AS Path Inference by Exploiting Known AS Paths

Jian Qiu and Lixin Gao
Dept. of ECE, University of Massachusetts, Amherst, MA 01002
{jqiu, lgao}@ecs.umass.edu

***Abstract*— Inferring AS-level end-to-end paths can be a valuable tool for both network operators and researchers. A widely known technique for inferring end-to-end paths is to perform traceroute from sources to destinations. Unfortunately, traceroute requires the access to source machines and is resource consuming.**

In this paper, we propose two algorithms for AS-level end-to-end path inference. The key idea of our algorithm is to exploit the AS paths appeared in BGP routing tables and infer AS paths based on these known AS paths. In addition, our algorithms infer AS paths on the granularity of destination prefix instead of destination AS. That is, we infer AS paths from any source AS to any destination prefix. This is essential since routing in the Internet is determined based on destination prefixes instead of destination ASs. The validation results show that our algorithm yields accuracy up to 95% for exact match and accuracy up to 97% for path length match. We further extend our algorithm to infer a set of potential AS paths between a source AS and a destination prefix. We find that on average, 86% of inferred AS path sets are accurate in the sense that one of the paths in the set matches the actual AS path. Note that our algorithms require BGP routing tables only and do not require additional data trace or access to either sources or destinations. We also demonstrate that the accuracy of this BGP-based inference approach cannot go beyond 90%.

I. Introduction

The Internet has evolved into the largest man-made distributed system in the world. The selection of end-to-end Internet paths depends on both routing protocols deployed and a large number of network operators involved. Yet, the ability to infer end-to-end Internet paths is essential for network operators as well as network researchers to perform traffic engineering, network diagnosis, and overlay network routing.

A well-known existing tool for inferring end-to-end paths is to perform traceroute from a source host to a destination host. However, traceroute is limited since it requires access to source hosts and it is resource consuming given a large set of paths are to be inferred. Access to a large collection of hosts in the Internet is challenging due to the distributed administration of Internet hosts. Although there have been around 1,000 traceroute/looking glass servers around the world [1], the provided sources are still far too few given the enormous scale and heterogeneity of the Internet.

Alternatively, BGP (Border Gateway Protocol) [2] routing information provides a relatively comprehensive view of AS level topology. Although it is infeasible to query the AS level paths from the BGP routers of the more than 20,000 ASs on the Internet, the public BGP information has been shown to be able to provide relatively complete and up-to-date Internet AS-level topologies [3] and part of routing policies [4]. Such information can be exploited to infer AS level end-to-end paths while no need to access either of the sources or the destinations and perform the resource-consuming probing.

Nevertheless, the BGP-based AS Path inference is not trivial. Internet path selection largely depends on routing policies, which in turn are defined independently by network operators in each individual AS and are seldom publicly available. Mao *et al* [5] are the first to conduct an intensive study on inferring AS paths between any two ASs based on BGP routing information. Their algorithm is based on a new AS relationship inference algorithm, and assumes the shortest AS path routing policy among available paths. They claimed accuracy up to 60% in the sense that one of the inferred shortest paths matches with the actual AS path, and accuracy up to 62% in terms of path length match. They further propose a novel scheme to infer the first AS hop by assuming the access to destination hosts. The evaluation results show that given the first AS hop information, the accuracy of path length match can be improved up to around 70%.

The AS Path inference can benefit a broad spectrum of network applications. For instance, in the overlay networks, knowing the AS Paths between any two nodes helps to find the nearest adjacent peering nodes in the AS-level [6]. Meanwhile, even knowing the length of the AS Path is enough. An example is that [7] shows that the AS Path length change dominates the end-to-end delay performance and can be utilized in the overlay network to choose best relay nodes in terms of delay performance. Also, in [8], the authors need to know the length of AS path from a remote AS to the local prefixes to perform ingress traffic engineering at the local AS. In addition, knowing all possible AS paths from a source AS to a destination prefix is also helpful. In the overlay networks, the overlay links should be disjoint so as to achieve independent underlay link performance [7], [6]. The knowledge of all potential AS Paths between nodes helps to explore the disjoint overlay links.

