal{P}'}$. Create a list L and add to L a branching node corresponding to the input graph G.
2. $L = \varnothing$?
   YES: STOP. An optimal solution is found with value V^*.
   NO: Select the next branching node G from L (i.e., the branching node that was added most recently to L), remove it from L, calculate a set $\mathcal{P}'$ of edge-disjoint s-t paths in G using iterated breadth-first search, and continue with Step 3.
3. Solve the LP-relaxation using only those variables that correspond to a path in $\mathcal{P}'$.
4. Solve the pricing problem. Is there a variable (a path) with negative reduced costs?
   YES: Add this variable to $\mathcal{P}'$ and go to Step 3.
   NO: An optimal solution to the LP-relaxation is found with value V_{LP}. Continue with Step 5.
5. $V_{\mathrm{LP}} > V^*$?
   YES: Continue with Step 6.
   NO: Go to Step 2.
6. Is the solution to the LP-relaxation integral?
   YES: $V^* = V_{LP}$. Go to Step 2.
   NO: Select a fractional vertex v. For each possible way of deleting all edges incident to v except for two edges that could lie consecutively on some valid path, create a new branching node (i.e., the graph obtained by deleting the respective edges) and add it to L. Go to Step 2.

As the name suggests, we consider triangles in the ToR graphs. We define a triangle as a subset of three vertices that are connected with customer-provider edges in such a way that they do not form a directed cycle. For example, if there are three vertices a, b, and c, and there is an edge from a to b, an edge from a to c, and a third edge from b to c, this is a triangle. For each such triangle $t = (a, b, c)$, we define $T_{abc} = \{p \in \mathcal{P} \mid p \text{ contains } \{a, b\} \text{ or } \{a, c\} \text{ or } \{b, c\}\}$.

Now, for every triangle t in a ToR graph, the following inequality states that the sum of all valid paths using one of the three edges in t must be smaller than or equal to one:

$$\sum_{p:\, p \in T_{abc}} x_p \leqslant 1 \quad \forall \text{triangles } (a, b, c) \in V^3. \tag{7}$$

It is clear that inequalities (7) are valid for (P). One can view inequalities (7) (as well as inequalities (2)) as a manifestation of clique inequalities for the node-packing problem. Indeed, when we build a graph in which there is a node for every path $p \in \mathcal{P}$, and where two nodes are connected iff the two corresponding paths share a vertex in the ToR graph, it is obvious that the node-packing problem on this graph is exactly problem (P). Notice that inequalities (2) and (7) need not constitute all clique inequalities in the node-packing graph. In §6.2, we report briefly on the computational effectiveness of these inequalities.

3.2. A Branch-and-Bound Algorithm

Our second algorithm for solving the vertex-disjoint paths problem is a non-LP-based branch-and-bound algorithm, in which we use the same 2-layer graph representation as explained in §3.1.

We start with an initial solution, computed by the iterated breadth-first search discussed in §3.1. The value of this solution is a lower bound on the integer optimum. Next, we compute a maximum flow in the 2-layer graph, where we assign a capacity of one to each vertex. The flow we find is not necessarily vertex disjoint (because it may happen that the maximum flow found uses a node in G_1 and its copy in G_2), so it is an upper bound on the optimal solution. We first check whether the flow found by the maximum flow procedure is vertex disjoint, or equal to the lower bound, in which case we have found the integer optimum. Otherwise, we have to branch, which we do as follows:

In every node in the search tree, we select a vertex v from the original graph that is used more than once in the flow found by the maximum flow procedure. This vertex v has a copy v_1 in the first layer, and a copy v_2 in the second layer of the 2-layer graph. Now we generate two new branches as follows:

In the first branch, we delete vertex v_1, and all its adjacent edges, from the 2-layer graph. In the second branch, we delete all incoming edges of v_2, except for the vertical edge (v_1, v_2), from the 2-layer graph. Next, we perform a maximum flow calculation in each branching node, and repeat this procedure until we have found the integer optimum. The correctness of the branching step follows from the observation that if a node occurs in a path of

Figure 4. Pseudocode of the branch-and-bound algorithm.

BRANCH-AND-BOUND ALGORITHM VERTEXDISJOINTPATHS

1. Calculate an initial solution consisting of a set of paths $\mathcal{P}'$ with value $V_{\mathcal{P}'}$ using iterated breadth-first search, and let $V^* = V_{\mathcal{P}'}$. Create a list L and let $L = \varnothing$.
2. Construct the 2-layer graph H, and add to L a branching node corresponding to H.
3. $L = \varnothing$?
   YES: STOP. An optimal solution is found with value V^*.
   NO: Select the next node H from L (i.e., the branching node that was added most recently to L), remove this node from L, and continue with Step 4.
4. Calculate a maximum flow MF with value V_{MF} in the 2-layer graph H.
5. $V_{MF} > V^*$?
   YES: Continue with Step 6.
   NO: Go to Step 3.
6. Does the maximum flow MF in H correspond to vertex-disjoint paths in G?
   YES: $V^* = V_{MF}$. Go to Step 3.
   NO: Select a vertex v that is used more than once in MF. Create two new branching nodes as follows:
   i. Delete copy v_1 of v from the 2-layer graph.
   ii. Delete all incoming edges of copy v_2 of v from the 2-layer graph, except for the edge (v_1, v_2).
   Add the branching node corresponding to each of these two branches to L. Go to Step 3.

the solution, it is either on the backward part of the path, which is permitted in the first branch, or it is on the forward part or it is the node where the path changes direction (see §2.1), which is permitted in the second branch.

The branch-and-bound algorithm is summarized by the pseudocode given in Figure 4. In our implementation, we actually compute min-cost maximum flows (all edges are assigned a cost of one) instead of standard maximum flows because we expect that maximum flows using a minimum total number of edges can reduce the number of branching nodes required.

4. Problem D: Minimum Cuts in ToR Graphs

We solve the minimum cut problem using the dual (D) presented in §2.2. Because (D) might have exponentially many constraints, we first compute a set $\mathcal{P}'$ of vertex-disjoint s-t paths using the iterated breadth-first search as described in §3.1 and start by solving the LP-relaxation of (D) using only the constraints for paths in $\mathcal{P}'$. Solving this small linear program gives us a solution y^*. Then, we check whether there is a valid path such that $\sum_{v \in p} y_v^* < 1$, again using a shortest-path algorithm in the 2-layer model of the graph (i.e., we solve the separation problem with respect to constraints (5)). If such a path is found, we add the corresponding constraint to our linear program (D) and repeat the procedure until no more valid paths with $\sum_{v \in p} y_v^* < 1$ can be found. The resulting solution y is an optimal solution

Figure 5. Pseudocode of the branch-and-cut algorithm.

BRANCH-AND-CUT ALGORITHM MINCUT

1. Let $V^* = \infty$. Create a list L and add to L a branching node corresponding to an empty set of inclusion/exclusion constraints.
2. $L = \varnothing$?
   YES: STOP. An optimal solution is found with value V^*.
   NO: Select the next node C from L (i.e., the branching node that was added most recently to L), remove this node from L, calculate a set of vertex-disjoint s-t paths $\mathcal{P}'$ using iterated breadth-first search, and continue with Step 3.
3. Solve the LP-relaxation using only the constraints that correspond to a path in $\mathcal{P}'$ and the inclusion/exclusion constraints from C.
4. Solve the separation problem. Is there a variable (a path) with negative reduced costs?
   YES: Add this variable to $\mathcal{P}'$ and go to Step 3.
   NO: An optimal solution to the LP-relaxation is found with value V_{LP}. Continue with Step 5.
5. $V_{\mathrm{LP}} < V^*$?
   YES: Continue with Step 6.
   NO: Go to Step 2.
6. Is the solution to the LP-relaxation integral?
   YES: $V^* = V_{\mathrm{LP}}$. Go to Step 2.
   NO: Select a vertex v such that y_v is fractional. Create two new branching nodes as follows:
   i. In the first node, add the exclusion constraint $y_v = 0$ to C.
   ii. In the second node, add the inclusion constraint $y_v = 1$ to C.
   Add these two nodes to L.

to the LP-relaxation of (D). In case the resulting solution y is fractional, we branch.

