Acyclic Type-of-Relationship Problems on the Internet: An Experimental Analysis[*]

Benjamin Hummel
Fakultät für Informatik
Technische Universität München

Sven Kosub
Fakultät für Informatik
Technische Universität München

ABSTRACT

An experimental study of the feasibility and accuracy of the acyclicity approach introduced in [14] for the inference of business relationships among autonomous systems (ASes) is provided. We investigate the maximum acyclic type-of-relationship problem: on a given set of AS paths, find a maximum-cardinality subset which allows an acyclic and valley-free orientation. Inapproximability and NP-hardness results for this problem are presented and a heuristic is designed. The heuristic is experimentally compared to most of the state-of-the-art algorithms on a reliable data set. It turns out that the proposed heuristic produces the least number of misclassified customer-to-provider relationships among the tested algorithms. Moreover, it is flexible in handling pre-knowledge in the sense that already a small amount of correct relationships is enough to produce a high-quality relationship classification. Furthermore, the reliable data set is used to validate the acyclicity assumptions. The findings demonstrate that acyclicity notions should be an integral part of models of AS relationships.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Protocols, Network Operations; F.2.2 [**Nonnumerical Algorithms and Problems**]: Computations on discrete structures

General Terms

Measurement, Algorithms

Keywords

Inter-domain Routing, AS Relationships, Algorithms

[*]All data sets and Java implementations of the algorithms used herein are available online at http://www14.in.tum.de/software/BGP/hummel-kosub-07.html.

1. INTRODUCTION

Interrelationship analysis of autonomous systems (ASes, for short) has recently attracted much attention in both theoretical and practical research on Internet inter-domain routing with BGP (see, e.g., [8, 19, 18, 7, 4, 14, 6]). This interest is motivated by insights how routing stability and quality in the Internet is influenced not only by physical connections between ASes but heavily by their business relationships. As business contracts are subject to privacy, computational tools and techniques are required to infer close-to-reality relationship classifications from publicly available resources such as WHOIS databases, the Internet Routing Registry [15], or BGP beacons (e.g., [17, 21]).

A useful approach is the interpretation of observable BGP routes. Techniques based on this approach usually work as follows: collect a set of AS paths from BGP routers, obtain an AS-level connectivity graph (the *AS graph*) by merging all AS paths, and label the AS graph with business relations such that all—actually, as many as possible—collected AS paths are valley-free. Valley-freeness is a characteristic property of AS paths based on economic rationality, which in a simplified version says that, in the direction of traffic, a customer-to-provider link should never follow a provider-to-customer link. The implementations range from purely combinatorial (e.g., [4, 14]) to purely heuristical one's (e.g., [8, 19, 23]). Though there has been some criticism of unrealistic classifications [7], the empirical findings are encouraging for further developments.

In [14], acyclicity has been added to valley-freeness as another structural condition of AS graph labelings. The rationale for acyclicity is that it is unlikely the case that an AS A is a provider of an AS B, AS B is a provider of an AS C, and AS C is a provider of AS A. It has turned out that finding labelings which are both valley-free on the path set and acyclic in the AS graph is easy, even if we want to respect explicit pre-knowledge (following the partialness-to-entireness methodology [23]). However, this theoretical feasibility and plausibility of the acyclicity approach to AS relationship inference has not been supported with empirical evidence in [14].

In this paper we contribute to the experimental analysis of acyclic type-of-relationship problems in three ways.

- First, we operationalize acyclicity. In [14], algorithms have been presented to test whether all paths of a given path set allow acyclic and valley-free orientations. In practice, collected path sets are expected to fail this test. Thus, these algorithms are of theoretical interest only. We consider the problem to find acyclic orientations which maximize the number of valley-free paths. We provide lower bounds on the approximability of this problem (which implies the NP-hardness as well) and we design a fast heuristic for finding acyclic orientations which are valley-free on a large part of a given path set.

