Acyclic Type-of-Relationship Problems on the Internet

Sven Kosub Moritz G. Maaß Hanjo Täubig

TUM-I0605
März 06

Acyclic Type-of-Relationship Problems on the Internet

Sven Kosub[*] *Moritz G. Maaß*[†] *Hanjo Täubig*[‡]
Fakultät für Informatik, Technische Universität München,
Boltzmannstraße 3, D-85748 Garching, Germany
{kosub,maass,taeubig}@in.tum.de

Abstract

We contribute to the study of inferring commercial relationships between autonomous systems (AS relationships) from observable BGP routes. We deduce several forbidden patterns of AS relationships that impose a certain type of acyclicity on the AS graph. We investigate algorithms for solving the *acyclic all-paths type-of-relationship* problem, i.e., given a set of AS paths, find an orientation of the edges according to some types of AS relationships such that the oriented AS graph is acyclic (with respect to the forbidden patterns) and all AS paths are valley-free. As possible AS relationships we include customer-to-provider, peer-to-peer, and sibling-to-sibling. Moreover, we examine a number of problem versions parameterized by sets K and U where K is set of edge types available for describing explicite pre-knowledge and U is set of edge types available for completion of partial orientations. A complete complexity classification of all 56 cases (8 type sets for pre-knowledge and 7 type sets for completion) is given. The most relevant practical result is a linear-time algorithm for finding an acyclic and valley-free completion using customer-to-provider relations given *any* kind of pre-knowledge. Interestingly, if we allow sibling-to-sibling relations for completions then most of the non-trivial inference problems become NP-hard.

1 Introduction

Numerous studies (e.g., [17, 21, 16, 20, 28, 22, 11] to name only few) have exposed that, in order to understand the dynamics of Internet inter-domain routing, it is not sufficient to possess deeper knowledge of the AS graph, i.e., the physical-connection topology among autonomous systems (ASs). Rather, most importantly, local routing policies of independent administrative domains and their interplay based on business relations have a critical influence on route (in)stability and routing quality. As business contracts are intentionally considered to be commercial secrets, many researchers have proposed methods to elicite this crucial information indirectly (see, e.g., [12, 26, 5, 9, 6, 30, 7]).

[*]Supported by Minerva Foundation of Max-Planck-Gesellschaft. Work done in part while visiting The Hebrew University of Jerusalem.

[†]Supported by DFG, grant Ma 870/5-1 (Leibnizpreis Ernst W. Mayr).

[‡]Supported by DFG, grant Ma 870/5-1 (Leibnizpreis Ernst W. Mayr).

In [12], the seminal work on inferring contractual relationships on basis of sets of AS paths observable from BGP updates, heuristic approaches were devised to classify relationships into customer-to-provider, peer-to-peer, and sibling-to-sibling. A key observation is that, under rational economic behavior, AS paths exhibit a regular, so-called valley-free structure, i.e., after traversing a provider-to-customer or peer-to-peer edge, the AS path cannot traverse a customer-to-provider or peer-to-peer edge. Thus, each path has some top provider. To identify top providers in AS paths, heuristics follow basically two evident assumptions (see, e.g., [15, 12, 27]):

A Providers are much larger than customers in terms of infrastructure.

B The size of an AS is proportional to its degree in the AS graph.

In principle the algorithms in [12] iteratively search for vertices with maximum degree in a given AS path set to identify top providers. This approach has been further elaborated in [26, 30]. Degree-based heuristics are certainly practicable if we are able to get a representative sample of all AS paths of the Internet (see [3] for a critical discussion). But, due to their over-sensitivity to path sets, they have weaknesses in well-defined analytical test situations (such as, e.g., in [11, 13]).

A purely combinatorial treatment, neither involving **A** nor **B**, was done in [5, 9, 6]. There, the authors described a linear-time algorithm solving the *all-paths type-of-relationship* problem, suggested in [26]: given a path set P, is there an orientation of the edges (indicating provider-to-customer or customer-to-provider relation) such that all paths of P are valley-free. The algorithm is based on a reduction to 2-SAT which is well-known to be solvable in linear time. In contrast, finding an orientation maximizing the number of valley-free paths is not polynomial-time approximable within factor $O(n^{1-\varepsilon})$ (unless NP = ZPP). Positively, for path lengths at most ℓ, there is a polynomial-time algorithm with approximation ratio $\frac{\ell+1}{2^\ell}$ (sometimes better by semi-definite programming for Max 2-SAT). Though computationally elegant and robust, these algorithms often lead to unrealistic relationships (i.e., well-known global providers appearing as customers of small ASs [7]).

To get closer to reality, several proposals have been made. One of them calls for partialness-to-entireness algorithms [30, 5]. The basic idea is to infer the entire AS relationships from partial information obtainable from data sources other than BGP paths (see, e.g., [25, 30]). Another proposal links degree-based heuristics and combinatorial optimization by considering weighted Max 2-SAT with weights depending on degree gradients [7]. Positive experimental results suggest the fruitfulness of incorporating degree information into a combinatorial setting. Nevertheless, over-sensitivity to path sets when using assumption **B** is still problematic.

Our contribution also lies in the middle of the heuristic and the combinatorial approaches to the inference problem. Instead of using both assumptions **A** and **B**, we will only employ assumption **A** to avoid path-set over-sensitivity. The size rule **A** (and some similar one's) bears enough information to impose a global structure on oriented AS graphs. In Section 3, we deduce several unrealistic relationship patterns which we will forbid to appear in AS graph orientations. A fundamental pattern of this type is an oriented cycle within the graph, as this would imply that some provider is its own customer. Acyclic orientations are constructed by the degree-based heuristic approach of [12, 26]. In contrast, the 2-SAT-based algorithm generally constructs cycles due to the internal computation of strongly connected components. As an example, Figure 1 shows a path set for which all valley-free orientations impose a cycle

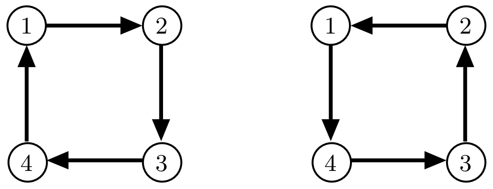


Figure 1: All valley-free orientations for path set $\{(1,2,3),(2,3,4),(3,4,1),(4,1,2)\}$.

on the AS graph. However, as a prerequisite for studies related to the Internet topology (see, e.g., [13, 8]), acyclicity is a requirement for realistic AS relationships.

In this paper, we focus on algorithms solving the *acyclic all-paths type-of-relationship* problem (Acyclic ToR), i.e., given a path set, find an orientation of the edges according to some types of AS relationships such that the oriented AS graph is acyclic (with respect to our forbidden patterns) and all AS paths are valley-free. As possible AS relationships we include customer-to-provider, peer-to-peer, and sibling-to-sibling. Furthermore, we meet the partialness-to-entireness requirement. We examine a number of versions Acyclic ToR (K,U) where K is the set of edge types available for describing explicite pre-knowledge and U is the set of edge types available for completion of partial orientations. In Section 4, we give a complete complexity classification of all 56 cases (8 type sets for pre-knowledge and 7 type sets for completion). The most relevant practical result is a linear-time algorithm for finding an acyclic and valley-free completion using customer-to-provider relations given *any* kind of pre-knowledge. Interestingly, if we allow sibling-to-sibling relations for completions, then most of the non-trivial inference problems become NP-hard.

2 An Abstract Model of BGP

We briefly describe a simple, abstract model of inter-domain routing in the Internet using BGP (see, e.g, [29, 24, 17, 12]).

2.1 The Selective Export Rule

The elementary entities in our Internet world are IP adresses, i.e., bit strings of prescribed length. An autonomous system (AS) is a connected group of one or more IP prefixes (i.e., blocks of contiguous IP adresses) run by one or more network operators which has a single and clearly defined routing policy [18]. An AS aims at providing global reachability for its IP adresses. To achieve this goal, ASs having common physical connections exchange routing informations as governed by their own local routing policies. BGP is the *de facto* standard protocol to manage data traffic between ASs for inter-domain routing as well as for route propagation.

Reachability in the Internet depends on (physical) connectivity and contractual relationships between ASs. The most fundamental binary business relationships are customer-to-provider (where the provider sells routes to the customer), peer-to-peer (where the involved ASs provide special routes to their customers but no transit for each other), and sibling-to-sibling (where both ASs belong to the same administrative domain). More peculiar relationships appear in the

AS v exports to	provider	customer	peer	sibling
own routes	Yes	Yes	Yes	Yes
customer routes	Yes	Yes	Yes	Yes
provider routes	No	Yes	No	Yes
peer routes	No	Yes	No	Yes

Figure 2: The Selective Export Rule.

real world (see, e.g., [12]). We restrict ourselves to the three mentioned types of relationships.