The branching is more straightforward than for the vertex-disjoint paths problem. If there is a vertex v such that $0 < y_v < 1$, we add a constraint $y_v = 0$ (an *exclusion* constraint) in one branch and $y_v = 1$ (an *inclusion* constraint) in the other branch to the linear program and solve it again, thus having two branches for a fractional vertex. If there are many fractional vertices, we simply branch on the first one that we find. Similarly to the previous case, we do not enter a branch where the optimal fractional solution is at least as large as the smallest integral solution found so far. The branch-and-cut algorithm is summarized by the pseudocode given in Figure 5.

We remark that the same approach can be used to solve the generalization of the problem where each vertex v has a weight w_v and the objective is to find a valid s-t cut of minimum weight. The only difference is that the objective function becomes $\min \sum_{v \in V \setminus \{s,t\}} w_v y_v$.

Notice that we use two different approaches for solving the linear programming relaxations of formulations (P) and (D). We found that there were no significant differences in running time between these two approaches.

5. Approximation Algorithms

In this section, we discuss two 2-approximation algorithms for finding the maximum number of vertex-disjoint paths

and the size of minimum cuts. Both algorithms are presented in Erlebach et al. (2005). To make the presentation self-contained, we repeat the description of these algorithms in this section. Section 5.1 deals with the algorithm for the problem of finding the maximum number of vertex-disjoint paths, and in §5.2 we give the algorithm for calculating the size of a minimum cut.

5.1. Vertex-Disjoint Paths

Before stating the approximation algorithm, we need some definitions. If the forward part of a path p_1 intersects a backward part of a path p_2 at a node v, we speak of a crossing at v. The two paths p_1 and p_2 can be recombined at the crossing to form a new path, consisting of the first part of p_1 and the last part of p_2. Given a graph $G = (V, E)$ and two vertices s and t, the algorithm is as follows.

2-Approximation Algorithm VertexDisjointPaths (Erlebach et al. 2005)

1. Construct the 2-layer graph, and calculate a maximum flow in this graph.
2. Add, for each path in this maximum flow, the corresponding path in the original graph G to $\mathcal{P}'$.
3. Define $\mathcal{F}$ as the set of all forward parts of paths in $\mathcal{P}'$, and $\mathcal{B}$ as the set of all backward parts.
4. Label all forward parts and all crossings as unscanned. Recombine the forward and backward parts as follows:
   (a) Select an unscanned forward part p_f from $\mathcal{F}$ that has at least one unscanned crossing.
   (b) Select the first unscanned crossing c on p_f, and let p_b in $\mathcal{B}$ correspond to a backward part containing c.
   (c) Recombine p_f and p_b at c. Label p_f and all previous crossings on p_b as scanned. If p_b was already recombined with some other forward part p'_f, mark p'_f as unscanned.
   (d) Are there any unscanned forward parts with unscanned crossings left?
   YES: Go to Step 4a.
   NO: Stop: a solution is found that is vertex-disjoint.

5.2. Minimum Cut Sizes

We now give the approximation algorithm for finding the minimum cut between two vertices s and t. Assume again that we have a ToR graph $G = (V, E)$ and two vertices $s, t \in V$. We also assume that there is no direct edge in G between s and t because an s-t cut does not exist in that case. The algorithm is then as follows:

2-Approximation Algorithm MinCut (Erlebach et al. 2005)

1. Construct the 2-layer graph, and calculate a minimum cut in this graph.
2. From the cut found in Step 1, construct a cut $\mathcal{C}$ in G as follows: $\mathcal{C}$ contains all vertices $v \in V$ for which at least one copy is in the cut found in Step 1.
3. Stop: $\mathcal{C}$ is a cut in G containing at most twice the number of nodes as in a minimum cut.

6. Computational Experiments

In this section, we first give a description of the data we used for our experiments (§6.1). Next, in §6.2, we discuss some issues concerning the implementation of the algorithms. In §6.3, we give computational results and discuss the performance of the different algorithms. The algorithms described in §§3, 4, and 5 are executed on a number of different ToR graphs, and we compare these results with those in the undirected model (where routing policies that are consequences of established economic relationships are not included) to quantify what the differences are with respect to the size of a minimum cut and the number of disjoint paths. Finally, in §6.4 we focus on the interpretation of the results.

6.1. Description of the Data

We use BGP tables from five different dates (April 2001, February 2002, April 2002, January 2003, and February 2004), available from the University of Oregon Route Views project website (OREGON 2008), to construct undirected graphs and four types of ToR graphs. This means that we have five different graphs for each of the five points in time, giving five undirected graphs and 20 ToR graphs in total. The undirected graphs are obtained by creating an undirected edge between two ASs if they appear consecutively in some path in the BGP tables. We also used one undirected graph model representing the Internet of April 1–16, 2002, which we obtained from CAIDA's Internet Topology Data Kit, ITDK0204 (CAIDA 2008). We refer to this graph as the *CAIDA graph*, to the undirected graphs based on Oregon route views data as *undirected BGP graphs*, and to the graphs that include AS relationships as *ToR graphs*. The types of ToR graphs are denoted by A, B, C, and D as follows:

- ToR graphs of type A are obtained using the algorithm from Erlebach et al. (2002). They contain only customer-provider edges, no peer-to-peer or sibling edges.
- ToR graphs of type B are obtained using the algorithm from Di Battista et al. (2003) by running the software bgpSat, publicly available from their Web page (BGPSAT 2008). A majority of the edges are classified by bgpSat as customer-provider edges, but the classification of some edges is left undetermined. We classify the latter edges as peer-to-peer edges (based on the assumption that the algorithm tries to identify all customer-provider relationships, and peer-to-peer edges are the most common type of relationship different from customer-provider relationships). Thus, type B graphs contain customer-provider edges and a few peer-to-peer edges.
- ToR graphs of type C are obtained from the Web page (CIMVP 2008) and have been produced with the algorithm from Subramanian et al. (2002). The algorithm classifies edges as peer-to-peer edges, customer-provider edges, or unknown edges. We treat the unknown edges as sibling edges (based on the assumption that edges that could not

Table 1. Comparison of edge classifications.

	Percentages of identically classified edges				
ToR graphs	18.04.2001	04.02.2002	06.04.2002	09.01.2003	10.02.2004
A vs. B	95.53	95.41	95.40	95.88	95.08
A vs. C	91.70	91.57	92.21	92.24	91.02
A vs. D	90.96	91.43	91.67	93.16	91.23
B vs. C	89.71	90.30	90.55	90.40	90.28
B vs. D	89.37	90.46	90.59	91.50	90.24
C vs. D	89.60	90.55	90.72	91.35	90.75

be classified as customer-provider or peer-to-peer by the algorithm fall into neither of these categories, and should thus be assigned the third type of relationship).

- ToR graphs of type D are obtained with the algorithm from Gao (2001) (using the implementation (LRIP 2008)) and contain customer-provider edges, peer-to-peer edges, and sibling edges.

All of the inference algorithms that we have used for the construction of ToR graphs are heuristics. Thus, it is interesting to see how many edges between ASs are classified in the same way by the different algorithms; this gives us a first idea of the similarity between the different ToR graphs. In Table 1, the percentages of identically classified edges are given for all six combinations of ToR graphs. For example, from all the edges that are classified in the ToR graphs of types A and B from April 2001, 95.53% are classified in the same way, as is shown in the first entry of the table. From this table, we see that approximately 90% of all edges are classified the same.

Because computing the maximum number of vertex-disjoint paths and the minimum cut size for *all* pairs of ASs would have yielded a huge number of instances (even when omitting vertices of degree 1, the graphs still contain roughly 7,000 to 11,000 vertices), we confine our calculations to approximately 1,000 pairs of ASs per graph. For this reason, we select 47 ASs as representatives and carry out the computations for all possible 1,081 pairs of these ASs. We have selected the ASs by taking 47 vertices among the vertices of largest degree in the biggest R component of the undirected BGP graph of April 2002. (A partition of Internet graphs into P, Q, R, and I components was proposed in Vukadinović et al. (2002). The biggest R component is the biggest connected component in the graph that is obtained after deleting all vertices of degree 1 and their neighbors.) All of the 47 selected ASs are vertices in that component that have at least seven neighbors within that component. Their AS numbers and descriptions are given in Table 2. The reason for choosing ASs with large degree was that the values of both connectivity measures are bounded by the degree of the nodes, and thus ASs with small degree would lead to a much smaller (and less interesting) range of values. Furthermore, many of the links that are missing from the available AS-level

Table 2. The 47 selected ASs with AS numbers and short descriptions obtained from Internet registries.