In this paper, we formulate two problems for the AS path inference. One is the *single AS path inference problem*, which infers a single AS path from an AS to a prefix. The other one is the *potential AS path inference problem*, which infers a set of AS paths from an AS to a prefix. Note that in contrast to [5], we infer AS-level end-to-end paths on the granularity of destination prefix instead of destination AS.

The key idea of our inference algorithms is to exploit known AS paths appeared in BGP routing tables. Using only BGP routing tables, our algorithm can yield accuracy up to 95% and average accuracy around 60% for the exact match. Because the inference of AS path length is also meaningful in some context, we also check the performance in terms of path length match. It shows that our algorithm is able to yield accuracy

up to 97% and average accuracy around 81%. Moreover, as suggested in [5], given the first hop AS, the average accuracy for AS path match can be improved to 78% for the exact match and 88% for path length match.

For the potential AS path inference problem, we find that by incorporating the path set stored at an AS node for a destination prefix, the inferred AS path set contains the actual AS path in 78% of our validation cases. Furthermore, we can extend our algorithm to achieve average accuracy of 86% in the sense that one of the paths in the inferred path set is the actual path although we limit the number of paths in the path set to no more than 10 only. Meanwhile, our experiments demonstrate that it is due to the incompleteness of AS graph derived from BGP routing tables that the inference accuracy cannot be higher than 90%. In this sense, our algorithm is able to capture 95% of the *inferable* AS paths.

We further perform intensive experiments to validate the robustness of our algorithms and examine the impact of the selection of the BGP vantage points, which provide the BGP routing tables for inference, on the accuracy. Besides that, despite the novel features that our algorithms have incorporated, they are still simple, efficient, and able to provide near real-time inference results.

The rest of the paper is organized as follows. The next section summarizes the related work. We introduce the models, formulate the problems and identify the challenges in section 3. Section 4 describes the details of our inference algorithms. In section 5, we first investigate the properties of the data that we will use for evaluation and then show the evaluation results. Finally, section 6 concludes the paper.

II. Related work

The related work in the area of Internet topology inference and discovery are in two folds: some mainly utilize the traceroute-like active measurement tools and the others heavily rely on the passive analysis of BGP routing information. Traceroute is a widely used network diagnostic tool to actively discover the router-level forwarding path between two end hosts. Based on traceroute, a number of sophisticated techniques and tools have been developed to discover the network topology in various granularities. Mercator [9] discovers the Internet router-level topologies. Rocketfuel [10] measures the PoP-level ISP topologies. Skitter [11] provides a relatively complete view of the Internet AS-level topology. Mao *et al* [12], [13] developed tools to discover AS-level forwarding with some systematic IP-to-AS mapping techniques. Internet topology inference based on BGP routing information is another approach. BGP routing tables are widely exploited to construct Internet AS-level topologies to characterize the Internet topology properties. For instance, Faloutsos *et al* [3] unveiled the Power-Laws in the Internet topologies. Gao [4] further modeled the commercial relationships between ASs and several heuristics were proposed to infer the AS relationships [4], [14], [15]. Furthermore, the AS relationships are found to skew the AS paths [16] and classify the ASs into hierarchical tiers [17], [14]. The information in BGP routing updates is also explored. Dimitropoulos *et al* [18] mined the AS path information in BGP updates to discover more complete Internet topologies. Andersen *et al* [19] explored the temporal correlations of the routing updates to find the geographical adjacency of these prefixes.