More specifically, let V be a set of AS numbers. For any $v \in V$, let $N(v) \subseteq V$ denote the set of its neighbor ASs, i.e., all numbers of ASs sharing a physical connection with v. The undirected graph $G = (V, E)$ where $E = \{ \{u, v\} \mid v \in N(u) \}$ is called a *connectivity graph* (at the AS level) or simply *AS graph.* Let $v \in V$ be any AS. According to the business relationship we divide the neighbors of v into the sets $\mathrm{Cust}(v)$ of all customers of v, $\mathrm{Prov}(v)$ of all providers of v, $\mathrm{Sibl}(v)$ of all siblings of v, and $\mathrm{Peer}(v)$ of all peering partners of v. Some of the sets may be empty. We let $\mathrm{Sibl}(v)$ contain v as well. Let $R(v)$ denote the set of all currently *active* AS paths in the BGP routing table of v, i.e., all AS paths that have been announced from neighboring ASs at a certain time and never been withdrawn. Assumed that there are no misconfigurations of BGP, all AS paths in $R(v)$ are loopless and not including v. Here, we say that an AS path is *loopless* whenever between two sibling ASs on the path, no non-sibling AS is passed. Based on the neighborhood classification, we further divide $R(v)$ into four categories. A loopless AS path $(u_1, \ldots, u_r) \in R(v)$ is

$$\begin{aligned}
\text{a } \textit{customer route} \text{ of } v &\iff_{\text{def}} \text{leftmost } u_i \notin \mathrm{Sibl}(v) \text{ lies in } \mathrm{Cust}(v),\\
\text{a } \textit{provider route} \text{ of } v &\iff_{\text{def}} \text{leftmost } u_i \notin \mathrm{Sibl}(v) \text{ lies in } \mathrm{Prov}(v),\\
\text{a } \textit{peer route} \text{ of } v &\iff_{\text{def}} \text{leftmost } u_i \notin \mathrm{Sibl}(v) \text{ lies in } \mathrm{Peer}(v),\\
\text{an } \textit{own route} \text{ of } v &\iff_{\text{def}} \text{for all } 1 \le i \le r,\ u_i \in \mathrm{Sibl}(v).
\end{aligned}$$

Now, typically (at least, recommendably), ASs set up their export policies according to the Selective Export Rule [1, 19, 12] as described in Figure 2. In our simplified model, the receiving AS gets from an AS those (locally preferred) routes destined for it and prolongated with the number of the sending AS as the new leftmost AS number in the path.

2.2 The Valley-Free Path Model

Valley-freeness is a graph-theoretical consequence of the Selective Export Rule. Let $G = (V, E)$ be an undirected (simple) graph. We assume that $(u, v) \in E \Leftrightarrow (v, u) \in E$. A (mixed) *orientation* φ *of* G is a mapping from E to T where T denotes set of possible edge-types. For instance, a directed graph is a graph oriented with type set $T = \{\leftarrow, \rightarrow\}$. We consider type sets having the following edge-types and interpretations:

$\rightarrow$	indicating a customer-to-provider relationship
$\leftarrow$	indicating a provider-to-customer relationship
—	indicating a peer-to-peer relationship
$\leftrightarrow$	indicating a sibling-to-sibling relationship

Throughout this paper, we only consider orientations φ that are consistent with respect to $\to$. That is, for all $(u,v) \in E$ if $\varphi(u,v) = \leftarrow$ then $\varphi(v,u) = \rightarrow$ and if $\varphi(u,v) = \rightarrow$ then $\varphi(v,u) = \leftarrow$. Thus, if we allow $\rightarrow$ as a possible edge type, then we immediately allow $\leftarrow$ as a possible edge type as well.

We extend φ from edges to walks homomorphically. Let $(v_0, v_1, \ldots, v_m)$ be any walk in a graph G. Then $\varphi(v_0, v_1, \ldots, v_m)$ is defined to be $\varphi(v_0,v_1)\varphi(v_1,v_2)\ldots\varphi(v_{m-1},v_m)$, i.e., in our setting generally, a word in $\{\leftarrow, \rightarrow, —, \leftrightarrow\}^*$. We will typically use regular expressions to describe walk types given an orientation. An important property of orientations is valley-freeness, which we state here in terms of regular patterns of paths.

Definition 1. [12] *Let G be any graph, and let $\varphi(G)$ be an orientation of G. A loopless path $(v_0, \ldots, v_m)$ is said to be* valley-free *in $\varphi(G)$ if and only if $\varphi(v_0, \ldots, v_m)$ belongs to*

$$\{\rightarrow, \leftrightarrow\}^*\{\leftarrow, \leftrightarrow\}^* \quad \cup \quad \{\rightarrow, \leftrightarrow\}^* — \{\leftarrow, \leftrightarrow\}^*.$$

The valley-freeness of paths abstracts the condition that autonomous systems never route data from one of their providers to another of their providers because they instead of earning money, they would have to pay twice for these data streams.

Theorem 2. [12] *Let $G = (V,E)$ be an AS graph. Let P be any subset of AS paths of all BGP routing tables, i.e., $P \subseteq \bigcup_{v \in V} R(v)$. If all ASs export their routes according to the Selective Export Rule, then there exists an orientation of P such that all AS paths in P are valley-free.*

3 Acyclicity Conditions

In the previous section, we have seen how in our simple, abstract BGP some rational economic behavior implies valley-freeness of locally observable routes. These routes reflect short-term behavior determined by routing policies based on commercial relationships. Commercial relationships typically are stable over a longer period and they impose a global structure on the connectivity graph independent of concrete BGP routes.

In this section, we summarize common knowledge on business relations between ASs to obtain a reasonable acyclicity structure within a connectivity graph. We do so by identifying patterns of oriented cycles which we will forbid to be contained in the graph. An oriented cycle can be interpreted as someone being its own provider and customer.

In Figures 3 and 4, the 16 non-isomorphic triads of the 64 possible orientations of a complete graph on three vertices are shown. Figure 3 lists 8 forbidden triads together with plausibility arguments why they are forbidden. The generalizations of the forbidden patterns are fairly obvious. Plausibility is based on size rules:

1. If AS u is a customer of AS v, then AS u has much smaller size (i.e., number of routers) than AS v (see, e.g., [15, 12, 26]). This is assumption **A** from the introductory section.

2. If AS u is a peering partner of AS v, then AS u and AS v are roughly of the same size (see, e.g., [23]). Moreover, we consider *roughly the same size* to be a transitive relation.

3. If AS u and AS v are siblings, then they count as one AS, i.e., we assume that the size of u is determined by its own number of routers and the number of routers of all its sibling ASs.

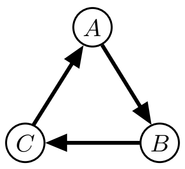
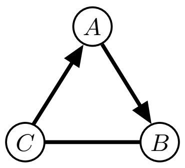
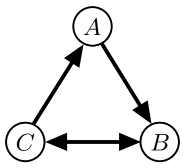
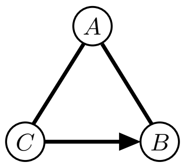
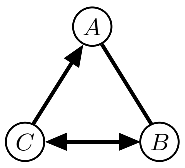
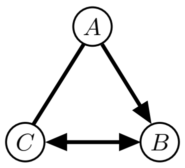
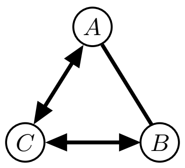
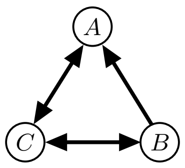
	Typically, C is provider of B only if C is much larger than B, A is provider of C only if A is much larger than C. So, B is not much larger than A. A contradiction to B being provider of A.	$\rightarrow^*$
	A typical criterion for a peer-to-peer relation is roughly the same size or traffic. This does not hold if A is much larger than C and B is much larger than A.	$\{\rightarrow, \text{—}\}^* \rightarrow \{\rightarrow, \text{—}\}^*$
	Typically, as B and C are siblings, they behave like one AS. So, B and C together are not much larger and much smaller than A at the same time.	$\{\rightarrow, \leftrightarrow\}^* \rightarrow \{\rightarrow, \leftrightarrow\}^*$
	Typically, C and A have a peer-to-peer relation if they are roughly the same size. The same holds for A and B. So, B should not be much larger than C.	$\{\rightarrow, \text{—}\}^* \rightarrow \{\rightarrow, \text{—}\}^*$
	Typically, A and B have a peer-to-peer relation if A and B together with its sibling C have roughly the same size. So, A is not much larger than C together with its sibling B.	$\{\rightarrow, \leftrightarrow, \text{—}\}^* \rightarrow \{\rightarrow, \leftrightarrow, \text{—}\}^*$
	Typically, A and C have a peer-to-peer relation if A and C together with its sibling B have roughly the same size. So, A is not much smaller than B together with its sibling C.	$\{\rightarrow, \leftrightarrow, \text{—}\}^* \rightarrow \{\rightarrow, \leftrightarrow, \text{—}\}^*$
	Due to the transitivity of the sibling-to-sibling relation, A and B are siblings. So, typically, A and B do not have a peculiar peer-to-peer relation. Note that larger cycles with siblings and at least two peer-to-peer relations make sense.	$\leftrightarrow^* \text{—} \leftrightarrow^*$
	Due to the transitivity of the sibling-to-sibling relation, A and B are siblings. So, A together with its siblings B and C is not much larger than C together with its siblings B and A.	$\{\rightarrow, \leftrightarrow\}^* \rightarrow \{\rightarrow, \leftrightarrow\}^*$

Figure 3: Forbidden triads and their generalized forbidden patterns.