AS	Description	AS	Description
32	Stanford University	5,413	GX Networks
237	Merit Network Inc. (USA)	5,459	LINX-AS, London Internet Exchange Ltd.
600	OARnet (USA)	6,079	RCN Corporation (USA)
680	DFN-IP service G-WiN	6,402	One Call Communications, Inc. (USA)
1,136	KPN Telecom OVN IO	6,774	BELBONE BELGACOM
1,237	Korea Institute of Science and Technology Information	6,830	UPC Distribution Services European broadband ISP services
2,500	Japan Network Information Center WIDE Project	7,091	ViaNet Communications (USA)
2,514	NTT PC Communications, Inc., Japan	7,623	HPCNET-AS High Performance Computing NETwork (HPCNET) Korea
2,518	C&C Internet Service mesh (NEC Corporation), Japan	7,660	APAN-JP Asia Pacific Advanced Network - Japan
2,647	SITA France	7,679	QTNET Kyushu Telecommunication Network Co., Inc.
2,687	IBM, NH USA	8,426	CLARANET-AS ClaraNET UK AS of European ISP
2,818	BBC Internet Services, UK	8,553	AVENSYS Avensys Networks Ltd UK
3,112	OARnet (USA)	9,270	APAN-KR-AS Asia Pacific Adv. Netw. Korea Consort. Net. Oper. Center
3,304	KPN Belgium	9,335	CIP-JAPAN-AS-AP ATT IPlus Asia and Pacific IP Network
3,333	RIPE NCC Operations	9,497	DIGITELONE Digital Telecommunications Philippines Inc.
3,491	CAIS Internet (USA)	10,099	HKUNICOM1-AP Voice over IP, ISP
3,557	Internet Systems Consortium, Inc. (USA)	10,764	National Center for Supercomputing Applications
3,582	University of Oregon	11,854	Internap Network Services (USA)
3,754	NYSERNet (USA)	12,359	INTELIDEAS Intelideas Autonomous System Madrid, Spain
4,197	ERX-GLOBALONLINE, Japan	12,457	ONO-SERVICE-PROVIDER, Spain
4,635	Hong Kong Internet Exchange–Route Server 1	13,129	Global Access Telecommunications, Inc.
4,725	ODN JAPAN TELECOM CO., LTD.	13,646	Cignal Global Communications, Inc. (USA)
5,000	Internet Online Services (USA)	14,390	Core Communications, Inc. (USA)
5,056	Iowa Network Services (USA)		

topologies due to measurement problems are likely to be peer-to-peer links between smaller ASs, and hence we expect the effect of missing links to be smaller when connectivity measures for ASs of large degree are considered. As one can see, the ASs are geographically well spread—they are from Europe, USA, and Asia. Furthermore, there are representatives of bigger and smaller ISPs, telecom nodes (e.g., Japanese and Belgian telecom), well-connected universities and research centers (e.g., Stanford University, University of Oregon, and National Center for Supercomputing Applications), exchange points (e.g., London and Hongkong Internet Exchange), etc. This means that we have chosen well-connected ASs with diverse functionalities and good geographic coverage while avoiding the highest-degree nodes in the Internet (which are neighbors of leaves), as well as nodes with very small degree.

6.2. Implementation Issues

We have implemented the algorithms in C++ using CPLEX 9.0 to solve linear programs and the LEDA library to process graphs. Our experiments were done on a Sun Fire 480R workstation with two 900 MHz processors (our code uses only one of them) and 4 GB main memory.

For all computations, we have removed vertices with degree 1 because they do not affect the number of disjoint paths or the cut sizes for any other pair of ASs. After pruning the leaf vertices, the graphs contain about 7,000 to 11,000 vertices and 20,000 to 30,000 edges.

For the computation of disjoint paths and cuts, we replaced each peer-to-peer edge $\{u, v\}$ by two edges (u, d) and (v, d), where d is a new dummy vertex. In this way, the valley-free path policy is preserved, while the graph consists of directed edges only.

At the start of the branch-and-price algorithm for computing the maximum number of disjoint s-t paths, we do some additional preprocessing on the graphs. First, we delete all vertices (except s and t) for which the indegree is equal to zero. These vertices can never belong to a valid path, so removing them will not affect the solution we find. Next, we check whether s and t belong to the same biconnected component of the underlying undirected graph. (A biconnected component of an undirected graph G is a subgraph of G such that we can remove any vertex of this subgraph without disconnecting it; see Harary 1969.) If so, we can run the algorithm on this component only (which is usually much smaller than the original graph), and in this way, we still get the optimal solution. If s and t do not belong to the same biconnected component, the number of valid paths between s and t will be either zero or one. Therefore, we check whether there exists a valid path from s to t, in which case the number of vertex-disjoint paths is equal to one. If no valid s-t path exists, our solution is equal to zero. The 47 ASs that we have selected for our experiments were all in the same biconnected component of the undirected BGP graphs, so that the preprocessing alone was never enough to solve the problem. Finally, we found that adding the valid inequalities discussed in §3.1.3 actually slows down the branch-and-price algorithm. In fact, the number of branching nodes needed to solve the problem decreases, as expected, but the time needed to process a single node increases more heavily than the decrease in number of branching nodes, so the computational results presented next are those obtained without using the additional valid inequalities.

For the minimum cut problem, we need to get a well-defined notion of minimum s-t cuts also for adjacent vertices. We handle such vertex pairs as follows: We remove the direct edge (or pair of edges, in the case of a sibling relationship between s and t) between the two vertices s and t, compute the size of a minimum s-t cut in the graph without that edge, and add one to the result. We do this in the undirected graphs as well as in the ToR graphs. Note that in the undirected model, the number of disjoint paths between two vertices is equal to the size of a minimum cut separating these two vertices. In ToR graphs, these values can differ.

6.3. Computational Results

Next, we are interested in the number of disjoint paths and the minimum cut size between pairs of ASs in the different graphs. Section 6.3.1 discusses the performance of the two exact algorithms for solving the vertex-disjoint paths problem. In §6.3.2, we give results for the performance of the algorithm for solving the minimum cut problem, and §6.3.3 deals with the results from the approximation algorithms.

6.3.1. Vertex-Disjoint Paths. We have tested both the branch-and-price and the branch-and-bound algorithm described in §3 to calculate the maximum number of vertex-disjoint paths for any pair of ASs. In Tables 3 and 6, we give the results of these computations.

Table 3 gives the computational results for both algorithms. The first two columns show the graph type and date. The third column contains the value of the integer optimum. Next we give, for both algorithms, four columns containing the computation times (in seconds), the number of branching nodes needed to solve the problems, the percentage of problem instances that are solved in less than one second, and the percentage of instances that are solved in more than 10 seconds. All values in this table are average values over the 1,081 pairs of ASs, so Table 3 contains results of over 20,000 problem instances. Although we could run the branch-and-price algorithm to completion on all pairs in all graphs, we had to terminate the branch-and-bound algorithm on a few pairs (at most 10 out of 1,081 pairs in each of the graphs) after several hours of computation time. The running time and the number of branching nodes shown for the branch-and-bound algorithm in Table 3 are thus the averages over the pairs for which the algorithm could be run to completion.

From Table 3 we conclude that, on the average, both algorithms perform well on the selected pairs of ASs.

Table 3. Results for vertex-disjoint paths problem in ToR graphs.