- Second, we validate the acyclicity assumptions from [14]. In doing so, one issue is obtaining a reliable data set. We report on several techniques employed. The graph we receive from the reliable data set when only using customer-to-provider relationship is acylic. Including peer-to-peer relationships is more problematic. It seems that we do not yet have the right understanding how peer-to-peer relationships affect the graph-theoretical structure of BGP routes and the AS graph.
- Third, we compare the inference quality of our heuristic to a set of standard algorithms from the literature. On the reliable data set, our heuristic produces the lowest number of misclassified customer-to-provider edges among all algorithms tested (in numbers: approximately 0.3% of all edges in the set are misclassified). We further test how the inference quality of the algorithms depend on initial pre-knowledge. Here, it is observed that for our approach the relative number of misclassified customer-to-provider edges is nearly independent of the amount of pre-knowledge. That is, already a small amount of correct relationships is enough to infer a high-quality classification among the ASes.

All in all, the findings of this paper indicate that the method proposed is a feasible and flexible heuristic with excellent inference quality (at least with respect to customer-to-provider relationships) and that acyclicity should be an integral part of any further accuracy improvements.

2. RELATED WORK

Several algorithms have been proposed to infer relationship types from AS paths. The first attempt was made in [8] where the valley-free path model was introduced and a heuristic was designed based on statistical properties of a given path set. This approach was pushed further in [19, 23] to combine valley-free path labelings obtained from different observation points and from sources other than AS paths. In [4] (and some precursor papers), a combinatorial approach is developped based on expressing valley-freeness of paths in terms of the 2SAT problem. A combination of the 2SAT-based formulation of valley-freeness and the statistical properties of path sets in terms of mathematical programming has been proposed in [7, 6]. The acyclicity approach to interrelationship analysis is from [14]. Computational techniques not based on AS path interpretation have been proposed and discussed in, e.g., [18, 23, 5, 6].

3. PRELIMINARIES

We briefly describe a simple, abstract model of inter-domain routing in the Internet using BGP (see, e.g., [22, 16, 9, 8]).

3.1 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 [10]. An AS aims at providing global reachability for its IP adresses. To achieve this goal, ASes having common physical connections exchange routing information as governed by their own local routing policies. BGP is the *de facto* standard protocol to manage data traffic between ASes for inter-domain routing as well as for route propagation.

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 1: The Selective Export Rule.

Reachability in the Internet depends on (physical) connectivity and the contractual relationships between ASes. The most fundamental binary business relationships are customer-to-provider (where the provider sells routes to the customer), peer-to-peer (where the involved ASes provide special routes to their customers but no transit for each other), and sibling-to-sibling (where both ASes belong to the same administrative domain). Evidently, sibling-to-sibling relations are transitive. More peculiar relationships appear in the real world (see, e.g., [8]). We restrict ourselves to the three mentioned types of relationships.

More specifically, let V be a set of AS numbers. The undirected graph $G = (V, E)$ where E corresponds to physical connections between ASes is called a *connectivity graph* at the AS level or simply *AS graph*. For any AS $v \in V$ denote the set of all siblings of v (including v itself) as $\mathrm{Sibl}(v)$, and $R(v)$ 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 ASes 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 ASes 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

a *customer route* of $v \Longleftrightarrow_{\mathrm{def}}$
leftmost $u_i \notin \mathrm{Sibl}(v)$ is a customer of v,
a *provider route* of $v \Longleftrightarrow_{\mathrm{def}}$
leftmost $u_i \notin \mathrm{Sibl}(v)$ is a provider of v,
a *peer route* of $v \Longleftrightarrow_{\mathrm{def}}$
leftmost $u_i \notin \mathrm{Sibl}(v)$ is a peering partner of v,
an *own route* of $v \Longleftrightarrow_{\mathrm{def}}$
for all $1 \le i \le r$, $u_i \in \mathrm{Sibl}(v)$.

Now, typically (at least, recommendably), ASes set up their export policies according to the Selective Export Rule [1, 12, 8] as described in Figure 1. In our simplified model, the receiving AS gets from an AS those routes destined for it and prolongated with the number of the sending AS as the new leftmost AS number in the path.

3.2 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 $(u, v) \in E \Leftrightarrow (v, u) \in E$. An *orientation* φ *of* G is a mapping from E to T where T denotes the set of possible edge-types, which are taken from:

$\rightarrow$	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 $\rightarrow$. That is, for all $(u,v) \in E$, $\varphi(u,v) = \leftarrow \;\Leftrightarrow\; \varphi(v,u) = \rightarrow$. Thus, if we allow $\rightarrow$ as a possible edge type, then we immediately allow $\leftarrow$ as a possible edge type as well. Instead of $\varphi(u,v) = \rightarrow$ for an edge $(u,v) \in E$ we also write $u \rightarrow v$.