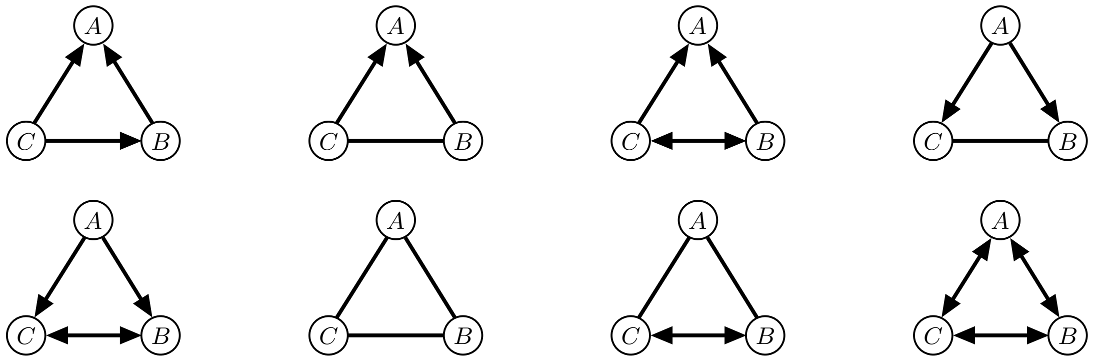


Figure 4: Allowed triads.

Each of these rules may have its exceptions but they certainly describe typical behavior. We idealize the BGP world in assuming that all contracts follow these rules.

The union of all generalized forbidden patterns given in Figure 3 leads to the following definition of an oriented cycle.

Definition 3. *Let G be any graph, and let $\varphi(G)$ be an orientation of G. Let C be any minimal cycle of G. C is said to be an* oriented cycle of $\varphi(G)$ *if and only if $\varphi(C)$ belongs to*

$$\{—, \leftrightarrow\}^* \rightarrow \{\rightarrow, —, \leftrightarrow\}^* \quad \cup \quad \{—, \leftrightarrow\}^* \leftarrow \{\leftarrow, —, \leftrightarrow\}^* \quad \cup \quad \leftrightarrow^* — \leftrightarrow^* .$$

The minimality of cycles is required since we exactly count occurrences of peer-to-peer edges in oriented cycles. To complement our view on forbidden triads, Figure 4 shows the 8 allowed triads.

Note that in the case that φ does not exhaust the full type set $\{\rightarrow, —, \leftrightarrow\}$, the patterns of oriented cycles simplifiy. For instance, if the type set is $\{\rightarrow\}$, then we obtain that a minimal cycle C is an oriented cycle if and only if $\varphi(C)$ belongs to $\rightarrow^*$ or $\leftarrow^*$ which is the usual understanding of a cycle. As a second example, if the type set is $\{\leftrightarrow, —\}$, then a minimal cycle C is an oriented cycle if and only if $\varphi(C)$ belongs to $\leftrightarrow^* — \leftrightarrow^*$.

We call an orientation *acyclic* if it contains no oriented cycles. In the forthcoming we will need fast algorithms for testing acyclicity which are all based on standard techniques (see, e.g., [4]).

Lemma 4. *Let K be any subset of $\{\rightarrow, —, \leftrightarrow\}$. Testing whether a given graph, with n vertices and m edges, which is oriented with type set K is acyclic can be done in time $O(n+m)$.*

Proof. We briefly describe algorithms for all cases individually.

1. The cases where the type set is either $\{—\}$ or $\{\leftrightarrow\}$ are trivial, as graphs oriented in these ways are always acyclic.

2. Acyclicity for type set $\{\rightarrow\}$ can be tested by topological sort.

3. For type set $\{\rightarrow, \leftrightarrow\}$, replace each sibling-to-sibling edge $u \leftrightarrow v$ with two edges $u \rightarrow v$ and $v \rightarrow u$. Compute the strongly connected components and test whether each component only contains sibling-to-sibling edges in the original orientation. This can be done in time $O(n+m)$.

4. Similarly, for type set $\{\rightarrow, —\}$, replace a peer-to-peer edge $u—v$ with two edges $u \rightarrow v$ and $v \rightarrow u$. Compute the strongly connected components and test whether each component only contains peer-to-peer edges in the original orientation. Again, this can be done in time $O(n+m)$.

5. For $\{—, \leftrightarrow\}$, compute the simply connected components with respect to the $\leftrightarrow$ edges and check for each oriented edge $u —v$ whether u und v belong to different components. If for some edge this is not true, we have learnt that the graph is not acyclic. This can be done in time $O(n+m)$.

6. Given the full type set $\{\rightarrow, —, \leftrightarrow\}$, we first test acyclicity of the oriented graph with type set $\{—, \leftrightarrow\}$ induced by ignoring customer-to-provider edges. If the graph passes the test, then we replace in the original orientation $\leftrightarrow$ and — edges with directed edges of opposite directions, compute the strongly connected components, and check for each $u \rightarrow v$ or $u —v$ whether u and v belong to the different components. If for some edge this is not true, the graph is not acyclic. Clearly, this can be done in time $O(n+m)$.

This completes the proof of the lemma. □

4 Combinatorial Inference of Relationships

Our central problem is: given any set of BGP routes, is there an orientation such that all BGP paths are indeed valley-free and the induced connectivity graph has the additional property of being acyclic? We bring this issue into a very general, purely combinatorial formulation. Let K and U be two type sets.

Problem:	Acyclic ToR (K, U)
Input:	Undirected graph G, path set P, partial orientation φ given by an edge set R with labels from K
Output:	A completion of the orientation φ using only edge types from U such that the completed orientation, with respect to type set $K \cup U$, contains no cycle and all oriented paths in P are valley-free, or indicate that such a completion does not exist

Interpretation. Applied to a real-world scenario, G is the AS graph, P stands for the set of observed BGP routes, e.g., gathered at certain observation points. The set R is an explicite pre-knowledge we have of certain relations between two ASs, as observable from several resources on the Internet. The task is to find a hypothetical valley-free and acyclic orientation in accordance with our pre-knowledge. In a test setting, G is a BGP-world model, P could be the set of information made available to BGP speakers, and R could describe a specified situation with types from K. Here, we want to find an orientation that guarantees valley-freeness and acyclicity and which does not destroy the given specification. For this purpose, an appropriate choice of a type set U is useful. The most important case, of course, is $U = \{\rightarrow\}$.

Input representation. Let N always denote the input size, i.e., $N = \|V(G)\| + \|E(G)\| + |P| + |R|$ where $|P|$ and $|R|$ are the sums of the path lengths in P and in R. The length of a path with

k edges is the sum of the lengths of the $k+1$ vertex descriptions. Since many of our algorithms run in linear time, we should say a word about the representation of instances. For the sake of simplicity we assume that all vertices and edges of G actually appear in P, i.e., $G = G(P) = (V(P), E(P)$ where $V(P)$ denotes the set of vertices appearing in P and $E(P)$ denotes the set of edges appearing in P. Thus, $N = O(|P|)$. We use $V(p)$ and $E(p)$ to denote $V(\{p\})$ and $E(\{p\})$. We suppose that an instance is given as an adjacency list of an AS graph $G(P)$ with vertex set $V(P)$ and edge set $E(P)$, where edges from R are already labeled according to the partial orientation. Moreover, we assume that the path set is represented as a collection of lists having cross-links to the corresponding edges in the AS graph and vice versa. Orientations are stored with the edges in $G(P)$. Since there are no more vertices and edges in $G(P)$ than appearing in P, this guarantees that we can always test valley-freeness in time $O(N)$.

Technical remarks. The complexity of the problem depends on the types of allowed orientations. The basic (and typically considered) combinatorial problem appears as case $K = \emptyset$, i.e., where we have no pre-knowledge. Without going into formal details, we easily see that if $K \subseteq K'$, then Acyclic ToR (K, U) is computationally not harder than Acyclic ToR (K', U), i.e., an algorithmic upper bound for Acyclic ToR (K', U) is an upper bound for Acyclic ToR (K, U) and an algorithmic lower bound for Acyclic ToR (K, U) is an algorithmic lower bound for Acyclic ToR (K', U). We say that Acyclic ToR (K, U) is NP-hard if it is NP-hard to decide whether for a given instance of Acyclic ToR (K, U), there is an acyclic and valley-free orientation in the sense of the problem definition. Evidently, if Acyclic ToR (K, U) is NP-hard, then for all $K \subseteq K'$, Acyclic ToR (K', U) is NP-hard.

In the forthcoming we will prove the complexity classification shown in Figure 5. From Lemma 4 we immediately obtain the entries for the second and third column of our classification.

Proposition 5. *Let $K \subseteq \{\rightarrow, \leftrightarrow, -\}$ be any type set. Then,* Acyclic ToR $(K, \{-\})$ *and* Acyclic ToR $(K, \{\leftrightarrow\})$ *can be solved in time $O(N)$.*

To further reduce the number of results to be proven, in the next subsection we will see that peer-to-peer relations can essentially be disregarded as options for completions. Subsections 4.2–4.4 then contain the remaining algorithms and hardness results.

4.1 Handling Peer-to-Peer Relations

To handle peer-to-peer edges in both type sets K and U, we first show that in an acyclic and valley-free orientation they exhibit a very simple structure, namely they induce a partial ordering among vertex sets. To see this, it is useful to introduce some further notation.

