Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/080412

Efficient Inference of AS-Level Paths in the Internet

Qixin Gao[1], *Feng Wang*[2] *and Lixin Gao*[3,*]

[1] Department of Computer Science, Northeastern University, Qinhuangdao, China
[2] School of Engineering and Computational Science, Liberty University, Lynchburg, VA 24502, USA
[3] Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003, USA

Received: 13 Jul. 2013, Revised: 15 Oct. 2013, Accepted: 16 Oct. 2013
Published online: 1 Jul. 2014

Abstract: Routing protocols maintain connectivity of Internet routers and hosts and determine the path that a packet traverses. Inferring Internet paths is critical for evaluating the performance of Internet applications and services, such as cloud services and content delivery. Unlike intra-domain routing protocols, which typically use the shortest paths, inte-domain routing protocol apply local policies for selecting routes/paths and propagating routing information. These routing policies are typically constrained by contractual commercial agreements between ASes. It is well-known that routing policies can impact the AS path that an AS may select for delivering a packet. Thus, the ability to infer the AS-level paths is critical to evaluate the impact of routing policies on the performance of Internet applications and services. In addition, inferring AS-level paths is also important for content providers, such as Google and Amazon, to determine routing policies to ensure small latency in delivering content. However, inferring AS-level paths is challenging. Internet path selection largely depends on routing policies, which in turn are determined independently by network administrators and are considered as confidential information. In this paper, we present three common routing policies in the Internet and formulate the problem of inferring routing policy conforming AS-level paths. We present efficient algorithms for inferring the Internet AS-level paths. The algorithms are proved to be optimal in terms of the ability of derive the policy-conforming AS-level paths. We further quantify the efficiency of these algorithms.

Keywords: BGP, policy-conforming path, AS-level path, algorithm

1 Introduction

With the continued growth of the Internet, large-scale Internet applications and services, such as cloud computing, on-line social applications, and content delivery, are widely deployed. The performance of those applications and services depends on the underlying routing protocols and routing policies. Routing within an Autonomous System (AS) is controlled by intra-domain protocols such as OSPF (Open Shortest Path First) and IS-IS (Intermediate System to Intermediate System), while routing between ASes is exchanged by an inter-domain routing protocol. Border Gateway Protocol (BGP) [20] is an inter-domain routing protocol that allows Autonomous Systems (ASes) to apply local policies for selecting routes and propagating routing information. These routing policies are typically constrained by contractual commercial agreements between ASes. It is well known that routing policies can impact the AS path that an AS may select for delivering a packet. A route in the Internet may take a longer AS path than the shortest AS path due to routing policies [18,19,7].

The ability to infer/predict the AS-level paths is critical to evaluate the impact of routing policies on the performance of Internet applications and services. However, inferring AS-level paths is not challenging. First, Internet path selection largely depends on routing policies, which in turn are defined independently by network operators in each individual AS and are considered as confidential information. Second, using traceroute probing to infer a large set of end-to-end paths is resource consuming. In addition, access to a large collection of hosts is challenging. Most of the studies on the Internet paths [5] focus on router-level path or on inferring AS-level paths from traceroute [6,7,8]. These studies are limited in the scope since traceroute data is typically collected from only a few vantage points.

In this paper, we present three common routing policies and formulate the problem of inferring AS-level paths in the Internet. In particular, we exploit the BGP routing tables from several vantage points, e.g.,

[*] Corresponding author e-mail: lgao@ecs.umass.edu

RouteViews [16] and the RIPE RIS [17], to infer the AS relationships. Based on the AS relationships, we present three common properties of the routing policies. First, it is typical that an AS does not transit traffic between its providers or peers. Second, it is common that an AS prefers its customer route over its provider or peer routes. Third, it is common that an AS prefers its peer routes over its provider routes. Based on these three properties, we formulate the problems of deriving the shortest path that conforms to these properties. We prove that it is possible to infer the shortest AS-level paths that conform to routing policy properties. Finally, we present several algorithms to infer the AS-level paths that conform to the three properties, and quantify the efficiency of the algorithms.

The remainder of the paper is structured as follows. Section 2 presents an overview of AS-level topology, commercial agreements, and common routing policies. In Section 3, we present an algorithm that computes the paths that conform the no-valley and prefer-customer routing policy, and an algorithm that conforms to no-valley, prefer-customer, and prefer-peer-over-provider routing policy. Section 4 presents an algorithm that computes the paths that conform to the no-valley routing policy. In Section 5, we review the related work. We conclude the paper in Section 6 with a summary.

2 Background

In this section, we first review the AS-level topology background and focus on annotating the topology with commercial relationships. We then describe routing policies commonly deployed in the Internet.

2.1 Internet Topology and Commercial Relationships

The Internet consists of a large collection of hosts interconnected by networks of links and routers, which is partitioned into thousands of autonomous systems (ASes). An AS has its own routers and routing policies, and connects to other ASes to exchange traffic with remote hosts. A router typically has very detailed knowledge of the topology within its AS, and limited reachability information about other ASes. Figure 1 shows an example of interconnectivity between ASes in the Internet.