We extend φ from edges to walks. 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 a word in $\{\leftarrow, \rightarrow, —, \leftrightarrow\}^*$. We typically use regular expressions to describe walk types given an orientation. An important property of orientations is valley-freeness, which is stated here in terms of regular patterns of paths.

Definition 1 ([**8**]). *Let G be a 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 ASes never route data from one of their providers to another of their providers because instead of earning money, they would have to pay twice for these data streams.

Theorem 2 ([**8**]). *Let $G = (V, E)$ be an AS graph and let P be a set of AS paths of all BGP routing tables, i.e., $P \subseteq \bigcup_{v \in V} R(v)$. If all ASes export their routes according to the Selective Export Rule, then there is an orientation of P such that all paths in P are valley-free.*

3.3 The Acyclicity Assumptions

Following [14], we summarize reasonable acyclicity structures within a connectivity graph, i.e., patterns of oriented cycles which are forbidden to be contained in the graph. An oriented cycle (in its simplest form) can be interpreted as someone being its own provider and customer. The following definition of an oriented cycle has been proposed in [14].

Definition 3 ([**14**]). *Let G be any graph, and let $\varphi(G)$ be an orientation of G. Let C be any minimal cycle of G, i.e., a cycle that does not contain a vertex twice. C is said to be an* oriented cycle *of $\varphi(G)$ if and only if $\varphi(C)$ lies in*

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

To exemplify the definition, Figure 2 shows the 16 non-isomorphic triads of the 64 possible orientations of a complete graph on three vertices. Half of them are oriented cycles according to Definition 3 and half of them are not. Note that in the case that φ does not exhaust the full type set $\{\rightarrow, —, \leftrightarrow\}$, the patterns of oriented cycles simplify. 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. We call an orientation *acyclic* if it contains no oriented cycles. Testing whether an orientation is acyclic can be done fastly by standard techniques (see, e.g., [3, 14]).

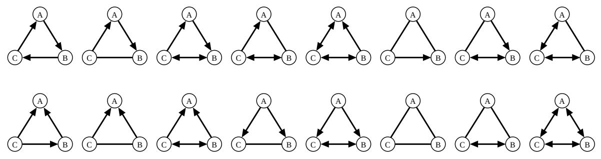

Figure 2: Non-isomorphic triads. All triads in the upper row are forbidden and all triads in the lower row are allowed.

4. THE MAXIMUM ACYCLIC TYPE-OF-RELATIONSHIP PROBLEM

An algorithm for finding an acyclic and valley-free orientation of an AS graph for a given path set is presented in [14]. We modify this algorithm to find an acyclic orientation which is valley-free for a *large part* of the AS paths (even when there is no valley-free orientation including all paths) followed by some improvements. Formally, we consider the following optimization problem (see [2] for notation):

Problem:	Maximum Acyclic ToR
Input:	AS path set P and induced AS Graph
Solution:	A subset $P' \subseteq P$ allowing an acyclic and valley-free orientation
Measure:	The cardinality $\|P'\|$ of the subset P'

We mention some theoretical results showing that this problem is computationally difficult. *All proofs can be found in the full version [11] of the paper.*

Theorem 4. *Unless* P = NP*, there is no polynomial-time approximation scheme for* Maximum Acyclic ToR.

The decision version of Maximum Acyclic ToR consists of all instances (P, k) such that P is a (multi)set of AS paths containing a subset $P' \subseteq P$ which has at least k paths and which allows an acylic and valley-free orientation.

Corollary 5. *The decision version of* Maximum Acyclic ToR *is* NP*-complete.*

On the positive side, we do not know any non-trivial bound on the approximation quality of Maximum Acyclic ToR. However, if we restrict path lengths, then we obtain a constant approximation ratio.

Theorem 6. Maximum Acyclic ToR *limited to paths of length at most $k \in \mathbb{N}_+$ can be approximated within a factor of $\frac{2^k}{(k+1)!}$ of the optimum in polynomial time.*

Unfortunately, these fractions decrease very quickly as path length increases, e.g., for path length 2 the fraction is $\frac{2}{3} \approx 66.7\%$, for length 3 it is $\frac{1}{3} \approx 33.3\%$, for length 4 it is $\frac{2}{15} \approx 13.3\%$, and for length 5 it is already $\frac{2}{45} \approx 4.4\%$.