			Branch-and-price				Branch-and-bound			
Type	Date	OPT	Time	BN	% ⩽1	% >10	Time	BN	% ⩽1	% >10
A	18.04.2001	6.38	0.83	1.96	98.06	0.65	9.44	34.42	94.26	1.11
	04.02.2002	7.88	1.26	2.02	94.36	1.30	2.63	10.84	88.99	2.59
	06.04.2002	8.66	2.61	4.55	93.15	1.85	4.40	16.25	87.60	3.61
	09.01.2003	7.35	0.80	1.58	94.91	0.65	2.28	5.31	94.63	1.85
	10.02.2004	8.10	6.26	6.77	86.12	3.79	15.40	32.86	86.96	4.44
B	18.04.2001	6.42	3.34	7.59	91.77	3.15	2.16	10.45	83.44	3.33
	04.02.2002	8.49	7.61	11.21	81.41	5.46	25.62	71.69	71.14	8.33
	06.04.2002	9.39	10.69	10.94	78.08	7.77	42.63	115.76	68.55	12.86
	09.01.2003	7.52	2.82	5.06	86.40	3.61	11.62	33.28	82.79	3.05
	10.02.2004	8.44	12.12	10.90	79.19	5.64	18.13	40.19	72.34	8.97
C	18.04.2001	6.14	1.25	2.29	94.54	1.30	3.81	11.02	86.12	4.53
	04.02.2002	7.98	2.04	2.76	86.86	2.68	18.50	43.59	69.29	6.94
	06.04.2002	8.46	2.96	3.71	86.77	2.41	4.28	10.68	69.47	3.33
	09.01.2003	6.61	0.88	1.57	93.06	0.83	1.73	4.38	84.55	2.59
	10.02.2004	7.80	1.94	1.97	80.48	2.50	70.27	133.88	71.14	10.73
D	18.04.2001	6.34	2.63	3.06	83.63	4.26	30.43	67.72	71.51	9.44
	04.02.2002	8.01	2.20	2.42	85.38	3.70	47.57	116.08	73.64	9.90
	06.04.2002	8.69	5.96	4.31	81.41	3.98	45.04	88.61	58.19	13.97
	09.01.2003	7.30	1.80	2.20	88.53	2.41	14.20	31.43	79.56	5.00
	10.02.2004	7.92	8.36	4.16	73.64	4.81	59.74	79.00	66.51	16.37

The running times of the branch-and-bound algorithm are much more variable. On 71.78% of all instances, the branch-and-bound algorithm was faster than the branch-and-price algorithm. On the other hand, the branch-and-price algorithm could solve all instances in reasonable time (the average running time over all instances is 3.92 seconds, whereas the instance with the longest running time took slightly more than one hour), whereas the running time of the branch-and-bound algorithm increased drastically for a few instances, thus leading to a larger average running time on most graphs. The number of branching nodes needed to find the integer optimum is much larger for the branch-and-bound algorithm in comparison to the branch-and-price algorithm. For the branch-and-price algorithm, the average number of branching nodes is surprisingly small because in about 89% of the problem instances the solution to the LP-relaxation is integral and we do not need to branch at all.

6.3.2. Minimum Cuts. The algorithm described in §4 to calculate the size of a minimum cut for a pair of ASs has also been executed on the ToR graphs. The results of these computations can be found in Table 4, and we compare these results with those from the undirected models as well (see Table 7).

Table 4 gives the results for the ToR graphs. The first two columns show the graph type and the date, the third column contains the optimal value of the minimum cuts, and, finally, we give the computation times (in seconds), the number of branching nodes needed to find the integer optimum, the percentage of problem instances that are solved in less than one second, and the percentage of instances that are solved in more than 10 seconds. Again, all values are average values over all 1,081 pairs of ASs for a specific graph type and date.

As can be seen from Table 4, the algorithm for finding the minimum cut sizes in ToR graphs is very fast, also compared to the computation times for the algorithms for finding the maximum number of vertex-disjoint paths. Again, the number of branching nodes needed to find the integer

Table 4. Results for the minimum cut problem in ToR graphs.

Type	Date	OPT	Time	BN	% ⩽1	% >10
A	18.04.2001	6.38	0.69	1.03	94.73	0.09
	04.02.2002	7.88	0.95	1.01	77.15	0.00
	06.04.2002	8.66	1.04	1.02	67.07	0.09
	09.01.2003	7.35	1.01	1.01	70.68	0.19
	10.02.2004	8.11	1.90	1.06	49.68	0.74
B	18.04.2001	6.43	0.82	1.11	89.18	0.46
	04.02.2002	8.52	1.86	1.33	59.85	2.41
	06.04.2002	9.42	1.85	1.16	47.92	2.22
	09.01.2003	7.53	1.23	1.03	60.13	0.09
	10.02.2004	8.44	1.83	1.05	41.81	1.11
C	18.04.2001	6.15	1.02	1.06	81.22	0.37
	04.02.2002	7.99	1.43	1.07	60.22	0.46
	06.04.2002	8.47	1.58	1.12	57.45	0.93
	09.01.2003	6.61	1.14	1.02	65.22	0.09
	10.02.2004	7.81	2.15	1.14	34.97	1.39
D	18.04.2001	6.35	1.26	1.17	71.42	0.74
	04.02.2002	8.02	1.34	1.05	63.74	0.28
	06.04.2002	8.70	3.04	1.32	49.58	1.02
	09.01.2003	7.30	1.22	1.01	50.32	0.09
	10.02.2004	7.93	2.08	1.11	27.10	1.76

Table 5. Results for approximation algorithms.

		Disjoint paths			Cut sizes		
Type	Date	OPT	APPROX	Time	OPT	APPROX	Time
A	18.04.2001	6.38	5.32	0.18	6.38	6.40	0.19
	04.02.2002	7.88	6.61	0.23	7.88	8.03	0.24
	06.04.2002	8.66	7.23	0.24	8.66	8.84	0.25
	09.01.2003	7.35	6.36	0.26	7.35	7.41	0.28
	10.02.2004	8.10	6.86	0.32	8.11	8.23	0.34
B	18.04.2001	6.42	5.28	0.18	6.43	6.53	0.19
	04.02.2002	8.49	6.82	0.24	8.52	8.70	0.25
	06.04.2002	9.39	7.45	0.26	9.42	9.57	0.27
	09.01.2003	7.52	6.18	0.28	7.53	7.66	0.30
	10.02.2004	8.44	6.93	0.35	8.44	8.66	0.37
C	18.04.2001	6.14	5.18	0.22	6.15	6.19	0.23
	04.02.2002	7.98	6.70	0.27	7.99	8.14	0.28
	06.04.2002	8.46	7.08	0.28	8.47	8.64	0.29
	09.01.2003	6.61	5.79	0.29	6.61	6.70	0.31
	10.02.2004	7.80	6.71	0.37	7.81	8.01	0.41
D	18.04.2001	6.34	5.19	0.22	6.35	6.49	0.23
	04.02.2002	8.01	6.57	0.27	8.02	8.09	0.27
	06.04.2002	8.69	7.20	0.27	8.70	8.97	0.28
	09.01.2003	7.30	6.24	0.28	7.30	7.37	0.30
	10.02.2004	7.92	6.74	0.35	7.93	8.08	0.38

optimum is small because the solution to the LP-relaxation is integral in 98.5% of the problem instances.

6.3.3. Approximation Algorithms. In Table 5, we give results for the 2-approximation algorithms presented in §5. In the first two columns, we show the graph types and the different dates. Columns 3 to 5 contain information on the number of vertex-disjoint paths, namely, the optimal value, the value found by the approximation algorithm, and the computation times, and in the last three columns we give the same results for the sizes of minimum cuts. Again, all values are average values over all 1,081 problem instances for a specific graph type and date.

From these results, we conclude that both approximation algorithms perform really well. For the problem of finding the maximum number of disjoint paths, we find that in 41.14% of all instances we get an optimal solution, and for 67.63% the difference between the optimal value and the value found by the approximation algorithm is at most one. For the problem of finding the minimum cut sizes, 90.82% of the instances are solved optimally, and in 97.63% of the instances, the difference between the optimum and the value of the approximation algorithm is at most one. Therefore, the heuristic for finding the minimum cut sizes is extremely well suited for this type of instances. The computation times for both algorithms are really fast: All problem instances for both problems are solved within less than one second of computation time. Hence, in cases where the exact optimum is not needed, but one is interested in fast algorithms that produce near-optimal results, the approximation algorithms are a good choice.