Since we mainly concern AS-level paths in this paper, we model the connectivity between ASes in the Internet using an AS graph $G = (V,E)$, where the node set V consists of ASes and the edge set E consists of AS pairs that exchange traffic with each other. Note that the edges of AS graph represent logical connections between ASes and do not represent the form of the physical connection. The logical connection indicates that the two ASes exchange traffic. Such traffic exchange can occur at

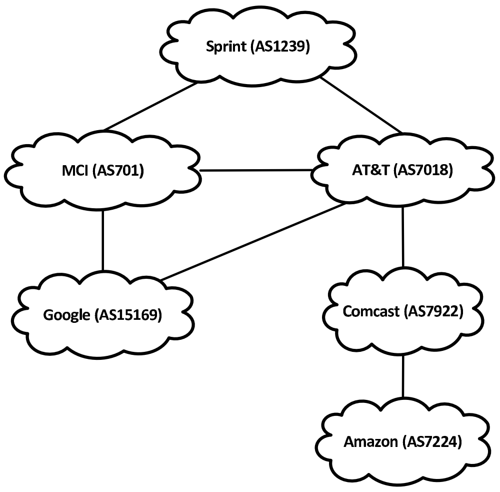


Fig. 1: *An AS graph representing AS-level connectivity*

multiple physical connections such as at a public exchange point or directed connected link. Further the physical connection at an exchange point does not necessarily mean a logical connection. Each AS in the Internet is represented by a 16-bit AS number, which brings to a total of 65,536 possible ASes. In 2013, the Internet has at least 43,000 ASes in use.

Routing policies are constrained by the commercial contractual agreements negotiated between administrative domain pairs. These contractual agreements include *customer-provider* and *peering*. A customer pays its provider for connectivity to the rest of the Internet. A pair of peers agree to exchange traffic between their respective customers free of charge. A mutual-transit agreement allows a pair of administrative domains to provide connectivity to the rest of the Internet for each other. This mutual-transit agreement is typically between two small administrative domains such as local ISPs who are located close to each other and who can not afford additional Internet services for better connectivity.

AS relationships are the key in determining AS paths. In order to represent the relationships between ASes, we use an *annotated AS graph* – a partially directed graph whose nodes represent ASes and whose edges are classified into *provider-to-customer*, *customer-to-provider*, and *peer-to-peer* edges. Furthermore, only edges between providers and customers in an annotated AS graph are directed. Figure 2 shows an example of an annotated AS graph. Note that when traversing an edge from a provider to a customer in a path, we refer to the edge as a *provider-to-customer edge*. When traversing an edge from a customer to a provider, we refer to the edge as a *customer-to-provider edge*. We call the edge between two ASes that have a peer-to-peer relationship a *peer-to-peer edge*. When an AS pair (u,v) has a provider-to-customer relationship, this implies that (v,u) has a customer-to-provider relationship. Throughout this

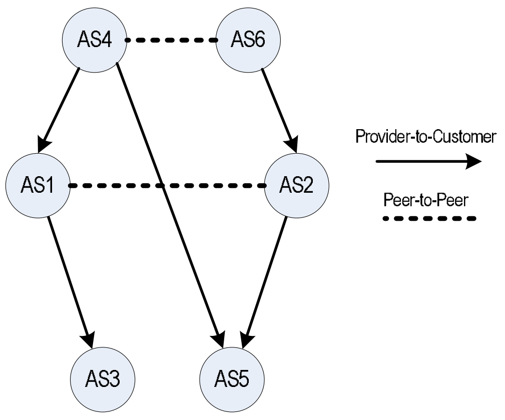


Fig. 2: *An Annotated AS graph representing contractual relationships between connected ASes*

paper, we use the term “provider-to-customer relationship” to imply both provider-to-customer and customer-to-provider relationships whenever it is clear.

Furthermore, we assume that provider-to-customer relationships are hierarchical. That is, there is no cycle among the provider-customer relationships. More formally, consider that the annotated AS graph consisting of only provider-to-customer edges, the graph is a directed acyclic graph, *i.e.*, there is no cycle in the graph. This is a reasonable assumption in practice since it is unlikely that an AS’s indirect provider is a customer of the AS, since an AS is typically a customer of an AS with a larger geographic coverage. Even in the presence of complex AS relationships [9], it is possible that provider-customer relationships are hierarchical for a specific destination.

2.2 Routing Policy

Routing policies typically conform to the commercial relationships between ASes. A customer pays its provider for connectivity to the rest of the Internet. A pair of peers agree to exchange traffic between their respective customers free of charge. A mutual-transit agreement allows a pair of administrative domains to provide connectivity to the rest of the Internet for each other. The commercial contractual relationships between ASes translate into the export rule that an AS does not transit traffic between two of its providers and peers. Formally, we define $customer(a)$, $peer(a)$, and $provider(a)$ as the set of customers, peers, and providers of a, respectively. We classify the set of routes in an AS into customer, provider, and peer routes. A route r of AS u is a *customer (provider, or peer) route* if the first consecutive AS pair in $r.as\_path$ has a provider-to-customer (customer-to-provider, or peer-to-peer) relationship. Given a route r, function $first(r)$ is used to return the first consecutive AS pair in the route. More precisely, let $r.as\_path = (u_1,u_2,\ldots,u_n)$. If $first(r.as) = (u_1,u_2)$ is a provider-to-customer (customer-to-provider or peer-to-peer) edge, then r is a customer (provider or peer) route. An AS selectively provides transit services for its neighboring ASes. The selective export rule translates into *no-valley* routing policy. Intuitively, if we image that a provider is always above its customers and two peering ASes are at the same level, then once an AS path goes down or remains at the same level, it does not go up or remain at the same level.

Furthermore, a route r is classified as a customer route of a if $first(r.as\_path) \in customer(a)$, a private-peer route if $first(r.as\_path) \in peer(a)$, or a provider route if $first(r.as\_path) \in provider(a)$. The AS relationships translate into the following rules that govern BGP export policies [10,11]; we refer to these rules as the *selective export rules*.