4.1 The Basic Heuristic

The algorithm from [14] for testing whether a path set allows acyclic and valley-free orientations is based on the observation that a leaf AS (i.e., one that itself has no customers) cannot be in the middle of any AS path. We describe it combined with the extension for discarding an interfering path, but before digging into the details we fix some notation.

Algorithm 1 Heuristic “AHeu”

```
1: Input: AS path set $P$, AS graph $G = (V, E)$ for $P$
2: Output: set $N$ of discarded paths, an acyclic orientation of
   $G$ with $\rightarrow$ edges (valley-free for all paths in $P \setminus N$)

3: $\text{count}(v)$ := number of paths for which $v$ is an inner node
4: $F := \{v \in V \mid \text{count}(v) = 0\}$, $R = \emptyset$, $N = \emptyset$, done := false
5: while $\neg$ done do
6:     while $F \neq \emptyset$ do
7:         remove vertex $u$ from $F$
8:         foreach $v \in V \setminus R$ with $\{u, v\} \in E$ do
9:             orient $\{u, v\}$ as customer-to-provider
10:            foreach $p \in P$ with $u$ and $v$ as neighbors do
11:                if $v$ is inner node of $p$ relative to $R$ then
12:                    $\text{count}(v) := \text{count}(v) - 1$
13:                    if $\text{count}(v) = 0$ then $F := F \cup \{v\}$
14:        $R := R \cup \{u\}$
15:    if $R = V$ then done := true
16:    else
17:        $v := \text{argmin}_{u \in V \setminus R}\, \text{count}(u)$
18:        foreach path $p \in P$ with $v \in p$ do
19:            if $v$ is an inner node of $p$ relative to $R$ then
20:                $N := N \cup \{p\}$
21:                foreach inner node $u$ of $p$ relative to $R$ do
22:                    $\text{count}(u) := \text{count}(u) - 1$
23:                    if $\text{count}(u) = 0$ then $F := F \cup \{u\}$
24: return $N$
```

During its execution the algorithm removes ASes which have been finished. To avoid having to change all the paths, we manage a set R of *removed* nodes. Given such a set, a node v in a path p is called *inner node of p relative to R* if it is surrounded in p by nodes u and w with $u, w \notin R$. A node not in R that is not an inner node for all paths of the paths set is called *free*.

In the algorithm (details in Algorithm 1) we count for each node v in $\text{count}(v)$ the number of paths for which v is an inner node. The set F of free nodes is then easily initialized by all nodes with count $= 0$. The main loop (lines 5–23) can be separated into two phases. The first phase (lines 6–14) is taken from the algorithm in [14]. While there is a free node u in F we interpret it as a leaf AS and thus orient the edges to its neighbors (not yet in R) as customer-to-provider. As it is removed afterwards (i.e., put into R) we adjust the count variables accordingly to find nodes which are now freed. If we can remove all ASes this way ($R = V$) we know that the orientation is valley-free and acyclic, as the nodes have been removed in topologically sorted order (each node had indegree 0 when it was removed).

If the first phase ran out of free nodes before all ASes could be removed, we need to create additional free nodes by discarding paths (starting from line 15). As we want to discard as little paths as possible, we select a node v which is an inner node for the minimal number of paths (as indicated by $\text{count}(v)$). By removing all those paths, v becomes a free node and we continue with the first phase.

We want to point out that the algorithm can easily be modified to work for a weighted path set, where the goal consists of minimizing the overall weight of the dropped paths.

4.2 Handling Pre-knowledge

If we already have partial information on the AS relationships we would like to incorporate this knowledge thereby improving the results of the algorithm. In [14] the influence of pre-knowledge on the complexity of testing whether an acyclic and valley-free orientation consistent with the pre-knowledge exists is discussed and an extension for the acyclic inference algorithm for handling known customer-to-provider edges is presented. As our heuristic is a modification of the algorithm given there, we can easily transfer this extension.

The idea is to introduce for each known customer-to-provider edge $u \rightarrow v$ a new path $(u, v, \perp)$, where $\perp$ is an artificial AS with $\text{count}(\perp) = \infty$. So the only way to make v a free node is to remove u which includes the introduction of an edge $u \rightarrow v$ as desired. Of course these new AS paths need not be constructed explicitly, but can be handled implicitly by modifying the heuristic above.