6.3.4. Sensitivity to the Number of Edges. Finally, let us consider the sensitivity of our methods to the number of edges in the graphs (for a fixed number of vertices). This is all the more relevant because recent findings suggest that the methods that are used to discover Internet AS-level topologies may miss a significant percentage of the existing links. For example, Cohen and Raz (2006) estimate that more than 50% of the links in the AS-level Internet may be missing from currently available AS-level topologies. Therefore, if more accurate AS-level Internet topologies become available in the future, our algorithms may need to deal with denser graphs than in the experiments reported here. Let us now discuss to what extent the phenomenon of missing edges may influence the outcomes of the computational experiments, or in other words, we comment on the sensitivity of our algorithms to the number of edges. Of course, adding edges to the network yields an increase in the number of variables in (P). However, because we use column generation to solve the LP-relaxation, it follows that the work in a single iteration still amounts to solving a shortest-path problem in a network with the same number of vertices (but containing more edges). Because running times of methods for the shortest-path problem scale linearly with the number of edges, it is to be expected that the work in each iteration remains reasonable. Similarly, the separation problem of (D) is a shortest-path problem and should thus scale linearly with the number of edges. Our branch-and-bound algorithm for (P) solves a network flow problem in each branching node; because there are network flow algorithms that scale linearly with the number of edges, one can expect a very moderate increase of running time on denser graphs. It is difficult to predict how the number of branching nodes will change if more realistic topologies (with missing edges included) are considered. However, many of the missing edges are likely to be peer-to-peer edges between smaller ASs, and such edges can only be used by paths between a limited number of other ASs; therefore, we do not expect that the addition of missing edges would lead to a significant increase in the number of branching nodes.

The main part of the running time of the approximation algorithms is a max-flow or min-cut computation in an ordinary directed graph. Therefore, one can again expect that the increase in running-time for denser graphs is very moderate for these algorithms.

6.4. Interpretation of the Results

In this section, we describe how the results that we obtained can be interpreted. In §6.4.1, we discuss the connectivity of the Internet as measured by the number of disjoint paths and minimum cut sizes. Then, to gain more insight into the AS hierarchy produced by the different inference algorithms, we computed, in all four types of ToR graphs, the fraction of pairs of ASs that are connected with directed customer-provider paths (§6.4.2), as well as the number of edges that are contained in directed customer-provider cycles (§6.4.3).

Table 6. Vertex-disjoint paths in ToR and undirected graphs.

Graph type	Avg. VDP	Min. VDP	Max. VDP
A	7.67	1	55
B	8.05	1	65
C	7.40	0	60
D	7.65	1	48
Undirected BGP	13.46	2	107
CAIDA	12.74	6	108

6.4.1. Connectivity Measures for the Internet. In Table 6, we compare the number of vertex-disjoint paths for the different types of ToR graphs, the undirected BGP graphs, and the CAIDA graph. In the second column, we give the average number of vertex-disjoint paths, averaged over all pairs of ASs and all dates of the specified graph type. The third column gives the minimum number of vertex-disjoint paths found, and the last column shows the maximum number of vertex-disjoint paths (the CAIDA graph is available only for one date, and 3 of our 47 selected ASs are missing from that graph; AS pairs involving a missing AS node were thus ignored for the CAIDA graph).

In Table 7, we compare the size of the minimum cuts in ToR graphs with results from the undirected models, and for all graph types we give the average over all pairs and dates, the minimum value of a minimum cut, and the maximum value. For the undirected BGP graphs and the CAIDA graph, these values are the same as for the vertex-disjoint paths problem because the max-flow min-cut equality holds for the undirected graphs.

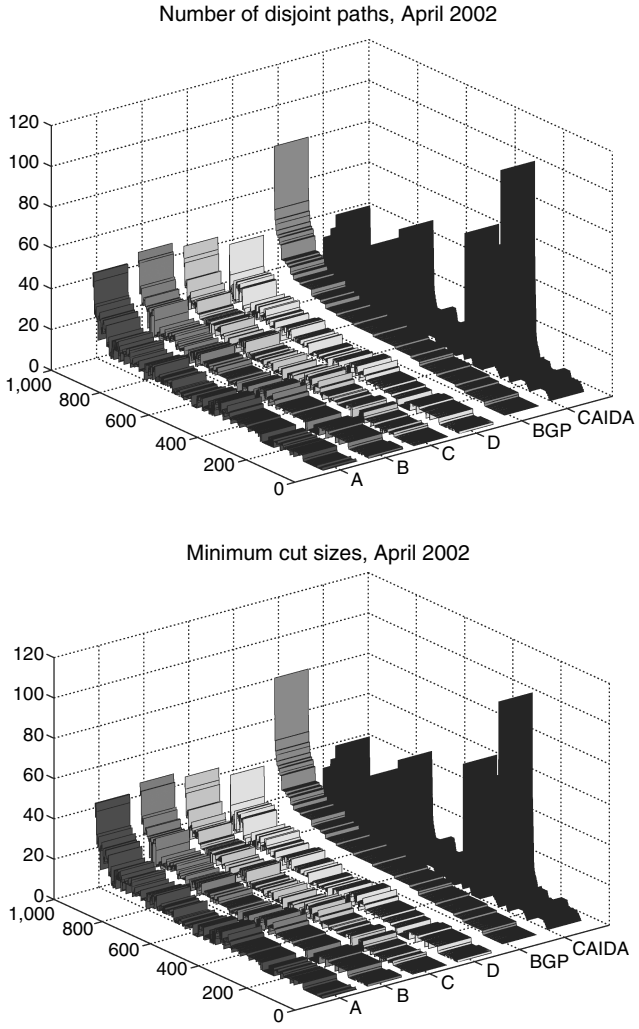
If we compare the connectivity of the ToR graphs with the undirected models, we see a big difference (see Tables 6 and 7). The number of disjoint paths, and the cut sizes, are much larger in the undirected models. For about 72% of all pairs, the number of disjoint paths (and the minimum cut size) is at least 1.5 times bigger in the undirected models, as compared to the ToR graphs, and for approximately 44%, these values in the undirected models are at least twice as large as in the ToR graphs.

When we look at the differences in connectivity between the four different ToR graphs, we see that there is no striking difference between the number of disjoint paths and the sizes of minimum cuts. Generally speaking, graphs of type B have the highest connectivity, and graphs of type C have the lowest connectivity (see the third column of Tables 3 and 4). However, the connectivity of the different ToR graphs seems to be similar.

Table 7. Minimum cut sizes in ToR and undirected graphs.

Graph type	Avg. CS	Min. CS	Max. CS
A	7.68	1	56
B	8.07	1	65
C	7.40	0	60
D	7.66	1	48
Undirected BGP	13.46	2	107
CAIDA	12.74	6	108

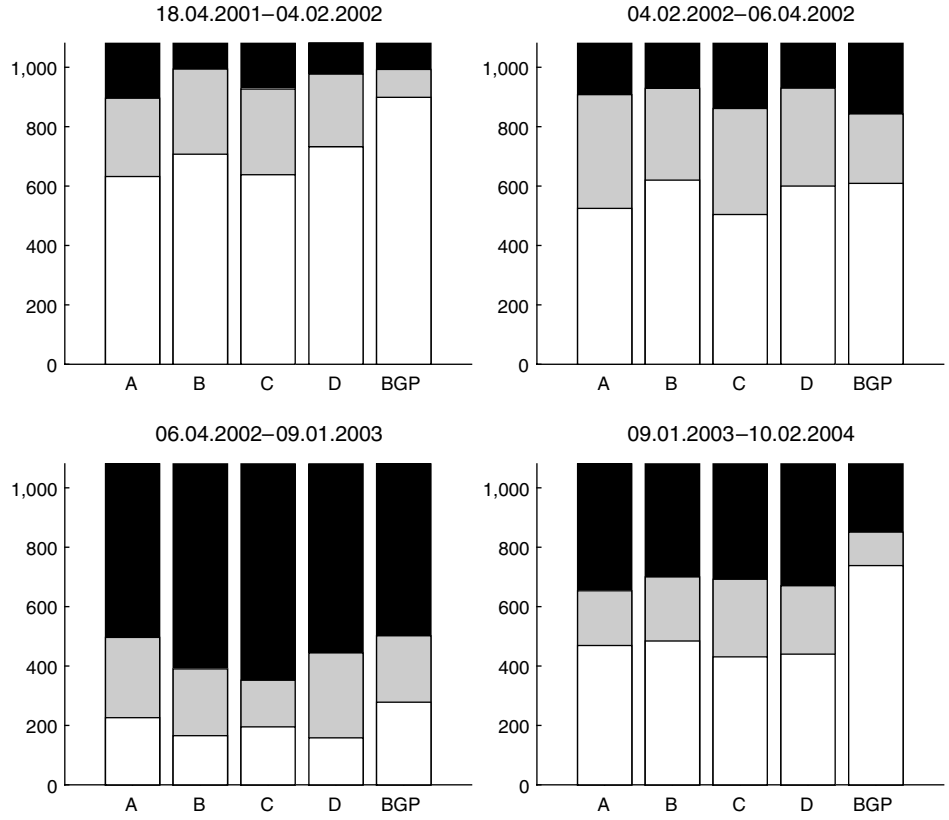
Figure 6. Comparison of ToR and undirected graphs.