Suppose we are given an orientation φ of a graph G using the type set $\{\rightarrow, -\}$. We say that a vertex v is *P2P-reachable* from vertex u in G if there exists an oriented path p from u to v in G only consisting of peer-to-peer edges. Notice that P2P-reachability is a reflexive, symmetric, and transitive relation; hence, an equivalence relation. The vertex set thus can be divided into equivalence classes which we call *P2P-components*.

Let $p = (w_0, \ldots, w_k)$ be a path of any path set P. We define a set-mapping $\xi[p]$ which for any vertex set $A \subseteq V(P)$ collects all positions in p that lie between two positions of vertices of

	U						
K	$\rightarrow$	—	$\leftrightarrow$	$\rightarrow$, —	$\rightarrow, \leftrightarrow$	—, $\leftrightarrow$	$\rightarrow$, —, $\leftrightarrow$
$\emptyset$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$
$\rightarrow$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	NP-hard	$O(N)$	NP-hard
—	$O(N)$	$O(N)$	$O(N)$	$O(N)$	NP-hard	NP-hard	NP-hard
$\leftrightarrow$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	$O(N)$
$\rightarrow$, —	$O(N)$	$O(N)$	$O(N)$	$O(N)$	NP-hard	NP-hard	NP-hard
$\rightarrow, \leftrightarrow$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	NP-hard	$O(N)$	NP-hard
—, $\leftrightarrow$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	NP-hard	NP-hard	NP-hard
$\rightarrow$, —, $\leftrightarrow$	$O(N)$	$O(N)$	$O(N)$	$O(N)$	NP-hard	NP-hard	NP-hard

Figure 5: Complexity classification of Acyclic ToR (K, U) for $K, U \subseteq \{\rightarrow, —, \leftrightarrow\}$.

A. More specifically, we define $\xi[p]$ for $A \subseteq V(P)$ as

$$\xi[p](A) =_{\text{def}} \{\ i \mid \text{ there are } \ell \le i \le r \text{ such that } [w_\ell \in A \wedge w_r \in A]\ \}.$$

Note that if $A \cap V(p) \neq \emptyset$, then $\min \xi[p](A)$ and $\max \xi[p](A)$ are positions where vertices from A occur.

For any valley-free orientation φ of a graph $G(P)$ for a path set P (we also say orientation of P), the turning point of a path $p = (w_0, \ldots, w_k) \in P$ is the maximal position $0 \le i \le k$ such that $\varphi(w_{i-1}, w_i) =\rightarrow$. If the orientation is additionally acyclic, then we can isolate the range of positions within paths where turning points occur. Observe that generally, in order to have an acyclic and valley-free orientation, in each path that contains vertices s and t P2P-reachable from each other and that are not neighbors, the orientation has to point from s in direction of t and from t in direction of s. Otherwise we would find an oriented cycle. An easy consequence is the following proposition.

Proposition 6. *Let P be a path set. Suppose we are given an acyclic and valley-free orientation of P using $\{\rightarrow, —\}$. Let x and y be vertices in the same P2P-component. For each path $p \in P$ containing both vertices x and y, the turning point of p belongs to $\xi[p](\{x, y\})$.*

The following lemma shows that in an acyclic and valley-free orientation, all pairs of vertices from the same P2P-component form a total ordering within each path.

Lemma 7. *Let P be a path set. Suppose we are given an acyclic and valley-free orientation using $\{\rightarrow, —\}$. Let $\{u, v\}$ and $\{x, y\}$ be two pairs of vertices such that $u \neq v$, $x \neq y$, $\{u, v\} \neq \{x, y\}$, v is P2P-reachable from u, and y is P2P-reachable from x. For all paths $p \in P$, if $\{u, v, x, y\} \subseteq V(p)$, then (1) $\{u, v\} \cap \{x, y\} = \emptyset$ and (2) $\xi[p](\{u, v\}) \subseteq \xi[p](\{x, y\})$ or $\xi[p](\{x, y\}) \subseteq \xi[p](\{u, v\})$.*

Proof. Suppose we have an acyclic and valley-free orientation of P using $\{—, \rightarrow\}$. First, let $p \in P$ be any path which does not contain a peer-to-peer edge (in particular, neither u and v are neighbors in p nor are x and y) and let both pairs $\{u, v\}$ and $\{x, y\}$ be contained in p.

From the observation above that lead us to Proposition 6, we obtain that the turning point belongs to $\xi[p](\{u,v\}) \cap \xi[p](\{x,y\})$. This excludes the case that $\xi[p](\{u,v\})$ and $\xi[p](\{x,y\})$ have at most one common element. Without loss of generality, assume that $\min \xi[p](\{u,v\}) \leq \min \xi[p](\{x,y\})$. Consider the vertex sequence

$$\left(w_{\min \xi[p](\{u,v\})}, \dots, w_{\min \xi[p](\{x,y\})}, w_{\max \xi[p](\{x,y\})}, \dots, w_{\max \xi[p](\{u,v\})}, w_{\min \xi[p](\{u,v\})}\right).$$

For this sequence not to be an oriented cycle in $G(P)$, it must hold that $\min \xi[p](\{u,v\}) < \min \xi[p](\{x,y\})$ and $\max \xi[p](\{x,y\}) < \max \xi[p](\{u,v\})$. Recall that the vertices at positions $\min \xi[p](\{u,v\})$ and $\max \xi[p](\{u,v\})$ can be connected in $G(P)$ by a path totally consisting of peer-to-peer edges. Equally, $\min \xi[p](\{x,y\})$ and $\max \xi[p](\{x,y\})$ can be bridged by such a path. The case of p having a peer-to-peer edge can be treated in a similar way. The statement of the lemma follows. □

Note that Lemma 7 also shows that in acyclic and valley-free orientations, paths contain at most two vertices from an equivalence class according to P2P-reachability.

In addition to Lemma 7, P2P-components show on all paths the same total ordering.

Lemma 8. *Let P be a path set. Suppose we are given an acyclic and valley-free orientation using $\{\rightarrow, —\}$. Let A and B be two distinct P2P-components. If there is a path $p \in P$ such that $\|A \cap V(p)\| \geq 2$, $\|B \cap V(p)\| \geq 2$ and $\xi[p](A) \subseteq \xi[p](B)$, then for each path $q \in P$ satisfying $\|A \cap V(q)\| \geq 2$ and $\|B \cap V(q)\| \geq 2$, it holds that $\xi[q](A) \subseteq \xi[q](B)$.*

Proof. Let P be a path set which allows an acyclic and valley-free orientation φ of $G(P)$ using $\{\rightarrow, —\}$. Let A and B be two equivalence classes according to P2P-reachability. Let $p \in P$ be a path which contains some distinct vertices $x_A, y_A \in A$ and some distinct vertices $x_B, y_B \in B$ such that $\xi[p](\{x_A, y_A\}) \subseteq \xi[p](\{x_B, y_B\})$. We may assume that x_A occurs before y_A in p and x_B occurs before y_B in q. Assume to the contrary that there is a path $q \in P$ containing distinct vertices $u_A, v_A \in A$ and $u_B, v_B \in B$ such that $\xi[q](\{u_A, v_A\}) \not\subseteq \xi[q](\{u_B, v_B\})$. From Lemma 7, we obtain $\xi[q](\{u_B, v_B\}) \subseteq \xi[q](\{u_A, v_A\})$. Here, we also may assume that u_B occurs before v_B in q and u_A occurs before v_A in q. By Proposition 6, the turning position of p belongs $\xi[p](A)$ and the turning position of q belongs to $\xi[q](B)$. It follows that $G(P)$ contains an oriented cycle that can be built by a path from u_A to u_B belonging to $\rightarrow^*$ (as in q), followed by a path from u_B to x_B belonging to $—^*$, followed by a path from x_B to x_A belonging to $\rightarrow^*$ (as in p), and followed by a path back from x_A to u_A belonging to $—^*$. A contradiction. □

We use the lemma to define an auxiliary directed graph $H(\varphi, P)$, depending on the path set P and an acyclic and valley-free orientation φ of P with type set $\{\rightarrow, —\}$, as follows:

$$\begin{aligned} V(H(\varphi,P)) &=_{\text{def}} \{\ A \mid A \subseteq V(P) \text{ is a P2P-component (with respect to } \varphi) \ \} \\ E(H(\varphi,P)) &=_{\text{def}} \{\ (A,B) \mid (\exists p \in P)[\ \xi[p](A) \subset \xi[p](B)\ ]\ \} \end{aligned}$$

By Lemma 8, this graph is always well-defined. Moreover, the graph is a directed acyclic graph since it describes a partial ordering of the vertex subsets.

We will use this auxiliary graph to see that under certain circumstances we can disregard peer-to-peer edges for finding acyclic and valley-free completions. Let P be a path set and let φ a partial orientation of P using —. We say that a P2P-component A is *φ-complete* if and only if for all edges $\{u, v\} \in E(P)$, if both u and v belong to A then $\varphi(u, v) = —$. Notice that the edge $\{u, v\}$ cannot be oriented as a customer-to-provider edge if we want to guarantee acyclicity.