– *Exporting to a provider:* In exchanging routing information with a provider, an AS can export its routes and its customer routes, but *usually does not export* its provider or peer routes. That is, an AS does *not* provide transit services for its provider.
– *Exporting to a customer:* In exchanging routing information with a customer, an AS can export its routes and its customer routes, as well as its provider and peer routes. That is, an AS *does* provide transit services for its customers.
– *Exporting to a peer:* In exchanging routing information with a peer, an AS can export its routes and its customer routes, but *usually does not export* its provider or peer routes. That is, an AS does *not* provide transit services.

As a result of the above export polices, paths received by an AS have the *no-valley property*. In a no-valley path, after traversing a provider-to-customer or peer-to-peer edge, it can not traverse a customer-to-provider or peer-to-peer edge. In other words, after traversing a provider-to-customer or peer-to-peer edge, the AS path must traverse provider-to-customer edges. That is, in a no-valley AS path $(u_1,u_2,\ldots,u_n)$, there is i such that $0 \le i < n+1$ and for all $0 < j < i$, (u_j,u_{j+1}) is a customer-to-provider edge, (u_j,u_{j+1}) must be a provider-to-customer edge for any $i+1 < j < n$, and (u_{i+1},u_{i+2}) can be either a peer-to-peer or provider-to-customer edge. For example, in Figure 2, AS paths (1, 4, 6, 2) and (1, 4, 5) are no-valley paths while as_path (4, 5, 2) and (4, 1, 2, 6) are not no-valley paths. Note that the selective export rule ensures that BGP routing table entries contain only no-valley AS paths. For example, if AS path (1, 4, 3) appears in a BGP routing table, then AS 4 exports its provider route (3) to its provider AS 1. This violates the selective export rule.

In addition to the no-valley routing policy, an AS typically chooses a customer route over a route via a provider or peer since an ISP does not have to pay its customer to carry traffic or maintain a traffic volume ratio between the traffic from and to a peer. In addition to the prefer-customer property, an AS might choose a peer route over a provider route since an AS has to pay for the traffic its provider carries for it. Note that these import polices do *not* restrict the preference among customer routes or among provider or peer routes, which provides ISPs with significant flexibility in selecting local policies. Formally, we have the following import policies for AS a:

– *Prefer-customer:* If $first(r_1.as\_path) \in customer(a)$ and $first(r_2.as\_path) \in peer(a) \cup provider(a)$, then $r_1.loc\_pref > r_2.loc\_pref$.
– *Prefer-peer-over-provider:* If $first(r_1.as\_path) \in peer(a)$ and $first(r_2.as\_path) \in provider(a)$, then $r_1.loc\_pref > r_2.loc\_pref$.

Note that each AS has economic incentive to follow the import routing policies, since traffic through a provider route will lead to payment to the provider. Further, these routing policy guidelines can ensure routing stability of the global Internet [12].

Based on import policies, each AS selects its best route among all received routes and announced its best route to neighbors based on its export policies. The route selection process determines the best route using the local-preference attribute first. It chooses the route with the highest local preference. If there is a tie, it should choose a route with the shortest AS path length. Note that this is a simplified version of the BGP decision process since we have not considered other possible attribute such as MED or IGP weight. However, for the sake of deriving possible AS-level paths, this is the first order approximation to estimate AS-level paths.

2.3 Policy-conforming Paths

Based on the analysis of possible routing policies, we have the following three common routing policies.

– *No-Valley Routing Policy*: each AS exports all routes that follow the selective export rule. Furthermore, each AS sets local preference for all routes to be the same. That is, each AS chooses an AS path that is the shortest among received paths.
– *No-Valley-and-Prefer-Customer (No-Valley-PC) Routing Policy*: each AS exports all routes that follow the selective export rule. Furthermore, each AS follow the prefer-customer import policy, and the local-preference of all customer routes is the same, and the local-preference of all provider and peer routes is the same.
– *No-Valley-and-Prefer-Customer-and-Peer-Over-Provider (No-Valley-PCPoP) Routing Policy*: each AS exports all routes that follow the selective export rule. Furthermore, each AS follows the prefer-customer and prefer-peer-over-provider import policy, and the local-preference of all customer routes is the same, the local-preference of all peer routes is the same, and the local-preference of all provider routes is the same.

Note that these three routing policies are the simplest routing policies that conform to routing policy guidelines. In reality, an AS can specify a diverse set of routing policies including its preference on customer (peer or provider) routes and filtering policies. For example, an AS can specify that it prefers routes through one of its neighbors over others. As we will see later our algorithms for No-Valley-PC and No-Valley-PCPoP Routing Policies can be expanded to more routing policies that conform to the guidelines.

Based on these common routing policies, we define the following policy-conforming paths:

– *No-Valley paths*: The AS paths derived from that all ASes follow the no-valley routing policies.
– *No-Valley-PC paths*: The AS paths derived from that all ASes follow the No-Valley-PC routing policy.
– *No-Valley-PCPoP paths*: The AS paths derived from that all ASes follow the No-Valley-PCPoP routing policy.

We note that these policy-conforming paths are derived from distributed routing decision processes by all ASes. As a result, No-Valley path might be different from the shortest path among all no-valley paths. We will derive the paths derived from distributed routing decision based on these routing policies. Further, each AS chooses only one best path for a destination. However, we do not enforce any tie breaking mechanism in route decision process. As a result, it is possible to have multiple policy-conforming paths for a destination. Therefore, policy-conforming paths is a set of possible paths that can be the best route. Throughout this paper, we describe paths relative to a fixed destination d. All our algorithms apply to any destination d.

3 Computing No-Valley-PC and No-Valley-PCPoP Paths

In order to derive policy-conforming paths, we need to classify all paths destined to the same destination address into three classes.