In Figure 6, the four types of ToR graphs are represented together with the undirected BGP graph and the CAIDA graph, all graphs taken from April 2002. We obtained similar results for the other four dates, but because we had the CAIDA graph only for April 2002, we chose to use this date for the illustration. The number of disjoint paths and the minimum cut size are shown for each of the 1,081 AS pairs in all six graphs. The values are sorted in order of nondecreasing values in the undirected BGP graph. As the figure shows, there is no striking difference among the ToR graphs. The values for the undirected BGP graph, however, are significantly higher than those for the ToR graphs. This clear difference between the undirected and ToR models confirms that to get an accurate picture of the Internet structure and connectivity, it is important to take routing policies into account.

The values for the CAIDA graph, which has about 6% more edges than the undirected BGP graph, are somewhat

Figure 7. Trends over time for ToR and undirected BGP graphs.



incomparable to those of the undirected BGP graph. For about 35% of the AS pairs, the CAIDA graph has more disjoint paths (up to 100 more paths for one pair), and for about 59% of the pairs, the undirected BGP graph has more disjoint paths (up to 69 more paths for one pair). This indicates that some parts of the Internet are denser (higher number of edges) in the CAIDA graph, whereas other parts are denser in the undirected BGP graph.

Let us now discuss trends over time. The trends for the number of disjoint paths between the different time periods are shown in Figure 7 for each of the four types of ToR graphs and for the undirected BGP graphs. There are four plots, each of them corresponding to a particular time period. In each plot, there is a bar for each of the five graph types. The white part of the bar represents the number of pairs of ASs for which the number of disjoint paths increased in this time period; the shaded part of the bar corresponds to the number of pairs for which the number of disjoint paths stayed the same, and the black part is the number of pairs for which the number of paths decreased in this time period. The results for the minimum cut sizes are similar, so we omit them here.

Figure 7 shows that the ToR graphs behave similarly for all time periods. In the first two time periods, the AS pairs with increasing connectivity form the majority. Then, in the third time period, more than half of the AS pairs display decreasing connectivity. Finally, in the fourth time period, the ToR graphs have roughly the same number of AS pairs with increasing and decreasing connectivity, respectively, whereas about 70% of the AS pairs display increasing connectivity in the undirected BGP graphs.

When we study the differences between the number of disjoint paths and the cut size in ToR graphs, we find that these numbers are equal for about 99% of all AS pairs in each of the ToR graphs (see the third column of Tables 3 and 4). The absolute difference between the minimum cut size and the number of disjoint paths was never larger than two for any of the AS pairs in any of the ToR graphs. Thus, the minimum cut size does not differ significantly from the maximum number of disjoint valid paths in our ToR graphs. Notice that this difference could be as large as a factor of two in general graphs (Erlebach et al. 2005).

Summarizing, the connectivity of the different ToR graphs is similar, and behaves similarly over time; connectivity of undirected graph models overestimates the connectivity: For about half of the pairs of ASs, the connectivity is twice as large when compared to the ToR graphs.

6.4.2. The Depth of the Provider Hierarchy in ToR Graphs. To examine the nature of AS paths in the different ToR graphs, we investigated how many pairs of vertices can be connected by directed paths, i.e., by paths going "only up" or "only down." A path $AS_1, \ldots, AS_k$ between two ASs AS_1 and AS_k such that each AS_i is a customer of AS_{i-1}, for $2 \leqslant i \leqslant k$, is called a *customer chain*. As described in §1.2, this is of interest because routing through

Table 8. Percentage of pairs of ASs connected by customer chains.

Date	A (%)	B (%)	C (%)	D (%)
18.04.2001	10.01	14.93	0.52	2.25
04.02.2002	7.52	14.03	0.56	0.67
06.04.2002	7.02	14.05	0.53	0.59
09.01.2003	6.84	13.62	0.47	0.84
10.02.2004	7.60	14.65	0.53	1.42

a customer brings profit, routing through a peer is neutral, and routing through a provider incurs costs for the sender (see Spring et al. 2003).

In our experiments, we check for all pairs of ASs in ToR graphs (except the pairs involving leaf vertices) whether one of the two ASs is connected to the other via a customer chain. Thus, for such a pair of vertices, it is possible to use a path only through customers (at least in one of the two directions) and thus take advantage of the "customer-preference" policy.

The statistics about customer chains in all four types of ToR graphs are given in Table 8. This table shows for each of the five dates the percentages of pairs of ASs that are connected by customer chains in all our types of ToR graphs.

About 6%–10% of all pairs in type A graphs and 13%–15% of all pairs in type B graphs are connected by customer chains. For graphs of type C and D, the number is significantly smaller, namely, below 1%. This is caused by the fact that ToR graphs of type C and type D contain substantially more edges that are not customer-provider edges. This finding indicates that in graphs of types A and B, the hierarchy seems to be similar and tends to be deep, whereas in graphs of types C and D, the hierarchy seems to be wider because there are many more pairs that are connected only through paths going "up and then down." Concluding, the types of paths that exist between a pair of ASs depend on the inference algorithm used. When using AS-level Internet graphs for retrieving information about the types of paths that exist, and their corresponding profitability, it matters which of the inference algorithms is used.

6.4.3. Directed Customer-Provider Cycles: Detecting Misclassifications. We call a directed cycle (as defined in §2) in a ToR graph a *customer-provider cycle* if it contains only customer-provider edges. If the Internet was a strictly hierarchical network (i.e., if levels can be assigned to the ASs in such a way that in any customer-provider relationship the customer is on a lower level than the provider), one would expect that there are no customer-provider cycles in ToR graphs at all. Therefore, one might use the existence of such cycles as a sign of a misclassification.

We check the existence of such cycles in each of the ToR graphs as follows. First, we remove all sibling edges and peer-to-peer edges from the graph. Then, for each customer-provider edge from AS_i to AS_j, we calculate a shortest directed path (i.e, a path with the smallest number of edges) from AS_j to AS_i. Such a path exists if and only if the edge from AS_i to AS_j is contained in at least one directed cycle. If such a path is found, it gives us a shortest customer-provider cycle containing the edge.

We find that there are no customer-provider cycles in the ToR graphs of type C, except in the graph for 09.01.2003; for the latter date, the type C graph contains a single customer-provider cycle with four nodes (AS11563, AS19035, AS17819, AS1668). In Table 9, we give the results that we obtained for ToR graphs of types A, B, and D. For each of the graphs, we show the total number of customer-provider edges that are contained in cycles, the minimum length of the shortest cycle containing a customer-provider edge, and the maximum length of the shortest cycle containing a customer-provider edge. We find that type B graphs have the largest number of customer-provider cycles, type A graphs have about half as many, and type D graphs have much fewer cycles than both A and B graphs.

Because the ToR graphs of types A and B contain no sibling edges and either no or very few peer-to-peer edges, a larger number of customer-provider cycles could be expected in these graphs. Table 9 confirms that significantly more edges are contained in customer-provider cycles in these graphs. Most of the cycles in the graphs of types A and B are caused by edges classified as customer-provider in A or B graphs, but classified in D graphs as peer-to-peer, sibling, or provider-customer edges.

In the A graph from 18.04.2001, there are 2,571 edges contained in cycles. Each of these edges is contained in a shortest cycle. Among these 2,571 shortest customer-provider cycles (we consider a cycle multiple times if it is the shortest customer-provider cycle of several edges), 1,909 have an edge classified as peer-to-peer in the corresponding D graph, 574 of the remaining ones have an edge classified as sibling edge in the D graph, and 67 of the remaining ones have an edge classified as provider-customer edge in D. Only 21 of the 2,571 cycles are also present in the D graph.

Table 9. Results for directed customer-provider cycles.