As the heuristic is allowed to drop paths hindering a consistent orientation, interpreting known edges as paths also allows dropping these edges in case of a conflict. Often however the pre-knowledge is trusted more than the set of AS paths. For this case we can increase the weight of these (virtual) paths introduced in this step. In our implementation all AS paths are weighted with 1 and all paths originating from known edges are assigned the same weight W. Thus a customer-to-provider edge may only be discarded, if we can “save” at least W AS paths instead. For the results shown later, W was set to 10.

4.3 Re-adding

After finding an AS path set which can be oriented valley-free, the DPP* heuristic from [4] enters a second phase where paths which had to be dropped before are re-added to the path set if possible. Two approaches for this are considered there. One is a voting process, the other one is for each single path to add it to the path set and only keep it if a valley-free orientation is still possible.

As the re-adding stage reduces the number of invalid paths we adapted this method for our heuristic. Unfortunately the simple and fast voting strategy does not work for our case as it does not necessarily preserve the global acyclicity. The alternative approach of adding paths one by one and retesting orientability works but is quite expensive if the number of dropped paths is high (as already observed by [4]). Therefore we adjusted re-adding as follows.

The first run of the heuristic returns a set N of discarded paths, from which we can determine the set of orientable paths P. We decide on the number $k > 1$ of additional runs we are willing to spend and partition N arbitrarily into k sets $N_1, \ldots, N_k$. For each such set N_i we then run the heuristic above for the path set $P \cup N_i$ and again receive a set of discarded paths N'. If $\|N'\| < \|N_i\|$ we could reduce the number of dropped paths and use $P := (P \cup N_i) \setminus N'$ from now on, otherwise we stick with the original P. As we treat edges from the pre-knowledge as AS paths as described before, this strategy works for re-adding those discarded edges as well. Results for the algorithm with re-adding are later shown for $k = 10$.

5. OBTAINING REAL-WORLD DATA

To run the inference algorithms we need valid AS paths used in the Internet. Additionally we are interested in at least partial information on the business relationships between autonomous systems, to both verify our inference results as well as using them as previous knowledge for the inference algorithms. As there is no single exhaustive source listing those relationships we have to use other publicly available information and try to extract them from it. We only give a short overview on the sources of the data (which was collected on 3/31/2006) here. More details are given in [11].

The AS paths were obtained from the routing tables available through route collectors [17, 21] and from public route servers [13]. From this set we removed erroneous paths and applied a normalization which discarded duplicate paths and path being subpaths of other AS paths, finally leaving us with a set of 2,002,680 paths of average length 3.43, containing 21,862 ASes and inducing 56,922 AS pairs.

Our set of known AS relationships results from applying methods similar to those described in [5, 18] to the WHOIS databases [15]. Additionally we used the approach from [23] on the BGP communities attribute ([20]) stored in the AS paths of our input set. Based on how often an edge orientation was found using the different methods we divided the edge set into two sets. The more reliable one, containing 2,739 customer-to-provider and 2,000 peer-to-peer edges, is used for the experiments in this paper. Results on the second set, which we expect to contain between 5 to 30 percent of incorrect data, are given in [11].

6. VALIDATING ACYCLICITY

Using the data from the previous section we intend to compare our acyclicity model to the real Internet. The graph we receive when only using customer-to-provider relationships actually is acyclic. However including peer-to-peer edges into these graphs creates cycles in both cases. To get an impression of how much acyclicity is violated we tested every triangle in the graphs. Out of 2,826 triangles 253 (9%) induce a directed cycle.

We take these results as an indication that indeed the overall structure of AS relationships is acyclic but our model of acyclicity is still imprecise when it comes to peer-to-peer relationships. This is probably mostly due to assuming the relation "roughly of the same size" to be transitive when interpreting peer-to-peer edges.

7. EXPERIMENTAL FINDINGS

In addition to our approach we implemented other algorithms for Maximum Acyclic ToRfor comparison, namely Gao's Heuristic [8], an approximation algorithm (APX) from [4] and the DPP* Heuristik based on a reduction to 2SAT (also [4]). We augmented these algorithms, which do no support the handling of preknowledge themselves, with a simple preprocessing routine which fixes all edges which can be inferred from the preknowledge and apllies the algorithm on the reduced AS-Graph. This preprocessing step and also an outline of these algorithms is given in [11].