– *Straight path*: a path that contains provider-to-customer edges but does not contain any customer-to-provider or peer-to-peer edge. Note that a straight path can be announced to any neighbor, and therefore can be extended by any edge.
– *Step path*: a path that first traverses a peer-to-peer edge and then traverses zero, one or more provider-to-customer edges. Note that a step path can be announced to a customer only, and therefore can be extended by a customer-to-provider edge only.
– *Arc path*: a path that contains one or several customer-to-provider edges, followed by zero or one peer-to-peer edge, followed by zero, one or several provider-to-customer edges. Note that an arc path can be announced to a customer only and therefore can be extended by a customer-to-provider edge only.

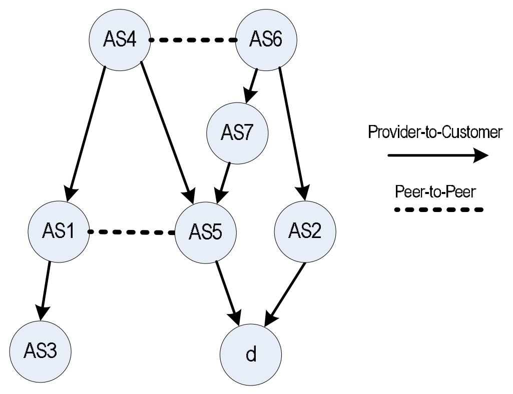


Fig. 3: *An annotated AS graph for illustrating arc, step, and straight paths to destination d*

For example, in Figure 3, AS paths (4,5,d), (6, 7, 5, d) and (6, 2, d) are the straight paths to reach destination d. AS path (4, 6, 2, d) and (1, 5, d) are the step paths to reach destination d. AS path (3, 1, 4, 6, 2, d), (3, 1, 4, 6, 7, 5, d), (3, 1, 4, 5, d) and (3, 1, 5, d) are the arc paths to reach destination d.

For the ease of exposition, we define two directed graphs with respect to the same destination prefix. G_{down} is the AS graph that consists provider-customer relationships only. It is the DAG where the edge of the graph goes from provider to customer when the traffic towards the destination is forwarded along the edge. G_{up} is the AS graph that consists customer-provider relationships only. It is the DAG where the edge of the graph goes from customer to provider when the traffic towards the destination is forwarded along the edge. Therefore, all the paths in G_{down} are straight paths, and all the paths in G_{up} are arc paths. For example, in Figure 3, G_{down} for d consists of AS 2, 4, 5, 6, and 7, while G_{up} consists of AS 3, 1, and 4.

For the three types of policy-conforming paths, we have the following lemmas:

Lemma 1. *If there is a straight path from an AS to d, the AS will receive a straight path whose length is the shortest among all straight paths.*

Proof. We prove by induction. Clearly, the lemma holds for d. In order to enumerate all ASes in G, we perform a topological sort on G_{up}. Suppose the lemma holds for all ASes before u in the topological order. We now show that it holds for AS u. Since there is a straight path from u to d, u has at least one customer. For each $v \in customer(u)$, v receives a shortest straight path. As a result, u will receive a shortest straight path from its customers according to the export policy.

Lemma 2. *If there is a step path from an AS to d, the AS will receive a shortest step path.*

Proof. Since there is a step path from the AS u to d, u has at least one peer. Further, for each of u's peers, v, which has a straight path, based on Lemma 1, v receives a shortest straight path. We know that v will announce the path to u according the export policy. Therefore, u will receive those paths. One of them must be the shortest step path for u since all received paths are shortest straight paths from u's peer.

Lemma 3. *If there is an arc path from an AS to d, the AS will receive a shortest arc path.*

Proof. We prove by induction. In order to enumerate all ASes in G, we perform a topological sort on G_{down}. It clearly holds for those ASes who do not have providers in G_{down}, since those ASes do not have an arc path to d. Suppose the lemma holds for all ASes before u in the topological order. We now show that it holds for AS u. For each $v \in provider(u)$, v receives a shortest straight, arc or step path. As a result, u will receive a shortest arc path from its providers according to the export policy.

Combining the above three lemmas, we have the following two theorems:

Theorem 1. *If an AS has a straight path to d, then its No-Valley-PC paths are the shortest paths among all straight paths. Otherwise, if the AS has an arc or step path to d, its No-Valley-PC paths are shortest paths among all step and arc paths.*

Theorem 2. *If an AS has a straight path to d, then its No-Valley-PCPoP paths are the shortest paths among all straight paths. Otherwise, if the AS has a step path to d, then No-Valley-PCPoP paths are the shortest paths among all step paths. If the AS has neither a step or straight path to d, then No-Valley-PCPoP paths are the shortest paths among all arc paths.*

In the following subsections, we first show how to compute shortest straight, step, and arc paths. In later subsections, we illustrate how to compute policy-conforming paths for common routing policies.

3.1 Shortest Straight Paths

In order to compute the shortest straight path, we focus on G_{down} and perform the shortest path search from d to all nodes. We find the shortest path from d to all other nodes by performing breadth-first search on G_{down} from d. This will give us shortest straight paths from all nodes to d by reversing the paths discovered. The algorithm is shown in Algorithm 1. Since this algorithm is based on breadth-first search, we can easily prove the following Lemma.

Lemma 4. *Algorithm 1 derives the shortest straight paths for each node.*

For example, in Figure 3, the shortest straight path at AS 6 is the AS path (6, 2, d), and the shortest straight path at AS 4 is (4, 5, d).