Lemma 9. *Let P be an arbitrary path set. Let φ be a partial orientation of P with type set $\{\rightarrow, —\}$ such that all P2P-components are φ-complete. There is an acyclic and valley-free completion of φ on P with type set $\{\rightarrow, —\}$ if and only if there is an acyclic and valley-free completion of φ on P with type set $\{\rightarrow\}$.*

Proof. Direction $(\Leftarrow)$ holds trivially.

For $(\Rightarrow)$, suppose $\hat{\varphi}$ is an acyclic and valley-free completion of φ on P with type set $\{\rightarrow, —\}$. We show that each peer-to-peer edge introduced by the completion can be changed into a customer-to-provider or a provider-to-customer edge without introducing oriented cycles while keeping paths valley-free. Let us consider the auxiliary graph $H(\hat{\varphi}, P)$. Suppose the vertex set $\{A_1, \ldots, A_r\}$ of $H(\hat{\varphi}, P)$ is topologically sorted, i.e., if (A_i, A_j) is an edge in $H(\hat{\varphi}, P)$ then $i < j$. We define an orientation ψ, for all $u, v \in V(P)$, as follows:

$$\psi(u, v) =_{\text{def}} \begin{cases} \hat{\varphi}(u, v) & \text{if } \hat{\varphi}(u, v) \neq — \text{ or both } u \text{ and } v \text{ lie in one P2P-component} \\ \rightarrow & \text{if } \hat{\varphi}(u, v) = — \text{ and } u \in A_i, v \in A_j \text{ with } i < j \\ \leftarrow & \text{if } \hat{\varphi}(u, v) = — \text{ and } u \in A_i, v \in A_j \text{ with } j < i \end{cases}$$

Since all P2P-components are φ-complete, ψ is a total function and thus, ψ is completion of φ. We have to prove the ψ is acyclic and valley-free. We examine both criteria separately.

1. *Valley-freeness.* Let e be an edge in the graph $G(P)$ such that $\hat{\varphi}(e) = —$. Since $\hat{\varphi}$ is a valley-free orientation, any path p of P that contains e must have an orientation of type $\rightarrow^* — \leftarrow^*$ or $\rightarrow^* — \leftarrow^*$. Orienting e either as $\rightarrow$ or as $\leftarrow$ satisfies valley-freeness. So does ψ.

2. *Acyclicity.* Let $e = \{u, v\}$ be any edge of $G(P)$ such that $\hat{\varphi}(e) = —$ and the vertices u, v do not belong to the same P2P-component (with respect to φ). Let C be any cycle containing e as its last edge. Recall from Definition 3 that an oriented cycle using type set $\{\rightarrow, —\}$ belongs to

$$—^* \rightarrow \{\rightarrow, —\}^* \ \cup \ —^* \leftarrow \{\leftarrow, —\}^*.$$

As $\hat{\varphi}(C)$ is an acyclic orientation, i.e., $\hat{\varphi}(C)$ does not show the just-mentioned pattern, we have the following three cases to consider:

   (a) $\hat{\varphi}(C)$ contains an edge oriented as $\rightarrow$ before coming an edge oriented by $\leftarrow$ and ending with the edge e oriented as $—$. Clearly, orienting e in the way ψ does does not introduce a cycle.

   (b) $\hat{\varphi}(C)$ contains an edge oriented as $\leftarrow$ before coming an edge oriented by $\rightarrow$ and ending with the edge e oriented as $—$. This is similar to the case above. Thus, orienting e in the way ψ does does not introduce a cycle.

   (c) $\hat{\varphi}(C)$ completely consists of peer-to-peer edges. Note that a cycle has at least three edges. Since u and v belong to different P2P-components there must exist further edges in C with vertices from different P2P-components. Since the P2P-components are totally ordered by natural numbers, ψ does not introduce a cycle.

This completes the lemma. ☐

Lemma 9 is a twofold generalization of a proposition in [5] which states that for any path set P, there exists a valley-free orientation with type set $\{\rightarrow, —\}$ if and only if there exists a valley-free using only $\{\rightarrow\}$. First, it proves the same property for additionally acyclic orientations. Second, it proves this property independently of the initially given partial orientation. Note that in respect thereof, the standard case of [5] appears as the nowhere-defined partial orientation.

4.2 Completing with Customer-to-Provider Relations

In this subsection we will see how we can compute, in linear time, an acyclic and valley-free orientation no matter of which kind of information our pre-knowledge is. That is, the goal is to show that Acyclic ToR $(\{\rightarrow, \leftrightarrow, —\}, \{\rightarrow\})$ is solvable in linear time (see Theorem 16). For the sake of clarity, it is reasonable to approach this theorem over several intermediate stages.

We start with the very base case $K = \emptyset$. For obtaining a linear time algorithm, the crucial observation is that each vertex appearing somewhere in the middle of a path has in-degree at least one, valley-freeness supposed. This allows us to employ a topological-sort approach.

Theorem 10. Acyclic ToR $(\emptyset, \{\rightarrow\})$ *can be solved in time* $O(N)$.

Proof. Suppose we are given a path set P. Let v be a vertex such that, for each path in P, if v lies on p then v is an endpoint of the path. If $G(P)$ can be acyclically oriented such that all paths are oriented valley-free then such a vertex v must exists: for any acyclic orientation of $G(P)$ there must exist at least one vertex u such that all edges $\{u, w\}$ are oriented as $u \rightarrow w$. Since all paths are valley-free, u cannot be in the middle of any path because that would result in an orientation containing $\leftarrow\rightarrow$. Therefore, one vertex v can be forced.

We iteratively reduce the problem by removing such vertices v for all path-ends and orienting the removed edges away from v as $v \rightarrow w$ for all neigbors w in P. (Note that P has changed.) Assume a reduced path $(v_0, \dots, v_m)$ is oriented as $\rightarrow^*\leftarrow^*$, then adding v to any end with the edge u, v_1 or u, v_m oriented as $\rightarrow$ results in a valley-free orientation. Furthermore, if the reduced graph is acyclic then the graph with v added is also acyclic.

A precise description is given as Algorithm 1. Clearly, this algorithm can be implemented in such a way that its running time is $O(|P|)$. ☐

The basic algorithm 1 described in Theorem 10 can be extended by additional linear-time preprocessing phases to handle non-trivial pre-knowledge.

Theorem 11. Acyclic ToR $(\{\rightarrow\}, \{\rightarrow\})$ *can be solved in time* $O(N)$.

Proof. We modify Algorithm 1 appropriately. Let $\perp$ be a vertex which is neither in path set P nor in edge set R. For each oriented edge $u \rightarrow v$ of R, add a path $(u, v, \perp)$ to the path set. Let P' be the resulting path set. Now apply Algorithm 1 on path set P' where $\perp$ is not considered as a vertex of $V(P')$. Clearly, this algorithm solves the problem correctly (by an argumentation similar to Theorem 10) and runs in time $O(N)$. ☐

For the case $K = \{\leftrightarrow\}$ we need some more notation. Let P be any given path set and let φ be a partial orientation of $G(P)$ given by the set R of sibling-to-sibling edges in $E(P)$. We define a vertex set $U \subseteq V(P)$ to be an *S2S-component* if for all vertices $u, v \in U$, there exists

Algorithm 1: Linear-time algorithm for Acyclic ToR $(\emptyset, \{\rightarrow\})$.

Input: Undirected graph G, path set P

Output: Acyclic and valley-free orientation of the induced graph $G(P)$, if it exists, or indication that it does not exist

```
foreach vertex $v \in V(P)$ do
    count($v$) := 0
end
foreach $p \in P$ do
    foreach vertex $v \in p$ do
        if $v$ is not an endnode of $p$ then
            count($v$) := count($v$) + 1
        end
    end
end
$U := \emptyset$
foreach vertex $v \in V(P)$ do
    if count($v$) = 0 then
        $U := U \cup \{v\}$
    end
end
$V' := \emptyset$
while $U \neq \emptyset$ do
    remove a vertex $u$ from $U$
    foreach vertex $v$ such that $v \in V \setminus V'$ and $\{u, v\} \in E(P)$ do
        orient $\{u, v\}$ as $u \rightarrow v$
        foreach $p \in P$ such that $u$ and $v$ are neighbors in $p$ do
            if $v$ has a neighbor $w$ in $p$ on the side opposite to $u$ such that $w \in V \setminus V'$ then
                count($v$) := count($v$) − 1
            end
            if count($v$) = 0 then
                $U := U \cup \{v\}$
            end
        end
    end
    $V' := V' \cup \{u\}$
end
if $V' \neq V$ then
    return path set $P$ fails to allow an acyclic and valley-free orientation
end
```

a path in $G(P)$ from u to v consisting only of sibling-to-sibling edges. Clearly, for any given set R the vertex set $V(P)$ can be divided into S2S-component which can be easily computed in linear time. We say that an S2S-component U is *S2S-complete* if for all $u, v \in U$, the edge $\{u, v\}$ belongs to R, i.e., $\varphi(u, v)$ is defined and $\varphi(u, v) = \leftrightarrow$. We say that an S2S-component U is *S2S-loopless* if for each path $p \in P$, there does not occur any vertex not belonging to U between two vertices from U on the path p. An S2S-component is *S2S-closed* if it is both S2S-loopless and S2S-complete. Obviously, completeness, looplessness, and closeness can easily be tested in linear time.