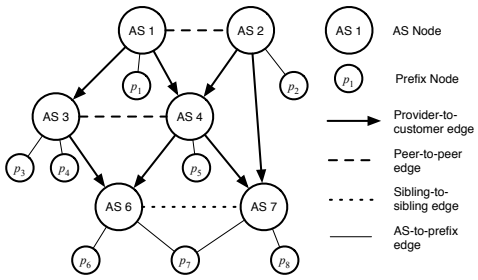


Fig. 1. An AS-prefix graph

III. Models and Problem Formulation

We use a simplified AS-level topology model for the AS path inference. The difference between our model and others is that the topological information of the prefixes is also integrated into the AS topology, which enables our inference algorithm to infer the AS paths specific to destination prefixes instead of destination ASs.

A. Topology Model

The Internet AS topology can be modeled as an *AS graph* $G = (V, E)$ where V is the set of ASs and E is the set of peering sessions between two neighboring ASs. A path in the AS graph is a loop-free AS sequence, i.e. $P = (v_k v_{k-1} \cdots v_1)$ where $v_i \neq v_j$ if $i \neq j$. $|P|$ represents the length of path P, i.e. the number of AS nodes of P. ψ denotes an *empty path* whose length is 0.

In the inter-domain routing system, two neighboring ASs exchange paths according to their commercial agreements [4]. Typically, the relationships can be classified into three categories: provider-to-customer, peer-to-peer and sibling-to-sibling. Each edge in the AS graph is labeled with the types of the AS relationships. The import and export policies complying with AS relationships lead to the *valley-free* property of the AS paths in an AS graph; that is, in a valid AS path, a provider-to-customer or peer-to-peer edge should be followed by a provider-to-customer edge only. In an AS graph with AS relationships, a valid path must be *loop-free* and *valley-free.*

In an AS graph, two valid paths can be concatenated into one valid path with operator "+", i.e. $(v_i \cdots v_1)+(u_j \cdots u_1) = (v_i \cdots v_1 u_j \cdots u_1)$ if this resulting path is both *loop-free* and *valley-free*; otherwise, the result is an empty path ψ.

We are able to find the paths from an AS to another AS in an AS graph, as has been done in [5]. However, BGP provides AS paths based on a source AS and a destination prefix but not two ASs. In order to infer the AS paths specific to destination prefixes, the topological information of the prefixes should be encoded into the AS graph. As shown in Figure 1, we also model the destination prefixes as nodes. The *AS-prefix graph* $G = (V, V_p, E, E_p)$ is composed of AS set V, the prefix set

V_p, the AS edge set E, which consists of the edges between ASs that are labeled with AS relationships, and the AS-prefix edge set E_p, which comprises of links connecting ASs to their originated prefixes. Note that a prefix is attached to the ASs that originate it. It is possible that a prefix might connect to multiple ASs [20]. A path in an AS-prefix graph can append a prefix at the tail of the path, but a prefix cannot appear at the other positions of the path. The length of a path counts for the number of ASs only.

Since the BGP information provides the AS path information specific to source ASs and destination prefixes only, for one AS, our algorithm can infer one AS path specific to one destination prefix only. Although there might be multiple paths from one source AS to one destination prefix in the inter-domain routing system, our measure results show that among all the known source AS and destination prefix pairs, only 0.8% of them have multiple paths, i.e. our model fit with the reality of the Internet in 99.2% cases. Meanwhile, our algorithms are capable of providing a set of candidate paths from a source AS to a destination prefix.

B. BGP Policy-driven path selection procedure

In this section, we first go through the path selection procedure of BGP and then see how the policy driven nature of BGP poses challenges to the AS path inference problem.

For a specific prefix p, an AS u maintains a path set $rib\_in(u)[p]$ that contains all the feasible paths from u to p learned from u's neighbors. Among these paths, u will choose the best one as its path to p, as is denoted by $path(u)[p]$.