Algorithm 1: Compute shortest straight paths

```
input  : Annotated AS graph $G$ and destination node $d$
output: Shortest straight path set from all nodes to node $d$
1  Set the straight path set of node $d$ to be a path with zero
   length ;
2  Set the straight path set of all nodes other than $d$ to be
   empty ;
3  Perform breadth-first search from $d$ on directed graph
   $G_{down}$ ;
4  for each level $i$ of the breath-first search do
5      for each node $u$ in level $i$ of the breadth-first search
       do
6          Let $u$'s parent set be the set of nodes in level $i-1$
           that connect to $u$ ;
7          for each of $u$'s parents, $v$ do
8              append each of shortest straight paths of v
               with $u$ and add the path to shortest straight
               path set ;
9          end
10     end
11 end
```

3.2 Shortest Step Paths

Once we derive the shortest straight paths, the shortest step paths of node u will be the shortest straight paths of all peers of u appended with u. Algorithm 2 presents the details of deriving the shortest step path. We also can easily prove Lemma 5 based on Algorithm 2.

Algorithm 2: Compute step paths

```
input  : Annotated AS graph $G$ and destination node $d$
         and shortest straight paths for all nodes
output: Shortest step paths from all nodes to node $d$
1  Set the shortest step path of node $d$ to be a path of length
   zero ;
2  for each node $u$ do
3      for each of $u$'s peer, $v$ do
4          if $v$'s shortest straight path has shortest length
           among all of $u$'s peer then
5              append each of $v$'s shortest straight paths to $u$
               and add the path to $u$'s shortest step path set ;
6          end
7      end
8  end
```

Lemma 5. *Algorithm 2 derives the shortest step path for each node.*

3.3 Shortest Arc Paths

Once we derive the shortest straight and arc paths, the shortest arc paths of node u will be the shortest arc, step, or straight paths of u's providers appended with u. Algorithm 3 presents the details of deriving the shortest arc paths.

Algorithm 3: Compute Arc Path

```
input  : Annotated AS graph $G$ and destination node $d$,
         straight, step and arc paths for all nodes
output: Shortest arc paths for all nodes to node $d$
1  Sort the nodes in the topological order of the DAG given
   by $G_{up}$;
2  for each node, $u$, in the topological order do
3      if $u$ does not have any provider then
4          Assign $u$'s shortest arc path set to be empty ;
5      end
6      else
7          for each provider, $v$, of $u$ do
8              if $v$'s shortest arc, step or straight path is the
               shortest among all of $u$'s providers then
9                  Append $v$'s shortest arc, step or straight
                   paths with $u$ and add the path to the $u$'s
                   shortest arc path set ;
10             end
11         end
12     end
13 end
```

Lemma 6. *Algorithm 3 derives the shortest arc path for each node.*

Proof. The algorithm extends all step and straight paths to arc paths. Since we do so in the topologically sorting order of G_{up}, each node uses the shortest arc path possible from all of its providers to derive the shortest arc path. Therefore, we can show by induction, we derive the shortest arc path for all nodes.

3.4 No-Valley-PC Paths

Once we derive the shortest straight, step and arc paths, we can derive No-Valley-PC path to be shortest straight paths if such a path exists. Otherwise, No-Valley-PC paths are shortest paths among arc and step paths. The algorithm is presented as follows.

Let us present the detailed description of the algorithm. Lines 1–3 are the phases of deriving the shortest straight, step and arc paths by using Algorithm 1, 2, and 3, respectively. Lines 4–22 are the main loop of the algorithm. At each execution of the loop as long as there is one or more straight paths, use the shortest straight path as the No-Valley-PC path set (line 6–8). Next, if a node does not have any straight path, it is checked if the node has any step path and arc path. If so, the shortest step path is used as the No-Valley-PC path set (line 10–12). If the arc path and the step path have the same length, both the

Algorithm 4: Compute No-Valley-PC Paths

```
input  : Annotated AS graph $G$ and destination node $d$
output: No-Valley-PC paths for all nodes to node $d$
1  Phase 1: Compute the shortest straight path ;
2  Phase 2: Compute the shortest step path ;
3  Phase 3: Compute the shortest arc path ;
4  Phase 4: Compute No-Valley-PC path ;
5  for each node do
6      if its straight path set is not empty then
7          Set its No-Valley-PC paths to be shortest straight path set ;
8      end
9      else
10         if shortest step path length $\leq$ shortest arc path length then
11             Set its No-Valley-PC path set to be the shortest step path set ;
12         end
13         else
14             if shortest step path length $=$ shortest arc path length then
15                 Set its No-Valley-PC path set to be union of the shortest arc path set and shortest step path set ;
16             end
17             else
18                 Set its No-Valley-PC path set to be the shortest arc path set ;
19             end
20         end
21     end
22 end
```

paths are used as the No-Valley-PC path set (line 14–16). Otherwise, the shortest arc path is used as the No-Valley-PC path set (line 18–19).

Based on Theorem 1, we have the following theorem.

Theorem 3. *Algorithm 4 derives No-Valley-PC paths.*

3.5 No-Valley-PCPoP Paths

Similar to the algorithm for computing No-Valley-PC paths, the algorithm for computing No-Valley-PCPoP path using straight and step paths first. If such a path does not exist, we derive No-Valley-PCPoP paths based on step and arc paths. The key difference here is that No-Valley-PCPoP paths are the shortest step if such a path exists. The algorithm is presented as follows.