Theorem 12. Acyclic ToR $(\{\leftrightarrow\}, \{\rightarrow\})$ *can be solved in time* $O(N)$.

Proof. As we are not allowed to introduce new sibling-to-sibling edges, we only have to check whether the set R of sibling-to-sibling edges induces an S2S-closed vertex set (meaning the set of vertices $V(R)$ occurring in R). If not reject the instance. Otherwise replace all occurrences of vertices from an S2S-component A with a representative vertex and remove consecutive occurrences of same vertices in all paths. Now we have an Acyclic ToR $(\emptyset, \{\rightarrow\})$ instance which can be solved in time $O(N)$ by Theorem 10. Note that computing the reduced instance only needs $O(N)$ preprocessing time. □

Corollary 13. Acyclic ToR $(\{\rightarrow, \leftrightarrow\}, \{\rightarrow\})$ *can be solved in time* $O(N)$.

Proof. Observe that the preprocessing phase of the algorithm described in the proof of Theorem 12 does not dependent on edges oriented with types other than $\{\leftrightarrow\}$. Thus, after this $O(N)$ preprocessing step we have to solve an Acyclic ToR $(\{\rightarrow\}, \{\rightarrow\})$ instance which is just Theorem 11. □

The case $K = \{—\}$ is more complex. Recall from Lemma 7 that in acyclic and valley-free orientations, pairs of P2P-reachable vertices build an inclusion chain within each path.

Theorem 14. Acyclic ToR $(\{—\}, \{\rightarrow\})$ *can be solved in time* $O(N)$.

Proof. For a given instance (G, P, R), let $S \subseteq P$ be the set of paths containing a peer-to-peer edge from R. Let $G_R(P)$ denote the graph induced by P which only contains the peer-to-peer edges from R. The algorithm has two phases.

In the first, structure-testing phase of the algorithm we do the following:

1. Check for each path of S whether there is at most one peer-to-peer edge and, if there is one, orient all edges towards that edge. If the check fails then return an indication and stop.

2. Next check for each path $p \in P \setminus S$ whether for all pairs $\{u, v\} \subseteq V(p)$ such that v is P2P-reachable from u in $G_R(P)$, the sets $\xi[p](u, v)$ (introduced in the preceding subsection) form an inclusion chain according to Lemma 7 and compute the vertex pair e^p such that $\xi[p](e^p)$ is minimal with respect to set-inclusion. Normally, this would imply a worst-case time $O(|P \setminus S|^2 \cdot |R|) = O(N^3)$ (with a linear-time preprocessing step to compute the connected components in $G_R(P)$). However, we can do better to stay within $O(N)$. We simply compute some pair of vertices with minimal distance of their positions in p (using buckets for each P2P-component) and take this pair as e^p. We postpone the test of the chain-condition until we have to test for acyclicity and valley-freeness anyway.

3. For each path $p \in P \setminus S$ and its corresponding unique minimal pair $e^p = \{u, v\}$, if it exists, orient both the edge going from u in direction of v and the edge going from v in direction of u as out-going edges. Orient all edges outside of $\xi[p](e^p)$ towards e^p.

4. Now test whether the graph induced by the already oriented edges is acyclic. If not reject the instance. Here, we would also identify cases where the above-mentioned chain-condition is not satisfied. Otherwise proceed as follows. Eliminate from each path $p \in P \setminus S$ all oriented edges. The remaining edges thus have indices in $\xi[p](e^p)$ not including e^p's vertices. Notice that until we have arrived at this point, by Lemma 7, we had no freedom of choice in orienting the edges as we did.

Let P' denote the remaining path set and let X denote the set of additionally oriented customer-to-provider edges. Note that $S \subseteq P'$. Since replacing all peer-to-peer edges with sibling-to-sibling edges does not destroy an acyclic and valley-free orientation, in the second phase of the algorithm we solve $(G, P' \setminus S, R \cup X)$ as an Acyclic ToR $(\{\rightarrow, \leftrightarrow\}, \{\rightarrow\})$ instance (where edges in R are taken as sibling-to-sibling edges). It is important to note that by our preprocessing phase, no new sibling-to-sibling edges have to be included. This computing step takes time $O(|P' \setminus S| + |R \cup X|) = O(N)$ by Corollary 13. Finally, re-insert all peer-to-peer edges from S in the obtained oriented graph, if the graph could be computed. Thus, the overall complexity is $O(N)$. □

Corollary 15. Acyclic ToR $(\{\rightarrow, —\}, \{\rightarrow\})$ *can be solved in time $O(N)$.*

Proof. On a given instance (G, P, R), we use basically the same algorithm as described in the proof of Theorem 14. The only difference is that we have additional customer-to-provider edges in the our set R that may contradict orientations of edges as forced by the peer-to-peer edges in a path. If we detect such a contradiction after the first phase of the algorithm, then we reject the instance. Otherwise we proceed as in the original algorithm. This is certainly an $O(N)$ algorithm. □

Theorem 16. Acyclic ToR $(\{\rightarrow, \leftrightarrow, —\}, \{\rightarrow\})$ *can be solved in time $O(N)$.*

Proof. For a given path set P and a partial orientation R, first test whether the S2S-components are S2S-closed. If note reject the instance. This is correct since we are not allowed to introduce new sibling-to-sibling edges and any allowed cycle in $G(P)$ containing only sibling-to-sibling edges cannot be transformed into an allowed cycle by replacing one edge with a customer-to-provider edge or a peer-to-peer edge. Otherwise compute a reduced path set P' and a new partial orientation R' which is the same as R with the sibling-to-sibling edges being removed which is basically the same as in the preprocessing phase of the algorithms used for Theorem 12. Finally, we solve $(G(P'), P', R')$ as an Acyclic ToR $(\{\rightarrow, —\}, \{\rightarrow\})$ instance in time $O(|P'| + |R'|) = O(N)$ using Corollary 15. □

4.3 Completions Using Peer-to-Peer Relations

We briefly discuss the implications of Theorem 16 for the inference problems where we use type set $\{\rightarrow, —\}$. Mainly, this can be handled with Lemma 9.

Theorem 17. Acyclic ToR $(\{\rightarrow, \leftrightarrow, —\}, \{\rightarrow, —\})$ *can be solved in time $O(N)$.*

Proof. First eliminate all sibling-to-sibling edges by computing a reduced instance $(G(P'), P', R')$ where R' only consists of customer-to-provider edges and peer-to-peer edges (as in Theorem 16). Second, solve $(G(P'), P', R')$ as an Acyclic ToR $(\{\rightarrow, —\}, \{\rightarrow\})$ instance (as in Corollary 15). Correctness follows from Lemma 9. ☐

The orientations obtained by Theorem 17 do not give us much information as they ignore peer-to-peer edges. In particular for the case $K = \emptyset$, it is therefore desirable to know how we obtain a candidate set for peer-to-peer edges from an acyclic and valley-free orientation with type set $\{\rightarrow\}$. To find such a set of *maximum* cardinality is already NP-hard for the case of only valley-free orientations [5]. Though it could be possible that we get a lower complexity if we additionally suppose an acyclic orientation, the proof in [5] actually shows that this is not the case: deciding whether there are at least k edges for a given path set P and a given valley-free and acyclic orientation with type set $\{\rightarrow\}$ that can be oriented as — is NP-complete. On the other hand, we can easily compute a *maximal* set of peer-to-peer candidate edges in time $O(|P|^2)$ simply by iterating over all edges and checking valley-freeness and acyclicity of the resulting orientation after re-orienting the edges solely.

4.4 Completions Using Sibling-to-Sibling Relations

Finally, we turn to the inference problem using type sets U for completions such that $\leftrightarrow$ lies in U. In contrast to all cases we considered so far and which all can be solved in linear time, we will now obtain computationally hard problems.

We first mention some exceptions that are easily solvable.

Proposition 18. *Let U be any type set containing $\leftrightarrow$. Then,* Acyclic ToR $(\emptyset, U)$ *and* Acyclic ToR $(\{\leftrightarrow\}, U)$ *can be solved in time $O(N)$.*

Proof. Orienting all not-yet-oriented edges as sibling-to-sibling edges is a solution. ☐

Proposition 19. *Let $K \subseteq \{\rightarrow, \leftrightarrow\}$ be any type set. Then,* Acyclic ToR $(K, \{\leftrightarrow, —\})$ *can be solved in time $O(N)$.*

Proof. Again, orient all remaining edges as sibling-to-sibling edges. In contrast to the previous proposition, we now have to check whether the orienation is acyclic and valley-free. This can be done in time $O(N)$. ☐

All remaining cases constitute NP-hard inference problems. To prove this, we take a closer look at sibling-to-sibling relations. As sibling-to-sibling relations establish an equivalence relation in any acyclic and valley-free orientation, we obtain a partition $[B_1, \ldots, B_r]$ of V from our orientation, i.e., a collection of non-empty subsets of V satisfying $B_i \cap B_j = \emptyset$ for $i \neq j$, and $B_1 \cup \cdots \cup B_r = V$, where the B_i are just equivalence classes according to the sibling-to-sibling relation, i.e., all B_i's are S2S-closed.