BGP is specified as a *policy-based* routing protocol [2]. Each AS path is associated with several configurable path attributes, such as local preference, MED (Multi-Exit-Discriminator), etc. These attributes enable BGP to select the best path by comparing relevant path attributes according to a complicated path decision process [21]. Network administrators implement their routing policies by manipulating these path attributes. In other words, given an AS u and a prefix p, $path(u)[p]$ is chosen with a path selection procedure $bestpath_{u,p}()$ from path set $rib\_in(u)[p]$, i.e. $path(u)[p] = bestpath_{u,p}(rib\_in(u)[p])$.

The rib_in of u is updated in the following way: once u receives a path P from one of its neighbors, if $(u) + P$ is a valid path and conforms to u' path import policies, u updates $rib\_in(u)[p]$ by an operator "$\Cup$": $rib\_in(u)[p] = rib\_in(u)[p] \Cup \{(u) + P\}$. "$\Cup$" is different from the ordinary union operator "$\cup$" in the sense that if $rib\_in(u)[p]$ previously installed a path from the same neighbor, the old path will be replaced by the new one. Next, the path selection procedure $bestpath_{u,p}()$ is triggered. Then the output best path $path(u)[p]$ is in turn announced to the relevant neighbors according to u's export policies.

C. AS Path Inference Problems

The objective of the paper is to infer the exact AS path that an AS uses to reach a destination prefix, which is referred to as the *single AS path inference problem*. Within the context of the BGP policy-driven path selection model, a general algorithm for the problem can be described as follows: *to find an function $bestpath'()$ for any AS u and prefix p, such that its output is most likely the output of $bestpath_{u,p}()$.* Note that $bestpath_{u,p}()$ is specific to p and controlled by the local routing policies of u. Because the local routing policies can be arbitrarily configured and usually independent of the configurations of the other ASs, it is extremely difficult to obtain the local information only relying on the BGP routing tables of several vantage points instead of the direct access to either the sources or the destinations. This policy-driven nature of BGP determines that we cannot infer AS path with 100% percentage accuracy except that we are able to accurately infer the local policies of any source AS for any destination prefix.

In practice, the routing policies of most of the ASs conform to a *typical routing preference* strategy, i.e. an AS prefers the paths through its customer links to those through its peer and provider links [22], [23]. It seems that $bestpath'()$ can be implemented according to the inferred AS relationships to conform to the typical routing preference strategy. However, the model of AS relationship is only an ideal model to describe the common commercial relationships between ASs. In practice, the ASs conduct routing configurations following the commercial contracts, but not the strictly defined ideal model [24]. Besides, the available AS relationship inference algorithms [4], [14], [15] can not guarantee 100% accurate inference [5]. In fact, it is reported that the adoption of typical routing preference strategy in an AS graph would cause routing divergence [23]. In [5], the authors adopted a *shortest path first* (SPF) path comparison procedure to compute the paths. In this paper, we also adopt the SPF-based path selection procedures. In contrast to [5], besides SPF, we also exploit some heuristics to explicitly figure out the possible best path among the potential ones.

Due to the policy-driven nature of BGP, no matter how we design the best path selection procedure $bestpath'()$, it is still challenging to rely on a uniformly defined procedure and some metrics derived from the public BGP routing information to infer AS paths that match with the actual path selection procedure $bestpath_{u,p}()$ for any particular u and p. Therefore, we also try to propose a relatively easier alternative of the single AS path inference problem, i.e. the *potential AS path inference problem*: to infer a set of AS paths from a source AS to a destination prefixes and some of these paths are *potentially* the actual paths that the source AS uses to reach the destination prefix.