We present the detailed description of the algorithm. Lines 1–3 are the phases of deriving the shortest straight, step and arc paths by using Algorithm 1, 2, and 3, respectively. Lines 4–17 are the main loop of the algorithm. At each execution of the loop a node's the shortest straight path is used as the No-Valley-PC path set (line 6–8). Next, if a node does not have any straight path,

Algorithm 5: Compute No-Valley-PCPoP Paths

```
input  : Annotated AS graph $G$ and destination node $d$
output: No-Valley-PCPoP path set for all nodes to node $d$
1  Phase 1: Compute the shortest straight path ;
2  Phase 2: Compute the shortest step path ;
3  Phase 3: Compute the shortest arc path ;
4  Phase 4: Compute No-Valley-PCPoP path ;
5  As long as there is a straight path, use it. for each node do
6      if its straight path set is not empty then
7          Set its No-Valley-PCPoP path set to be shortest straight path set ;
8      end
9      else
10         if its step path set is not empty then
11             Set its No-Valley-PCPoP path set to be shortest step path set ;
12         end
13         else
14             Set its No-Valley-PCPoP path set to be shortest arc path set ;
15         end
16     end
17 end
```

it is checked if the node has a step path. If so, the shortest step path is used as the No-Valley-PC path set (line 10–12). Otherwise, the shortest arc path is used as the No-Valley-PC path set (line 14–16).

Based on Theorem 2, we have the following theorem.

Theorem 4. *Algorithm 5 derives the shortest No-Valley-PCPoP path.*

Proof. It is clear that the derived path is no-valley and prefer-customer from Theorem 3. Prefer-peer-over-provider is reflected in the fact that No-Valley-PCPoP path is set to the arc path only if there is no straight or step path.

Both the algorithms of computing No-Valley-PC paths and No-Valley-PCPoP paths traverse each edge of the annotated AS graph at most twice. Therefore, it takes $O(E+N)$ time to derive the desired paths from all ASes to a destination AS, where N and E are the number of ASes and edges, respectively, in the annotated AS graph. For all pair AS paths, it takes $O((E+N)N)$ time to compute AS paths from all ASes to all destination ASes.

4 Computing No-Valley Paths

We first show the algorithm that derives the shortest path among all arc, step and straight paths. We note that the No-Valley path is derived from distributed routing decision process given the no-valley routing policy. As a result, No-Valley shortest path might be different from the shortest paths among all arc, step and straight paths. For

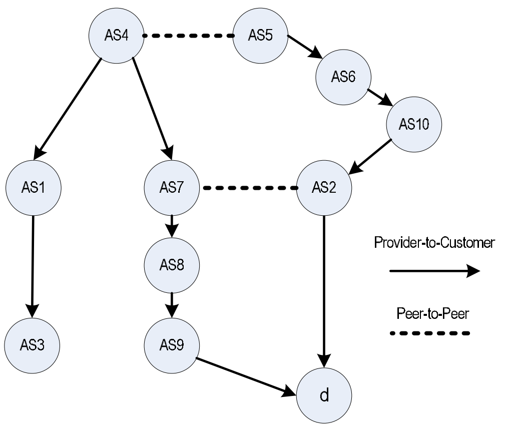


Fig. 4: *An AS graph showing that the shortest no-valley path is not necessarily the same path derived from distributed routing decision with the no-valley shortest path routing policy.*

the sake of the contrast, we define *Global-No-Valley paths* to be the shortest paths among arc, step and straight paths. Algorithm 6 is used to derive the Global-No-Valley path for each node.

Algorithm 6: Compute Global-No-Valley Paths

```
input  : Annotated AS graph $G$ and destination node $d$
output: Global-No-valley paths for all nodes to node $d$
1 Phase 1: Compute the shortest straight path ;
2 Phase 2: Compute the shortest step path ;
3 Phase 3: Compute the shortest arc path ;
4 Phase 4: Compute No-Valley path ;
5 for each node do
6     Set global-no-valley path set to be the shortest among
      arc, straight, step path set of the node.
7 end
```

While we have shown the algorithm for computing the shortest arc, step and straight paths, the derived algorithm might not be the no-valley paths derived from distributed decision process. For example, in Figure 4, the distributed decision process will give a no-valley path of $(4,5,6,10,2,d)$ for AS 4 while the Global-No-Valley path is $(4,7,8,9,d)$, since the no-valley path for AS 7 is $(7,2,d)$ instead of $(7,8,9,d)$. Thus, $(7,8,9,d)$ is not announced to AS 4. Therefore, the no-valley path of AS 4 is $(4,5,6,10,2,d)$.

We present an algorithm for computing the no-valley path for each node. Since each node does not have the prefer-customer routing policy, essentially, it will choose the shortest path among the paths received from its neighbors. Despite the fact that each node chooses the path among the paths received, it still needs to keep track of the kind (arc, step, or straight) of the path it chooses, since it will propagate the chosen path in a way that is consistent with the type of the path. That is, it announces all paths to customers, only straight paths to providers and peers.

In order to ensure that each node always chooses the shortest path received, we use the similar mechanism as the Dijkstra's algorithm. Each node keeps track all the paths received from neighbors and their corresponding type. We select nodes to finalize their no-valley paths based on the length of their chosen paths. The node with shortest chosen path is selected. Once selected, the node propagate its paths to all its neighbors based on the type of the path. In order to differ a no-valley path derived from distributed routing decision from a Global-No-Valley path, we call the no-valley path derived from distributed routing decision as a *No-Valley-Consistent* path. We present the algorithm in Algorithm 7. Note that although we present the algorithms for computing the no-valley path length, our algorithms can be easily extended to derive no-valley path set.