Date	Type	Total	Min.	Max.	Type	Total	Min.	Max.	Type	Total	Min.	Max.
18.04.2001	A	2,571	3	9	B	4,046	3	8	D	318	3	20
04.02.2002	A	2,441	3	9	B	4,710	3	8	D	16	3	5
06.04.2002	A	2,278	3	10	B	4,825	3	9	D	9	3	3
09.01.2003	A	2,182	3	10	B	4,858	3	9	D	69	3	11
10.02.2004	A	3,453	3	8	B	6,802	3	9	D	428	3	14

Qualitatively similar results are obtained for all dates for the A and B graphs.

Analyzing the directed cycles in the D graphs, we found that all customer-provider cycles can be eliminated by deleting a very small number of edges (12, 4, 3, 8, and 11 edges, respectively, in the five D graphs from 18.04.2001 to 10.02.2004).

We manually checked 10 edges that were contained in more than 50 discovered cycles (up to 348 cycles) in the D graph from 10.02.2004, using the Nemecis tool (NEMECIS) to access data from Internet Routing Registries. For three of these edges, there was no information in the Internet registries, six of them were classified as peer-to-peer edges (i.e., at least one of the two ASs registered this particular edge as peer-to-peer), and only one edge was registered as customer-provider (confirming its classification in the D graph).

Concluding, directed customer-provider cycles are a good starting point for the detection of misclassifications, in particular if their analysis is combined with a comparison between the different ToR graphs and checking entries in Internet registries. Such cycles can be used to introduce peer-to-peer edges and sibling edges into the ToR graphs of types A and B, which essentially contain only customer-provider edges (in our type B graphs, the only peer-to-peer edges are those that were left unclassified by the algorithm from Di Battista et al. 2003).

7. Conclusions

We have studied the maximum number of disjoint valid paths and the minimum cut size for selected AS pairs. Because both problems are $\mathcal{NP}$-hard, we have designed and implemented several algorithms based on branch-and-bound or branch-and-price paradigms. We have shown that the algorithms make it possible to compute optimal values for these connectivity measures on real AS-level Internet graphs. For the problem of finding the maximum number of disjoint paths between any pair of ASs, we have implemented two exact algorithms, the first one being a branch-and-price algorithm based on an integer programming formulation of the problem, and the second one being a branch-and-bound algorithm in which we perform a max-flow calculation in each node of the search tree. From the results, we conclude that the latter algorithm is often faster than the first, but may require excessive computation times on certain inputs, whereas the computation times for the branch-and-price algorithm are always acceptable and do not display such a variability. For the problem of finding the minimum cut sizes, we have implemented a branch-and-cut algorithm that performs really well, with average computation times around one to two seconds for instances with up to 11,000 nodes and 30,000 edges (after pruning nodes of degree one).

The results of these algorithms allow us to quantify the differences in connectivity between ToR graphs and the traditional undirected model of the Internet, which ignores routing policies. We find that about 44% of the selected AS pairs have more than twice as many disjoint paths in the undirected model than in the ToR graphs, which confirms that the use of ToR graphs is crucial for Internet analysis and simulations that are sensitive to connectivity properties, e.g., in studies concerning topological robustness, multipath routing, etc. We have also investigated the increase of connectivity over time and found that the number of disjoint paths between ASs seems to grow for fewer AS pairs in the ToR graphs than in the undirected graph model.

Using our new algorithms, we have compared different types of graphs with inferred AS relationships (ToR graphs) regarding connectivity measures as well as other path characteristics. Comparing the ToR graphs with each other, we find that on the average they do not differ much with respect to the number of disjoint paths and the minimum cut sizes between AS pairs. On the other hand, concerning the hierarchy (observed indirectly by counting the number of AS pairs connected through customer chains), it turns out that A and B graphs are relatively similar to each other, but different from C and D graphs—their hierarchy appears to be deeper than that of C and D graphs. In addition, we find that the investigation of short directed customer-provider cycles in the ToR graphs can help to detect misclassifications and may lead to new approaches for introducing peer-to-peer or sibling relationships into A and B graphs, which can make these models more realistic.

While our investigations provide some insight into the properties of the ToR graphs produced by the different available inference algorithms, it is not possible for us to identify one of these algorithms as better than the others. Researchers who employ ToR graphs in their research should be aware of the differences in the ToR models produced by different algorithms and make sure that their conclusions are not biased by the choice of ToR graph. Our findings can help in making informed decisions about the choice of a ToR graph model: If only connectivity-related aspects of the ToR graph are important, all four types of ToR graphs are similarly well suited. However, if the presence of customer-provider cycles or the number of peer-to-peer edges is important, one needs to be aware of the fact that different ToR graphs are very different with respect to these characteristics.

Furthermore, our approach of adapting the classical connectivity measures, maximum number of disjoint paths, and minimum cut size to valley-free paths in ToR graphs can be useful in further research on robustness issues in the Internet. Besides, it may be possible to adapt our branch-and-price approach to incorporate other types of constraints on valid paths, thus allowing the analysis of connectivity properties of other networks with special routing constraints as well.

The known algorithms for inferring AS relationships (see Gao 2001, Subramanian et al. 2002, Di Battista et al. 2003, and Erlebach et al. 2002) all need data from BGP routing tables as input. Because the data from BGP routing tables

is not always complete or accurate (the impact of this is demonstrated convincingly by the huge difference in the number of disjoint paths for certain AS pairs in the undirected BGP graph and the CAIDA graph; see Figure 6), it would be an interesting question whether BGP routing tables are a necessary input of any algorithm inferring AS relationships. An inference algorithm not based on BGP routing tables (but, for example, based on (artificial) AS-properties) could be used in synthetic graph models obtained from Internet topology generators. A different approach in the latter direction has been explored in Chang et al. (2003b), where a new optimization-driven model for Internet growth is presented that allows the generation of synthetic AS graphs containing only customer-provider relationships.

Finally, let us summarize our findings by answering the research question posed in §1.

- We obtain optimal solutions to the problems of finding the maximum number of vertex-disjoint paths and minimum cut sizes, using the new exact algorithms proposed in this paper. The algorithms all require a small amount of time to find these optimal values.
- The performance of the approximation algorithms is reasonable. Especially for the problem of finding minimum cut sizes, the approximation algorithm performs really well. Both these algorithms are much faster than the exact algorithms.
- The impact of BGP policies on Internet routing has been observed in a number of other contexts, and our results confirm the importance of considering ToR graphs instead of undirected graph models when studying robustness issues of the AS-level Internet. More importantly, our results allow us to quantify the difference in connectivity of undirected and ToR graph models.
- The different heuristics (see Gao 2001, Subramanian et al. 2002, Di Battista et al. 2003, and Erlebach et al. 2002) used for constructing a topology do not differ much with respect to the connectivity measures. On the average, they all have a similar number of disjoint paths and minimum cut size between pairs of ASs. The constructed topologies do, however, differ with respect to the types of paths that exist between a given pair of ASs. Graphs of types C and D have many fewer customer chains than graphs of types A and B. This is relevant because different types of paths may yield different profits for an individual AS. Finally, because graphs of types A and B do not have sibling edges, and no or very few peer-to-peer edges, a relatively large number of customer-provider cycles exist in these graphs. Directed customer-provider cycles in ToR graphs can be useful for the detection of misclassified edges.

Acknowledgments

This research was partly done while the first author was with the Computer Engineering and Networks Laboratory at ETH Zürich. The fourth author was supported in DICS-Project 1838, Robustness of the Internet at the Topology and Routing Level, by the Hasler Foundation. The authors thank the referees for their comments, which improved the presentation of the paper, and they thank Maciej Kurant for pointing out an error in the original version of the manuscript.

Data for CAIDA (2008) was collected as part of CAIDA's Skitter initiative. Support for Skitter is provided by DARPA, NSF, and CAIDA sponsorship.

References

Ahuja, A., T. L. Magnanti, J. B. Orlin. 1993. *Network Flows*: *Theory, Algorithms, and Applications*. Prentice-Hall, Englewood Cliffs, NJ.

Akella, A., S. Chawla, A. Kannan, S. Seshan. 2003a. Scaling properties of the Internet graph. *ACM PODC 2003*. ACM, New York, 337–346.