To compare our acyclic inference heuristic (to which we refer as AHeu) to existing algorithms we executed all of them on the path set from Section 5 and compared the resulting edge classification to the edge set described there. We are interested in both the number of paths which are not oriented correctly (i.e., are not valley-free) and the number of customer-to-provider edges that were not inferred. As none of the algorithms is capable of identifying peer-to-peer relationships we do not compare the inferred results to our known peer-to-peer edges. An exception is Gao's algorithm as it introduces sibling-to-sibling edges which we do not want to include in the inferred results, thus paths containing a sibling-to-sibling edge are counted as invalid as well.

Algorithm	Invalid paths	Misclassified c-to-p for reliable edge set
Gao	27.366% (249 not valley-free) (54,7811 with s-to-s)	1.387% (4 as p-to-c) (34 as s-to-s)
APX	4.483% (89,775 not valley-free)	5.330% (146 as p-to-c)
DPP*	0.519% (10,391 not valley-free)	0.913% (25 as p-to-c)
AHeu	0.483% (9,666 not valley-free)	0.292% (8 as p-to-c)
AHeu (re-add)	0.413% (8,278 not valley-free)	0.329% (9 as p-to-c)

Figure 3: Results for inferring only customer-to-provider relationships

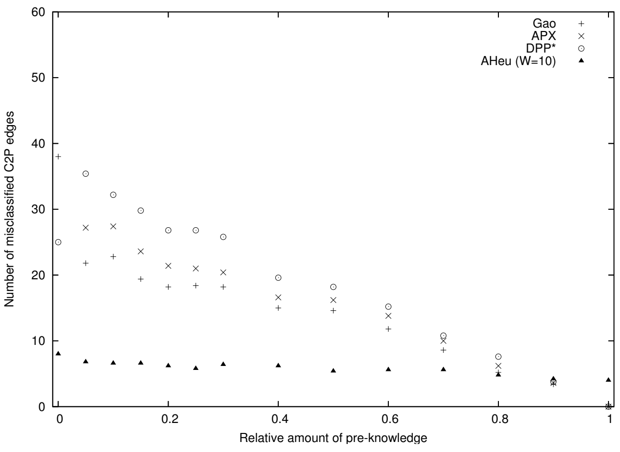

Figure 4: Number of edges misclassified

The detailed results of our experiments are shown in Figure 3. As seen our heuristic has the lowest number of invalid paths as well as the least number of errors when compared to the reference data. Additionally it is exemplified how using the re-adding strategy we can lower the number of invalid paths even more but at the cost of reliability of the resulting edge classifications.

Another aspect we are interested in is the behavior of these algorithms when having initial pre-knowledge of some of the edges. Therefore we repeated the experiment described before but provided the algorithms with a certain fraction of the edges used for comparison later. As the choice of edges provided to the algorithm has some influence on its results we averaged all results over 5 random samples of the edge set. Additionally the same samples were used for all of the algorithms. Our heuristic is the only one explicitly supporting pre-knowledge, so we used a simple preprocessing algorithm described in [11] to augment the remaining algorithms accordingly. This should be kept in mind when comparing the results as thus our heuristic is the only one capable of "trading" edge errors (i.e., violated pre-knowledge) for violated paths. It is interesting to note that already a small portion of pre-knowledge fixes a large portion of the edges[1].

[1]No matter if we used 5% or 100% of the known edges as pre-knowledge, about 42,000 edges of the AS graph were fixed after preprocessing. On the other hand about 20,000 AS path had to be dropped in this phase (details in [11]).