We present the detailed description of Algorithm 7. Lines 1–3 are the initialization phase. First, a given destination d is selected. Lines 6–35 are the main loop of the algorithm. At each execution of the loop a node's no-valley path is propagated to its neighbors. If the no-valley path is a straight path, the path is propagated to the node's all neighbors and the neighbor chooses the shortest path among its arc, step and straight paths (line 9–21). If the no-valley path is a step or an arc path, the path is propagated only to the node's customers, and the customers choose the shortest path among their arc, step and straight paths (line 22–35).

Theorem 5. *Algorithm 7 derives no-valley paths.*

Proof. First of all, the derived path is a no-valley path since we use only straight, step, and arc path to derive the no-valley path. Second, we can show by induction that each selected node gets the no-valley path. Assume all nodes selected before this node get the no-valley paths. Then this node gets the shortest path among all received paths at this point. This path is the no-valley path for the node.

The above algorithm traverses each edge of the annotated AS graph at most twice. Further, selecting nodes with shortest arc, step and straight path requires $N\log N$ time (if we use a heap to store the information of each node path length), where N is the number of nodes in the annotated AS graph. Therefore, it takes $O(E+N\log N)$ time to compute no-valley paths from all ASes to a destination AS, where N and E are the number of ASes and edges, respectively, in the annotated AS graph. For all pair AS paths, it takes $O(NE+N^2\log N)$ time to compute AS paths from all ASes to all destination ASes.

Algorithm 7: Compute No-Valley Path

```
   input  : Annotated AS graph $G$ and destination node $d$
   output: No-valley path length for all nodes to node $d$
1  Initialization ;
2  Set the straight, arc, step, and no-valley path of node $d$ to
   be an empty path ;
3  Set the straight, arc, step, and no-valley path length of all
   nodes other than node $d$ to be infinity ;
4  Make node $d$ to be the selected node and set its no-valley
   path to be straight type;
5  Propagate paths to neighbors ;
6  while the selected node, $u$'s no-valley path length is not
   infinity do
7      Update the path length of neighbors of $u$ as follows ;
8      if $u$'s no-valley path is of type straight then
9          for each provider of $u$ do
10             set the straight path length of the provider to
               be the min of no-valley path length of $u$ +1
               and the straight path length of the provider ;
11             set the no-valley path of the provider to be
               the min of its arc, straight, step paths ;
12         end
13         for each peer of $u$ do
14             set the step path length of the peer to be the
               min of no-valley path length of $u$ +1 and the
               step path length of the peer ;
15             set the no-valley path of the peer to be the
               min of its arc, straight, step paths ;
16         end
17         for each customer of $u$ do
18             set the arc path length of the customer to be
               the min of no-valley path length of $u$ +1 and
               the arc path length of the customer ;
19             set the no-valley path of the customer to be
               the min of its arc, straight, step paths ;
20         end
21     end
22     if $u$' no-valley path is of type step then
23         for each customer of $u$ do
24             set the arc path length of the customer to be
               the min of no-valley path length of $u$ +1 and
               the arc path length of the customer ;
25             set the no-valley path of the customer to be
               the min of its arc, straight, step paths ;
26         end
27     end
28     if $u$'s no-valley path is of type arc then
29         for each customer of $u$ do
30             set the arc path length of the customer to be
               the min of no-valley path length of $u$ +1 and
               the arc path length of the customer ;
31             set the no-valley path of the customer to be
               the min of its arc, straight, step paths ;
32         end
33     end
34     Select a node with shortest no-valley path length
       among nodes that have not been selected ;
35 end
```

5 Related Work

Several work have been focused on inferring AS-level paths. To the best of our knowledge, the most related to our work is [7], Mao *et al.* investigate the feasibility of inferring AS-level path without direct access to end-points. The AS-level paths are inferred by finding the shortest policy paths in an AS graph. The AS graph is built based on BGP tables from multiple vantage points and router-level paths from traceroute servers. And then, the AS paths is obtained by inferring the shortest policy paths in the AS graph. Their algorithm to infer the policy path is similar to ours. But we propose several algorithms to infer AS level paths under different routing policies. Therefore, our study complements their work.

In [13], Sobrinho *et al.* presented an algebraic theory to understand the minimum number of links in a network whose failure causes the network to become disconnected. They investigate the connectivity provided by route-vector protocols in the presence of customer-provider and peer-peer routing policies. Even though our work does not consider failures in a network, the proposed algorithms can be used to infer the minimum number of AS paths in a fault tolerance scenario. Feamster *et.al.* [14] studied routing protocol stability under certain rankings and filters that are commonly used in practice. They prove that ranking routes and configuring filters autonomously may not ensure routing stability so that a stable path assignment requires ASes to rank routes based on AS-path lengths. Different with their work, our work is based on the assumption that the underlying routing is stable.

Some work at discovering AS-level topology are based on using traceroute data. For example, in [15], Mao *et.al.* present methods to solve the challenge of mapping of an IP address to the correct AS number. Their techniques can improve the mapping of IP addresses to corresponding ASes, which could be used as an AS-level traceroute tool. In this paper, we infer AS level paths. We plan to use traceroute data to infer AS level paths as our future work.

6 Discussion and conclusion

We describe common routing policies in the Internet and formulate the problem of computing the paths that conform to these routing policies. ISPs have incentive to conform to the two routing policies described and therefore it is important to understand how to compute the routing paths that conform to these routing policies. We present efficient algorithms for these computations and show the complexity of these algorithms. We show that our algorithms are efficient for a large AS graph of the Internet.

We notice that our algorithms to derive the policy-conforming paths require prior knowledge about the Internet topology on AS level and AS relationships.