Akella, A., B. Maggs, S. Seshan, A. Shaikh, R. Sitaraman. 2003b. A measurement-based analysis of multihoming. *SIGCOMM 2003*. ACM, New York, 353–364.

Barnhart, C., E. Johnson, G. Nemhauser, M. Savelsbergh, P. Vance. 1998. Branch-and-price: Column generation for solving huge integer programs. *Oper. Res.* **46**(3) 316–329.

BGPSAT. 2008. Project Web page of G. Di Battista, M. Patrignani, and M. Pizzonia, Computing the types of the relationships between autonomous systems. Accessed June 2008, http://www.dia.uniroma3.it/compunet/relationships/.

Caesar, M., J. Rexford. 2005. BGP routing policies in ISP networks. *IEEE Network Magazine* **19**(special issue on interdomain routing) 5–11.

CAIDA. 2008. The Cooperative Association for Internet Data Analysis. Internet Topology Data Kit 0204. Accessed June 2008, http://www.caida.org/tools/measurement/skitter/.

Chang, H., S. Jamin, W. Willinger. 2003a. Internet connectivity at the AS-level: An optimization-driven modeling approach. *SIGCOMM '03*. ACM, New York, 33–46

Chang, H., S. Jamin, W. Willinger. 2003b. What causal forces shape Internet connectivity at the AS-level? Technical Report CSE-475–03, Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor.

Chang, H., S. Jamin, W. Willinger. 2006. To peer or not to peer: Modeling the evolution of the Internet's AS-level topology. *Proc. INFOCOM '06*. IEEE, Piscataway, NJ, 1–12.

Chang, H., S. Jamin, Z. M. Mao, W. Willinger. 2005. An empirical approach to modeling inter-AS traffic matrices. *Proc. Internet Measurement Conf.* USENIX Association, Berkeley, CA, 139–152.

Chang, H., R. Govidan, S. Jamin, S. J. Shenker, W. Willinger. 2004. Towards capturing representative AS-level Internet topologies. *Comput. Networks J.* **44**(6) 737–755.

CIMVP. 2008. Project Web page of S. Agarwal, L. Subramanian, J. Rexford, and R. H. Katz. Characterizing the Internet hierarchy from multiple vantage points. Accessed June 2008, http://www.cs.berkeley.edu/~sagarwal/research/BGP-hierarchy/.

Cohen, R., D. Raz. 2006. The Internet dark matter—On the missing links in the AS connectivity map. *Proc. INFOCOM '06*. IEEE, Piscataway, NJ, 1–12.

Di Battista, G., M. Patrignani, M. Pizzonia. 2003. Computing the types of the relationships between autonomous systems. *Proc. INFOCOM '03*. IEEE, Piscataway, NJ, 156-165.

Erlebach, T., A. Hall, T. Schank. 2002. Classifying customer-provider relationships in the Internet. *Proc. IASTED Internat. Conf. Comm. Comput. Networks*, Cambridge, UK, 538–545.

Erlebach, T., A. Hall, A. Panconesi, D. Vukadinović. 2005. Cuts and disjoint paths in the valley-free path model of Internet BGP routing. *Proc. First Workshop on Combinatorial and Algorithmic Aspects of Networking* (*CAAN 2004*). LNCS 3405, Springer, Berlin, 49–62.

Feigenbaum, J., C. Papadimitriou, R. Sami, S. Shenker. 2002. A BGP-based mechanism for lowest-cost routing. *Proc. 21st Sympos. Principles of Distributed Comput.* ACM, New York, 173–182.

Gao, L. 2001. On inferring autonomous system relationships in the Internet. *IEEE/ACM Trans. Networking* **9**(6) 733–745.

Gao, L., F. Wang. 2002. The extent of AS path inflation by routing policies. *Proc. IEEE Global Internet Sympos. 2002*. IEEE, Piscataway, NJ, 2180–2184.

Grötschel, M., L. Lovász, A. Schrijver. 1988. *Geometric Algorithms and Combinatorial Optimization*. Springer, Berlin.

Harary, F. 1969. *Graph Theory*. Addison-Wesley, Cambridge, MA.

Huston, G. 1999a. Interconnection, peering and settlements—Part I. *Internet Protocol J.* **2**(1) 2–16.

Huston, G. 1999b. Interconnection, peering and settlements—Part II. *Internet Protocol J.* **2**(2) 2–23.

Labovitz, C., A. Ahuja, R. Wattenhofer, S. Venkatachary. 2001. The impact of Internet policy and topology on delayed routing convergence. *Proc. INFOCOM '01*. IEEE, Piscataway, NJ, 537–546.

LRIP. 2008. Project Web page of D. R. Figueiredo, Z. Ge, and S. Jaiswal. Logical relationship inference program (implementation of algorithms from Gao 2001). Accessed June 2008, http://www-net.cs.umass.edu/∼ratton/AS/.

Mahadevan, P., D. Krioukov, M. Fomenkov, B. Huffaker, X. Dimitropoulos, kc claffy, A. Vahdat. 2005. Lessons from three views of the Internet topology. Technical report TR-2005–02, CAIDA.

NEMECIS. 2008. University of California, Riverside. Accessed June 2008, http://ira.cs.ucr.edu:8080/Nemecis.

OREGON. 2008. Route views project website. University of Oregon. Accessed June 2008, http://www.routeviews.org.

Quoitin, B., C. Pelsser, L. Swinnen, O. Bonaventure, S. Uhlig. 2003. Interdomain traffic engineering with BGP. *IEEE Comm. Magazine* **41**(May) 122–128.

Rimondini, M., M. Pizzonia, G. Di Battista, M. Patrignani. 2004. Algorithms for the inference of the commercial relationships between autonomous systems: Results analysis and model validation. *Proc. IPS 2004, Internat. Workshop on Inter-Domain Performance and Simulation*, 33–45.

Spring, N., R. Mahajan, T. Anderson. 2003. Quantifying the causes of path inflation. *SIGCOMM '03*. ACM, New York, 113–124.

Subramanian, L., S. Agarwal, J. Rexford, R. Katz. 2002. Characterising the Internet hierarchy from multiple vantage points. *Proc. INFOCOM '02*. IEEE, Piscataway, NJ, 618–627.

Tangmunarunkit, H., R. Govidan, S. Shenker. 2001a. Internet path inflation due to policy routing. *Proc. SPIE ITcom '01*, Vol. 4520. SPIE, 188

Tangmunarunkit, H., R. Govidan, S. Shenker. 2003. Internet topology: Discovery and policy impact. K. Park, W. Willinger, eds. *The Internet As a Large-Scale Complex System*. Oxford University Press, Oxford, UK, 121–160.

Tangmunarunkit, H., R. Govidan, S. Shenker, D. Estrin. 2001b. The impact of routing policy on Internet paths. *Proc. INFOCOM '01*. IEEE, Piscataway, NJ, 736–742.

Teixeira, R., K. Marzullo, S. Savage, G. Voelker. 2003. Characterizing and measuring path diversity of Internet topologies. *Proc. SIGMETRICS '03*. ACM, New York, 304–305.

Vanderbeck, F., L. A. Wolsey. 1996. An exact algorithm for IP column generation. *Oper. Res. Lett.* **19** 151–159.

Vukadinović, D., P. Huang, T. Erlebach. 2002. On the spectrum and structure of Internet topology graphs. *Innovative Internet Computing Systems*. No. 2346, *Lecture Notes in Computer Science*. Springer, Berlin, 83–95.

Wang, H., Y. R. Yang, P. H. Liu, J. Wang, A. Gerber, A. Greenberg. 2007. Reliability as an interdomain service. *Proc. ACM SIGCOMM 2007, Kyoto, Japan*. ACM, New York, 229–240.

Wang, X., D. Loguinov. 2006. Wealth-based evolution model for the Internet AS-level topology. *Proc. INFOCOM '06*. IEEE, Piscataway, NJ, 1–11.

Wu, J., Y. Zhang, Z. M. Mao, K. Shin. 2007. Internet routing resilience to failure: Analysis and implications. *Proc. CoNext 2007*. ACM, New York.

Xia, J., L. Gao. 2004. On the evaluation of AS relationship inferences. *Proc. IEEE Global Commun. Conf.* (*GLOBECOM 2004*). IEEE, Piscataway, NJ, 1373–1377.