If the orientation is not already known to us then we search for suitable candidates for such partitions. We say that a pair $(P, [B_1, \ldots, B_r])$ is an *admissible decomposition* of P if and only if $[B_1, \ldots, B_r]$ is a partition of V such that the path set obtained by replacing all vertices of the same block with a unique representative and afterwards removing all multiple occurrences of representatives in all paths, allows an acyclic and valley-free orientation without $\leftrightarrow$. (Notice that this also includes S2S-looplessness of paths.) We further define the standard refinement relation $\subseteq$ on partitions which allows us to order partitions. Let $[A_1, \dots, A_s]$ and $[B_1, \dots, B_r]$ be two partitions of the same set V. We define

$$[A_1, \dots, A_s] \subseteq [B_1, \dots, B_r] \iff_{\text{def}} (\forall i, 1 \leq i \leq s)\,(\exists j, 1 \leq j \leq r)\,[\ A_i \subseteq B_j\ ].$$

Intuitively, the refinement relations holds between two partitions if we can union components of the finer partition to obtain the coarser partition, i.e., the partition with fewer components. We easily observe that admissible decompositions behave monotonically with respect to the refinement relation, i.e., if for partitions $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A} \subseteq \mathcal{B}$ and $\mathcal{A}$ is an admissible decomposition of P, then so is $\mathcal{B}$.

The following five theorems contain the simplest cases (in terms of pre-knowledge type sets) that are NP-hard and that all together are sufficient to imply NP-hardness for all inference problems not considered so far.

Theorem 20. Acyclic ToR $(\{\to\}, \{\to, \leftrightarrow\})$ *is* NP-*hard.*

Proof. For the proof of the NP-hardness, we use Two-in-Three Sat, i.e., the problem where we ask, given a 3CNF H, whether there is an assignment to all variables of H such that in each clause exactly two of the three literals are true. This problem is easily seen to be NP-complete by reduction from the well-known NP-complete problem One-in-Three SAT [14]. It will be enough to reduce Two-in-Three Sat to the decision version of Acyclic ToR $(\{\to\}, \{\to, \leftrightarrow\})$. Let H be an arbitrary 3CNF having $m \geq 3$ clauses $C_1, \dots, C_m$, each having exactly three different literals, and variables $x_1, \dots, x_n$, i.e., $H = C_1 \wedge C_2 \wedge \dots \wedge C_m$. We construct a path set P on the vertex set $\{C_1, \dots, C_m, x_1, \dots, x_n, \overline{x_1}, \dots, \overline{x_n}\}$. Define the following sets of paths:

$$\begin{aligned}
P_1 &=_{\text{def}} \{\ (C_1, C_2, C_3),\ (C_2, C_3, C_1),\ (C_3, C_1, C_2)\ \} \cup \{(\ C_1, C_j, C_2) \mid 4 \leq j \leq m\} \\
P_2 &=_{\text{def}} \{\ (x_i, x_j, \overline{x_i}) \mid 1 \leq i, j \leq n \text{ and } i \neq j\} \cup \{\ (x_1, C_i, \overline{x_1}) \mid 1 \leq i \leq m\} \\
P_3 &=_{\text{def}} \{\ (l_{i1}, C_i, l_{i2}),\ (l_{i2}, C_i, l_{i3}),\ (l_{i3}, C_i, l_{i1}),\ (l_{i1}, \overline{l_{i2}}, l_{i3}) \mid \\
&\qquad\qquad 1 \leq i \leq m \text{ and } C_i = (l_{i1} \vee l_{i2} \vee l_{i3})\ \}
\end{aligned}$$

The set P_1 guarantees that all clause vertices C_i belong to the same set, P_2 separates the literals from their negated literals, and P_3 indicates which literals will be satisfied. Now define $P =_{\text{def}} P_1 \cup P_2 \cup P_3$. Note that clearly, P can be computed in time polynomial in the number of clauses and variables of the input. We will show that

$$\begin{aligned}
&H \in \text{Two-in-Three Sat} \iff \\
&\qquad (G(P), P, \{x_1 \to \overline{x_1}\}) \text{ is a solvable Acyclic ToR } (\{\to\}, \{\to, \leftrightarrow\}) \text{ instance.}
\end{aligned}$$

We prove both directions separately.

For ($\Rightarrow$), let $I : \{x_1, \dots, x_n\} \to \{0, 1\}$ be an assignment to variables witnessing that $H \in$ Two-in-Three Sat. Define a set U to consist of all literals made true and all clauses. More specifically,

$$\begin{aligned}
U =_{\text{def}} &\ \{\ x_i \mid 1 \leq i \leq n \text{ and } I(x_i) = 1\ \} \cup \{\ \overline{x_i} \mid 1 \leq i \leq n \text{ and } I(x_i) = 0\ \} \cup \\
&\cup \{\ C_i \mid 1 \leq i \leq m\ \}.
\end{aligned}$$

Hence, $\overline{U} = \{ \ \overline{x_i} \mid 1 \leq i \leq n \text{ and } I(x_i) = 1 \ \} \cup \{ \ x_i \mid 1 \leq i \leq n \text{ and } I(x_i) = 0 \ \}$. Clearly, $x_1 \in U \Leftrightarrow \overline{x_1} \notin U$. We are done if we can show that $[U, \overline{U}]$ is an admissible decomposition of P. In the following we use U (or $\overline{U}$) to denote a representative element of U (or, $\overline{U}$, respectively). We now consider all path sets individually:

1. Since $C_i \in U$ for all $1 \leq i \leq m$, all paths in P_1 have the form (U, U, U) which simplifies to (U).
2. Without loss of generality, suppose $x_i \in U$ which immediately implies that $\overline{x_i} \notin U$. It follows that the paths of P_2 have the form $(U, U, \overline{U})$ or $(U, \overline{U}, \overline{U})$ which both simplify to $(U, \overline{U})$.
3. In each clause, exactly two literals are satisfied. Without loss of generality, suppose that l_{i_1} and l_{i2} are these literals for clause C_i. Then, the paths of P_3 that correspond to C_i all have forms (U, U, U), $(U, U, \overline{U})$, or $(U, \overline{U}, \overline{U})$, thus, simplify to (U) or $(U, \overline{U})$.

Consequently, after eliminating multiple occurrences of paths, P simplifies to a subset of $\{(U), (\overline{U}), (U, \overline{U}), (\overline{U}, U)\}$ which, obviously, allows acyclic and valley-free orientations. Hence, $(G(P), P, \{x_1 \to \overline{x_1}\})$ is a solvable instance of Acyclic ToR $(\{\to\}, \{\to, \leftrightarrow\})$.

For $(\Leftarrow)$, we assume that $(G(P), P, \{x_1 \to \overline{x_1}\})$ is a solvable Acyclic ToR $(\{\to\}, \{\to, \leftrightarrow\})$ instance, i.e., there exists a decomposition $[U, \overline{U}] \subset [V]$ of P such that $x_1 \in U \Leftrightarrow \overline{x_1} \notin U$. Note that we can restrict ourselves to such 2-component decompositions because of the $\subseteq$-monotonicity of admissible decompositions. Without loss of generality, we may assume that $C_1 \in U$. Thus, $\{C_1, \ldots, C_m\} \subseteq U$ because of the definition of P_1. Furthermore, we have for all $1 \leq i \leq n$, $x_i \in U \Leftrightarrow \overline{x_i} \notin U$ (otherwise $U = V$ or $\overline{U} = V$ because of definition of P_2). This allows to define an assignment $I : \{x_1, \ldots, x_n\} \to \{0, 1\}$ as follows

$$I(x_i) =_{\text{def}} \begin{cases} 1 & \text{if } x_i \in U, \\ 0 & \text{if } \overline{x_i} \in U. \end{cases}$$

We have to prove that I satisfies exactly two literals in each clause. Let C_i be any clause with literals l_{i1}, l_{i2}, and l_{i3}. As there are paths in P_3 having the form (l_{ij}, C_i, l_{ik}), there is an $l_{ir} \in U$. Without loss of generality, we assume that l_{i1} is such a literal. Since there exists for each pairs of literals of C_i such a path in P_3, there exists another literal $l_{is} \in U$, $r \neq s$. Without loss of generality, l_{i2} is such a literal. Due to the path $(l_{i1}, \overline{l_{i2}}, l_{i3}) \in P_3$, we know that l_{i3} is not in U. Overall, by definition of I, for each clause exactly two literals are made true. This shows $H \in$ Two-in-Three Sat. ☐

Theorem 21. *Acyclic ToR $(\{—\}, \{\to, \leftrightarrow\})$ is NP-hard.*

Proof. We describe a reduction from Acyclic ToR $(\{\to\}, \{\to, \leftrightarrow\})$. Let (G, P, R) be an instance to that problem. Let $u \to v \in R$ be any oriented edge. We are done if we can force such an edge to be oriented in the same way without using a customer-to-provider edge. This can be achieved as follows. We add to the path set two new paths: $(u, v, \perp)$ and $(u, \perp, v)$ where $\perp$ is neither contained in P nor in R. Furthermore, we add the peer-to-peer edge $v —\perp$ to R. (Note that we can take the same new vertex $\perp$ for all edges in R.) It suffices to show that u and v cannot be siblings. Assume to the contrary, that u and v are siblings. Then, to avoid an oriented cycle $(u, v, \perp, u)$, u must have a peer-to-peer edge with $\perp$. However, this is a contradiction to valley-freeness, as $(u, \perp, v)$ now has two peer-to-peer edges. ☐