Just like other methods to infer AS level paths, such as RouteScope [7], our method is a static method. Therefore, they are intended to predict the potential AS path rather than infering real-time AS level paths. We also note that The AS-level paths and AS relationships are relatively stable and do not change significantly comparing with BGP routing dynamics.

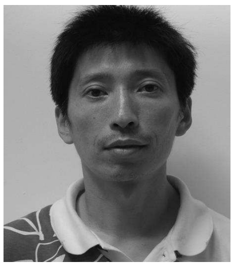
Inferring AS path has an important implication on AS path inflation occurring in the Internet. Here, we give one example to highlight that AS path inflation can impact the content-driven architecture. Today, content traffic, such as YouTube video, becomes the dominant type of traffic in today's Internet, the underlying internet architecture is required to deliver the content efficiently. Large-scale content providers, such as Google, Yahoo and Facebook move the content closer to the end user to reduce the origin content server and improve performance for clients. However, moving content physically close to an end user does not necessarily mean the paths taken by content traffic are the shortest paths. Based on our analysis in AS path inflation, we believe that the paths taken by an end user to pull the content could be longer than we expect. Therefore, the content placement selection need to take policy-conforming paths into consideration.

Acknowledgement

This work is partially supported by the NSF grant CNS-0917078. The authors are grateful to the anonymous referees for helpful comments and constructive feedbacks.

References

[1] J. W. Stewart, BGP4: Inter-Domain Routing in the Internet. Addison-Wesley, (1999).
[2] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet routing convergence, in Proc. ACM SIGCOMM, (2000).
[3] C. Labovitz, R. Wattenhofer, S. Venkatachary, and A. Ahuja, "The impact of Internet policy and topology on delayed routing convergence, in Proc. IEEE INFOCOM, (2001).
[4] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the Internet topology, in Proc. ACM SIGCOMM, (1999).
[5] H. Tangmunarunkit, R. Govindan, D. Estrin, and S. Shenker, "The impact of routing policy on Internet paths, in Proc. IEEE INFOCOM, (2001).
[6] Z. M. Mao, D. Johnson, J. Rexford, J. Wang, and R. Katz, "Scalable and Accurate Identification of AS-level Forwarding Paths, in IEEE INFOCOM, (2004).
[7] Z. Morley, M. Lili, Q. Jia, and W. Y. Zhang, "On AS-level Path Inference, in ACM SIGMETRICS, (2005).
[8] L. Gao and F. Wang, "The extent of as path inflation by routing policies, in Proc. IEEE GLOBAL INTERNET, (2002).
[9] Y. Liao, L. Gao, R. Guerin, and Z.-L. Zhang, "Safe interdomain routing under diverse commercial agreements, IEEE/ACM Trans. Netw., **18**, 1829-1840 (2010).
[10] C. Alaettinoglu, "Scalable router configuration for the Internet, in Proc. IEEE IC3N, (1996).
[11] G. Huston, "Interconnection, peering and settlementsPart II, in Internet Protocol Journal, (1999).
[12] L. Gao and J. Rexford, "A stable Internet routing without global coordination, in Proc. ACM SIGMETRICS, (2000).
[13] J. L. Sobrinho and T. Quelhas, "A theory for the connectivity discovered by routing protocols, IEEE/ACM Trans. Netw., **20**, 677-689 (2012).
[14] N. Feamster, R. Johari, and H. Balakrishnan, "Implications of Autonomy for the Expressiveness of Policy Routing, in ACM SIGCOMM, (Philadelphia, PA), (2005).
[15] Z. M. Mao, J. Rexford, J. Wang, and R. H. Katz, "Towards an accurate as-level traceroute tool, in SIGCOMM, 365-378 (2003).
[16] D. Meyer, "University of Oregon Route Views Project", http://www.routeviews.org/, (2004).
[17] "RIPE RIS, Ripe routing information service", http://www.ripe.net/ris
[18] Hongsuda Tangmunarunkit, Ramesh Govindan, Scott Shenker and Deborah Estrin, "The Impact of Routing Policy on Internet Paths", in INFOCOM, 736-742 (2001).
[19] Neil Spring, Ratul Mahajan and Thomas Anderson, "The causes of path inflation", in SIGCOMM '03, 113–124 (2003).
[20] Y. Rekhter, T. Li and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", Request for Comments, **4271**, (2006).



Qixin Gao received the Ph.D. degree in Institute of Computer Science and Engineering, Northeastern University, Shenyang, China, in 2008. He is currently with Northeastern University at Qinhuangdao, China. His research interests include Internet routing, image processing, and massive data processing.

Feng Wang is an associate professor with the School of Engineering and Computational Sciences at Liberty University. He received his Ph.D degree in Electrical and Computer Engineering at the University of Massachusetts, Amherst. He received his B.E degree from Zhejiang University in China, and M.S. degree from Yanshan University in China. His research interests include network verification, Internet routing, and wireless networks.

Lixin Gao is a professor of Electrical and Computer Engineering at the University of Massachusetts at Amherst. She received her Ph.D. degree in computer science from the University of Massachusetts at Amherst in 1996. Her research interests include social networks, Internet routing, network virtualization and cloud computing. Between May 1999 and January 2000, she was a visiting researcher at AT&T Research Labs and DIMACS. She was an Alfred P. Sloan Fellow between 2003-2005 and received an NSF CAREER Award in 1999. She won the best paper award from IEEE INFOCOM 2010, and the test-of-time award in ACM SIGMETRICS 2010. Her paper in ACM Cloud Computing 2011 was honored with Paper of Distinction. She received the Chancellors Award for Outstanding Accomplishment in Research and Creative Activity in 2010, and is a fellow of ACM and IEEE.