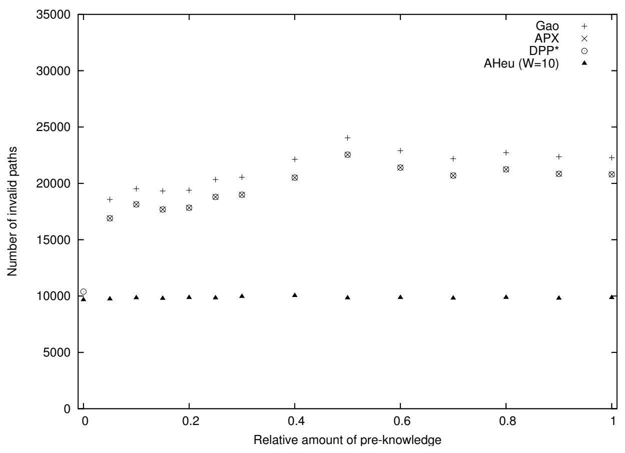

Figure 5: Number of invalid paths

As stated before we are interested both in the number of misclassified edges and the number of invalid paths. These numbers are given in Figures 4 resp. 5 using the edge set both as pre-knowledge and for comparison. According to these plots the performance of our heuristic is hardly influenced by the amount of pre-knowledge available which we take as an indication for the high quality of the inferred results. We provide analogous plots for the full data set in [11]. They show the same trend with the difference of a higher number of overall errors due to the inexact data in this set. This illustrates nicely how AHeu keeps giving good results even in the presence of partially invalid pre-knowledge while the other algorithms (partly due to the inflexible preprocessing step) have to trust this knowledge at the cost of a huge number of invalid paths.

8. CONCLUSION

We studied the acyclicity approach to AS relationship inference introduced in [14] from an experimental point of view. On the one side, we presented both theoretical and practical evidence that this approach is feasible and, in large parts, accurate. The described, heuristic algorithm AHeu turned out to be easily implementable, fast, and flexible with respect to incorporating initial pre-knowledge, and outperformed the state-of-the-art algorithms proposed in the literature. Moreover, the acyclicity of all customer-to-provider relationships within the reliable data set could be confirmed. These findings suggest to integrate acyclicity notions in detailed models of AS relationships.

On the other side, we have learned that acyclicity with respect to peer-to-peer relationships is not yet fully captured. The underlying assumption that the roughly-the-same-size relation is transitive seems too much a simplification. We consider finding a more accurate problem formulation involving acyclicity and peer-to-peer relationships as the main open issue of this paper.

Acknowledgments

We thank Wolfgang Mühlbauer for helpful discussions.

9. REFERENCES

[1] C. Alaettinoğlu. Scalable router configuration for the Internet. In *ICCCN'96*. IEEE, 1996.

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

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

[4] G. Di Battista, T. Erlebach, A. Hall, M. Patrignani, M. Pizzonia, T. Schank. Computing the types of the relationships between autonomous systems. *IEEE/ACM Transactions on Networking*, 15(2):267–280, 2007.

[5] G. Di Battista, T. Refice, M. Rimondini. How to extract BGP peering information from the Internet Routing Registry. In *MineNet'2006*, pp. 317–322. ACM, 2006.

[6] X. A. Dimitriopoulos, D. V. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, K. C. Claffy, G. F. Riley. AS relationships: Inference and validation. *ACM SIGCOMM Computer Communication Review*, 37(1):31–40, 2007.

[7] X. A. Dimitriopoulos, D. V. Krioukov, B. Huffaker, K. C. Claffy, G. F. Riley. Inferring AS relationships: Dead end or lively beginning? In *WEA'05*, LNCS #3503, pp. 113–125. Springer, 2005.

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

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

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

[11] B. Hummel, S. Kosub. Acyclic type-of-relationship problems on the Internet: An experimental analysis. Technical Report TUM-I0709, Fakultät für Informatik, TU München, February 2007.

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

[13] T. Kernen. traceroute.org web site. http://www.traceroute.org.

[14] S. Kosub, M. G. Maaß, H. Täubig. Acyclic type-of-relationship problems on the Internet. In *CAAN'06*, LNCS #4235, pp. 98–111. Springer, 2006.

[15] Merit Network Inc. Internet Routing Registry. http://www.irr.net.

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

[17] RIPE NCC. Routing Information Service (RIS). http://www.ripe.net/ris/.

[18] G. Siganos, M. Faloutsos. Analyzing BGP policies: Methodology and tool. In *INFOCOM'04*, pp. 1640–1651. IEEE, 2004.

[19] L. Subramanian, S. Agarwal, J. Rexford, R. H. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *INFOCOM'02*, pp. 618–627. IEEE, 2002.

[20] P. Traina, R. Chandra, T. Li. BGP community attribute. RFC 1997, The Internet Society, 1996.

[21] University of Oregon. Route Views project page. http://www.routeviews.org.

[22] I. van Beijnum. *BGP*. O'Reilly, 2002.

[23] J. Xia, L. Gao. On the evaluation of AS relationship inferences. In *Globecom'04*, vol. 3, pp. 1373–1377. IEEE, 2004.