Theorem 22. Acyclic ToR $(\{—\},\{\leftrightarrow,—\})$ *is* NP*-hard.*

Proof. For $U=\{\leftrightarrow,—\}$, a careful analysis of the proof of Theorem 20 reveals that a 3CNF H belongs to Two-in-Three Sat if and only if $(G(P),P,\{x_1 — \overline{x_1}\})$ is a solvable instance of Acyclic ToR $(\{—\},\{\leftrightarrow,—\})$. □

Theorem 23. Acyclic ToR $(\{—\},\{\rightarrow,\leftrightarrow,—\})$ *is* NP*-hard.*

Proof. For $U=\{\rightarrow,\leftrightarrow,—\}$, note that the proof of Theorem 22 relies on a partition of the vertex set into two components. Thus any acyclic and valley-free orientation assigns to edges who's vertices belong to different components the same edge type. Thus, we obtain that a 3CNF H belongs to Two-in-Three Sat if and only if $(G(P),P,\{x_1 — \overline{x_1}\})$ is a solvable Acyclic ToR $(\{—\},\{\rightarrow,\leftrightarrow,—\})$ instance. □

Theorem 24. Acyclic ToR $(\{\rightarrow\},\{\rightarrow,\leftrightarrow,—\})$ *is* NP*-hard.*

Proof. Let K be a type set containing $\rightarrow$ but not $—$. Since simultaneously replacing *all* peer-to-peer edges of an acyclic and valley-free orientation with sibling-to-sibling edges maintains acyclicity and valley-freeness, (G,P,R) is a solvable Acyclic ToR $(K,\{\rightarrow,\leftrightarrow,—\})$ instance if and only if (G,P,R) is a solvable Acyclic ToR $(K,\{\rightarrow,\leftrightarrow\})$ instance. Thus, Acyclic ToR $(K,\{\rightarrow,\leftrightarrow,—\}$ is NP-hard by Theorem 20. □

5 Conclusion

We studied algorithmic solutions for the acyclic all-paths type-of-relationship problem. In particular we designed a linear-time algorithm for finding an acyclic and valley-free completion of a partial orientation of a set of AS paths, that only uses customer-to-provider relations for completion whereas the partial orientation can be expressed with arbitrary types of standard AS relationships. Based on some evident assumptions on the size of ASs, acyclicity conditions are given in terms of forbidden graph patterns. The algorithm provides prospects for combining combinatorial methods with more non-combinatorial techniques to explore the solution space of possible Internet hierarchies. To evaluate the quality of this algorithm, we plan to supplement the theoretical study of this paper with experimental investigations. In contrast, permitting sibling-to-sibling relations for completion makes most of the problem versions NP-hard.

Several algorithmic problems remain open. However, from a theoretical point of view, the most interesting open question is whether acyclicity of the Internet hierarchy can be deduced via a game-theoretic analysis. Are acyclically oriented AS graphs Nash equilibria for classes of network creation games (in the sense of [10, 2])?

Acknowledgment. For helpful hints and discussions we thank Vinay Aggarwal, Stefan Eckhardt, Klaus Holzapfel, Olaf Maennel, Johannes Nowak, Thomas Schwabe, Arne Wichmann. In particular we are deeply grateful to Benjamin Hummel for carefully proofreading earlier drafts of this paper.

References

[1] C. Alaettinoğlu. Scalable router configuration for the Internet. In *Proceedings of the 5th International Conference on Computer Communications and Networks (ICCCN'96)*. IEEE Computer Society Press, Washington, D.C., 1996.

[2] S. Albers, S. Eilts, E. Even-Dar, Y. Mansour, and L. Roditty. On Nash equilibria for a network creation game. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'06)*, pages 89–98. SIAM, Philadelphia, PA, 2006.

[3] H. Chang, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger. Towards capturing representative AS-level Internet topologies. *Computer Networks*, 44(6):737–755, 2004.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 2nd edition, 2001.

[5] G. Di Battista, T. Erlebach, A. Hall, M. Patrignani, M. Pizzonia, and T. Schank. Computing the types of the relationships between autonomous systems. *IEEE/ACM Transactions on Networking*, 2006. To appear. This is an expanded version of [6, 9].

[6] G. Di Battista, M. Patrignani, and M. Pizzonia. Computing the types of the relationships between autonomous systems. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03)*, pages 156–165. IEEE Computer Society Press, Washington, D.C., 2003.

[7] X. A. Dimitriopoulos, D. V. Krioukov, B. Huffaker, K. C. Claffy, and G. F. Riley. Inferring AS relationships: Dead end or lively beginning? In *Proceedings of the 4th International Workshop on Experimental and Efficient Algorithms (WEA'05)*, volume 3503 of *Lecture Notes in Computer Science*, pages 113–125. Springer-Verlag, Berlin, 2005.

[8] T. Erlebach, A. Hall, A. Panconesi, and D. Vukadinović. Cuts and disjoint paths in the valley-free path model of Internet BGP routing. In *Proceedings of the 1st Workshop on Combinatorial and Algorithmic Aspects of Networking (CAAN'04)*, volume 3405 of *Lecture Notes in Computer Science*, pages 49–62. Springer-Verlag, Berlin, 2004.

[9] T. Erlebach, A. Hall, and T. Schank. Classifying customer-provider relationships in the Internet. In *Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN'02)*, pages 538–545. ACTA Press, Calgary, 2002.

[10] A. Fabrikant, A. Luthra, E. Maneva, C. H. Papadimitriou, and S. J. Shenker. On a network creation game. In *Proceeding of the 22nd ACM Symposium on Principles of Distributed Computing (PODC'03)*, pages 347–351. ACM Press, New York, NY, 2003.

[11] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs. Locating Internet routing instabilities. In *Proceedings of the ACM SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'04)*, pages 205–218. ACM Press, New York, NY, 2004.

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

[13] L. Gao and J. Rexford. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking*, 9(6):681–692, 2001.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* A Series of Books in the Mathematical Sciences. W. H. Freeman and Company, New York, NY, 1979.

[15] R. Govindan and A. Reddy. An analysis of Internet inter-domain topology and route stability. In *Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'97)*, pages 850–857. IEEE Computer Society Press, Washington, D.C., 1997.

[16] T. G. Griffin and B. J. Premore. An experimental analysis of BGP convergence time. In *Proceedings of the 9th Annual International Conference on Network Protocols (ICNP'01)*, pages 53–61. IEEE Computer Society Press, Washington, D.C., 2001.

[17] T. G. Griffin and G. T. Wilfong. An analysis of BGP convergence properties. *ACM SIGCOMM Computer Communication Review*, 29(4):277–288, 1999.

[18] J. Hawkinson and T. Bates. Guidelines for creation, selection, and registration of an autonomous system (AS). RFC 1930, The Internet Society, 1996.

[19] G. Huston. Interconnection, peering and settlements—Part II. *The Internet Protocol Journal*, 2(2):2–23, 1999.

[20] C. Labovitz, A. Ahuja, R. Wattenhofer, and S. Venkatachary. The impact of Internet policy and topology on delayed routing convergence. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'01)*, pages 537–546. IEEE Computer Society Press, Washington, D.C., 2001.

[21] C. Labovitz, G. R. Malan, and F. Jahanian. Origins of Internet routing instability. In *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'99)*, pages 218–226. IEEE Computer Society Press, Washington, D.C., 1999.

[22] O. Maennel and A. Feldmann. Realistic BGP traffic for test labs. In *Proceedings of the ACM SIGCOMM 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'02)*, pages 31–44. ACM Press, New York, NY, 2002.

[23] W. B. Norton. Internet Service Providers and peering. Equinix White Paper, Equinix, Inc., Foster City, CA, 2001.

[24] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771, The Internet Society, 1995.

[25] G. Siganos and M. Faloutsos. Analyzing BGP policies: Methodology and tool. In *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*, pages 1640–1651. IEEE Computer Society Press, Washington, D.C., 2004.

[26] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02)*, pages 618–627. IEEE Computer Society Press, Washington, D.C., 2002.

[27] H. Tangmunarunkit, J. Doyle, R. Govindan, S. Jamin, S. J. Shenker, and W. Willinger. Does AS size determine degree in AS topology? *ACM SIGCOMM Computer Communication Review*, 31(5):7–10, 2001.

[28] H. Tangmunarunkit, R. Govindan, S. J. Shenker, and D. Estrin. The impact of routing policy on Internet paths. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'01)*, pages 736–742. IEEE Computer Society Press, Washington, D.C., 2001.

[29] I. van Beijnum. *BGP*. O'Reilly & Associates, Sebastopol, CA, 2002.

[30] J. Xia and L. Gao. On the evaluation of AS relationship inferences. In *Proceedings of the 47th Annual IEEE Global Telecommunications Conference (Globecom'04)*, volume 3, pages 1373–1377. IEEE Communication Society, New York, NY, 2004.