IV. Inference Algorithms

A. Methodology

Our algorithms rely on the snapshots of BGP routing tables from a number of vantage points. The inter-domain topology information can be obtained from three major data sources. The first one is to use traceroute-based tools to explore the inter-domain path. However, this approach, relying on large amounts of active measurements, not only is costly in terms of traffic overhead and time-consuming (usually weeks [11]) but also cannot guarantee to capture complete and up-to-date path

information in a short time period. The second one is the Internet Routing Registries [25], which maintain the routing policies of each AS. However, the information in the databases, provided and updated by the administrators voluntarily, might be out-of-date, inaccurate and incomplete [26]. The last and the most frequently used one is the collection of BGP data provided by several public data repositories. Although even the collection of BGP tables cannot provide a perfectly "complete" Internet AS topology due to the partial views of a limited number of vantage points [27], the combined data can provide a much more complete and up-to-date AS topology than the other two sources regarding the time and the resource constraints. Moreover, the BGP tables are easy to access and can be processed with low costs. More importantly, the routes to majority of the prefixes in the Internet have been found to be stable for days [28]. Among the available data sources, only the BGP information can be obtained and processed within hours while ensuring the relative completeness and accuracy. Therefore, similar to [5], we also exploit the BGP routing tables from several vantage points to infer the AS paths. Because we utilize BGP routing tables as the basis of our inference algorithm, we are able to provide near real-time inference results based on the most recent BGP table dumps. We have setup a website [29] to provide an online AS path inference service. Users are able to access our inference results through the provided web interfaces or APIs. Our goal is to provide a general-purpose AS path inference service to the research communities.

We proposed two algorithms to solve the single AS path inference problem and the potential AS path inference problem. Similar to [5], the algorithms infer the path based on the AS-prefix graph extracted from BGP routing tables and use SPF-like criteria to compute the paths. Two major points described below distinguish our algorithms from that in [5].

B. Incorporating Known Paths

The AS-prefix graph is extracted from the BGP tables of N given vantage point ASs. For each destination prefix p, we know at most N known paths to this destination before we perform the inference algorithm. A disadvantage of [5]'s approach is that it does not utilize these known paths. Thus, it does not guarantee the inference accuracy even for the vantage point ASs themselves. The intuition of our algorithm is to incorporating these known paths into inference process. The details are described as follows.

Suppose that path $P_k = (v_k v_{k-1} \cdots v_1 p) = loc\_rib(v_k)[p]$ is the known path from a vantage point v_k to a destination prefix p. According to the BGP model, the best path of v_k is derived from the best path of v_{k-1} with path concatenation. Thus one of the best path of v_{k-1} to p at this time MUST be the sub-path $P_{k-1} = (v_{k-1} v_{k-2} \cdots v_1 p)$. Similarly, one of the best path of v_{k-2} to p must be the sub-path $P_{k-2} = (v_{k-2} \cdots v_1 p)$, and so forth. Thus we can derive $(k-1)$ best paths from a known path of length k. We call both the known path P_k and the derived best path $P_{k-1}, \cdots, P_1$ as *sure paths*. The ASs that have the *sure paths* are the *base ASs*.

When we combine the sure paths derived from the routing tables of the N vantage points, a base AS might have multiple paths to the same destination prefix. In this case, a path attribute *frequency index*, denoted by $P.freq$ for a sure path P, is introduced to identify the frequency that path P is observed by the vantage points. For a prefix p, if one of its sure path is the sub-path of the AS paths of n vantage points, we say the *frequency index* of this sure path to p is n. For instance, in Figure 2, AS A and AS C are vantage points. The frequency index of the sub-path (DE) is 2 and that of the sub-path (CDE) is 1. Intuitively, assuming that the vantage points are *evenly distributed* over the Internet, the frequency index should be a reasonable metric that indicates how frequently this sub-path is possibly used by the other ASs to reach the destination prefix node.

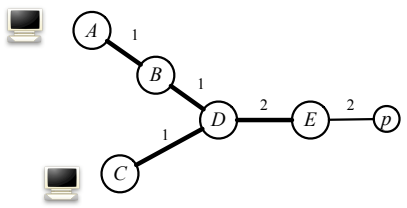


Fig. 2. An illustration of the attribute *frequency index*

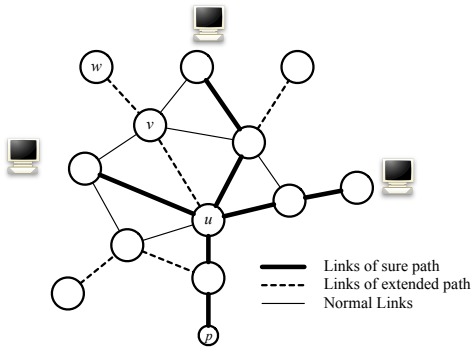


Fig. 3. Extending use paths to infer AS paths

[30] found that it was prevalent in the Internet that the AS paths from different ASs to a destination will finally converge to some shared sub-paths at some intermediate ASs. In the context of this paper, this observation implies that the sure paths of some base ASs to a destination prefix are frequently reused by the other ASs to reach this prefix. Therefore, we are able to infer the AS paths from an AS to a destination prefix by extending the sure paths to this AS instead of inferring the AS paths from this AS to the destination hop by hop.

A sure path can be extended in the following way. As shown in Figure 3, given a AS-prefix graph G and the sure path P of AS u, for a neighbor of u, say v, if the new path $(v) + P$ is valid, i.e. $(v) + P \neq \psi$, sure path P can be extended to an *extended path* $(v) + P$ for v. Furthermore, for a neighbor of v, say w, if $(wv) + P \neq \psi$, the extended path $(v) + P$ can be further extended to $(wv) + P$ for w.

We associate two path attributes to an extended path. One is the *unsure length*, denoted by $P_v.ulen$ for the extended path P_v, which is the length of the extended part of the

```
INITACTIVEQUEUE(p, queue, G, baseASset)
 1  for v ∈ baseASset
 2  do APPEND(queue, ∪v)
 3     path(v)[p] ← sure path of v
 4     SORT(rib_in(v)[p])

KNOWNPATH(p, G = (V, E), baseASset)
 1  queue ← ∅
 2  INITACTIVEQUEUE(p, queue, G, baseASset)
 3  while queue.length > 0
 4  do u ← POP(queue, 0)
 5     for v ∈ peers(u)
 6     do P_u ← rib_in(u)[p][0]
 7        if v ∉ baseASset and (v) + P_u ≠ ψ
 8           then tmppath ← rib_in(v)[p][0]
 9                rib_in(v)[p] ← rib_in(v)[p] ⊎ {(v) + P_u}
10                SORT(rib_in(v)[p])
11                if tmppath ≠ path(v)[p][0] and v ∉ queue
12                   then APPEND(queue, v)
13  return {rib_in(v)|∀v ∈ V}
```

Fig. 4. Pseudo codes of the KNOWNPATH algorithm that incorporates known paths

```
COMPAREPATHSPF(P_1 = (uv_1 ··· p), P_2 = (uv_2 ··· p))
1  if |P_1| ≠ |P_2|
2     then return |P_1| − |P_2|
3  if P_1.ulen ≠ P_2.ulen
4     then return P_1.ulen − P_2.ulen
5  if P_1.freq ≠ P_2.freq
6     then return P_2.freq − P_1.freq
7  return v_1 − v_2

COMPAREPATHLUF(P_1 = (uv_1 ··· p), P_2 = (uv_2 ··· p))
1  if P_1.ulen ≠ P_2.ulen
2     then return P_1.ulen − P_2.ulen
3  if P_1.freq ≠ P_2.freq
4     then return P_2.freq − P_1.freq
5  if |P_1| ≠ |P_2|
6     then return |P_1| − |P_2|
7  return v_1 − v_2
```

Fig. 5. Pseudo codes of the two path comparison procedures

extended path. For instance, for the extended path $(v) + P$, its unsure length is 1 and for $(wv) + P$, its unsure length is 2. The other path attribute is the *frequency index*, denoted by $P_v.freq$ for the extended path P_v, which has the same value as the frequency index of its base sure path. Intuitively, the *unsure length* of an extended path is the measure of uncertainty of the path extending operation since the longer the sure path is extended, the less confident we are of the inference accuracy. The *frequency index* of an extended path also has some intuitive meaning. For a source AS and a destination prefix, given two inferred paths with the same unsure length, we would say that the one that has a higher frequency index might be the correct one since we have seen more ASs using the sure path of this path to reach the destination than the other one.

Our inference algorithm is derived from the Bellman-Ford algorithm. The Bellman-Ford algorithm iteratively computes the shortest paths from all other nodes to a destination node, starting from the initial state in which only the path of the destination node is known. On the contrary, our inference algorithm, which is called KNOWNPATH, starts from the initial state in which all the base ASs install their sure paths. Then the algorithm computes the shortest paths for all the other ASs by extending the sure paths. The pseudo codes of the KNOWNPATH algorithm is shown in Figure 4. The algorithm maintains a global variable $queue$ which stores the ASs whose best paths have been changed. In the initial state, all the base ASs install their sure paths and thus are put into $queue$. Later, in each iteration, it is the turn of an AS in $queue$ to propagate its best path to its neighbors. If the best path of one of its neighbors changes, this neighbor will be put into $queue$. Note that the base ASs will never be added into $queue$ after the initial round.

We apply SPF-based path comparison procedures to select the best path, which provides a *monotonic* ranking of the paths in rib_ins [31]. Thus the path selection procedure $bestpath'()$ can be implemented by simply sorting the paths in rib_in in descending order of preference and then returning the first one. Thus the resulting rib_in is an ordered set and the best path is $rib\_in(u)[p][0]$. According to the formulations of the single AS path inference problem and the potential AS path inference problem, given the ordered path set $rib\_in(u)[p]$ for a source AS u and destination prefix p, the solution to the single AS path inference problem is its best path $rib\_in(u)[p][0]$ and the solution to the potential AS path inference problem is its path set $rib\_in(u)[p]$ or the first K paths of $rib\_in(u)[p]$, denoted by $rib\_in(u)[p][:K]$, if the size of a potential path set is limited by a given parameter K.

[5] employees the shortest path length as the single criterion to select the best paths, which might lead to multiple best paths between two ASs. In contrast to [5], equipped with the two additional path attributes *unsure length* and *frequency index*, we try to apply some SPF-based path comparison heuristic to figure out the single best path. As shown in Figure 5, the first strategy is the procedure COMPAREPATHSPF, which is almost the basic form of the SPF strategy expect that we utilize the unsure length and the frequency index of the extended paths for the further tie breaking. In this procedure, given two paths in the rib_in of an AS, in the first step, the shorter one will be preferred; in the second step, the one with a shorter unsure length is preferred; in the third step, the one with a higher frequency index is preferred; finally, in order to get a deterministic result, the one that has lower first hop AS number is preferred. We call this path comparison strategy as *Shortest Path First* (SPF) strategy. The second strategy is the procedure COMPAREPATHLUF, which utilize the intuitions about the unsure length and the frequency index to try to minimize the unsure length and maximize frequency index of the path. The procedure is similar to COMPAREPATHSPF except the priorities of the criteria. The second and third steps are performed prior to the first step. We call this strategy as *Least Uncertainty First* (LUF) strategy. In the path inference algorithm, once the rib_in of an AS is updated, the AS will sort the paths in its rib_in by calling one of the path comparison procedures. We will compare the performance of the inference algorithms adopting these two path comparison strategies in the experiments.

C. Incorporating Potential Paths

In the Bellman-Ford algorithm, a node propagates a single best path to its neighbors. A disadvantage of the single path

```
MULTIPATHS(p, G = (V, E), baseASset, M)
 1  queue ← ∅
 2  INITACTIVEQUEUE(p, queue, G, baseASset)
 3  while queue.length > 0
 4  do u ← POP(queue, 0)
 5     for v ∈ peers(u)
 6     do if v ∉ baseASset
 7           then tmppathset ← path(v)[p][: M]
 8                for P_u ∈ rib_in(u)[p][: M]
 9                do if (v) + P_u ≠ ψ
10                      then rib_in(v)[p] ← rib_in(v)[p] ∪ {(v) + P_u}
11                for (vx ··· p) ∈ rib_in(v)[p]
12                do if x == u and (x ··· p) ∉ rib_in(u)[p]
13                      then rib_in(v)[p] ← rib_in(v)[p] − {(vx ··· p)}
14                SORT(rib_in(v)[p])
15                if tmppathset ≠ path(v)[p][0 : M] and v ∉ queue
16                   then APPEND(queue, v)
17  return {rib_in(v)|∀v ∈ V}
```

Fig. 6. Pseudo codes of MULTIPATHS algorithm that allow the multiple paths advertisement

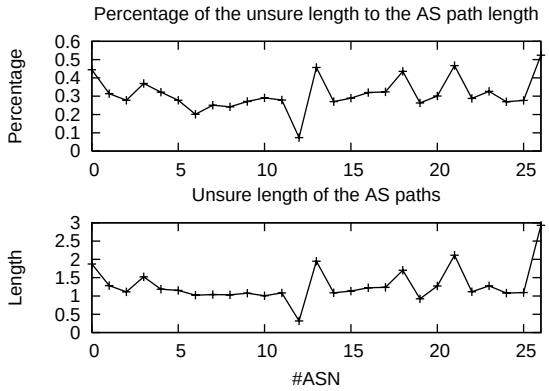


Fig. 7. Weight of the unsure length in the AS paths of the examined ASs

inference algorithm is that the inference inaccuracy of the predecessor ASs' paths will be inherited by the successor ASs. Thus, the inference accuracy of the best path will exponentially decrease with the growth of the unsure length. Our experimental results show that if we look at the best M paths in rib_in instead of the first one, the chance that the actual path shows in these M paths can increase significantly. If the best M paths are allowed to be propagated to the neighbors, the successor ASs will have higher chance to acquire the correct paths. Consequently, the chance that we are able to find the correct path in the successors' $rib_in(u)[p]$s will be increased and the inference accuracy for the potential AS path inference problem will be improved. Therefore, on the basis of the previous codes, we further modify the KNOWNPATH algorithm to support the advertisement of multiple paths, which is called MULTIPATHS, as shown in Figure 6.

In the MULTIPATHS algorithm, maximal M paths are allowed to be propagated from an AS to its neighbors, where M is a configurable parameter. In each iteration, an AS u propagates its best M paths to its neighbors. At first, the M paths are added to each neighbor's rib_in. We use the ordinary union operator "$\cup$" instead of "$\Cup$" to add the new path. Then each neighbor will withdraw the paths that are previously learned from u but now no longer show in u's first M paths. Once the rib_in of a neighbor is updated and a path change is detected, this AS will be put into $queue$.

V. EXPERIMENTS AND VALIDATION

In this section, we use BGP data to validate the performance of the algorithms. Before doing this, we first introduce the selection of the data sources of BGP routing tables.

A. Data Sources

In this paper, we use the routing tables from ROUTEVIEWS [32], RIPE RIS project [33] and CERNET BGP VIEW [34]. All of them have been collecting BGP routing data around the world for several years. In the experiments, we mainly focused on the BGP data collected at 16:00 PM GMT on 10/10/2004. In order to evaluation the performance of the algorithm over time, we also examine the performance of the algorithms over the data sets collected at 00:00 AM GMT on 7/10/2004, 12/22/2004 and 3/20/2005 respectively.

We mainly use the routing tables from ROUTEVIEWS as the basis of inference. Table I listed these ASs and their relevant information in ascending order of AS number. The third column of the table shows their names and countries and the fourth column shows the tiers that these ASs located in the Internet hierarchy [14] and their degrees as well as degree ranks. The information shows that these ASs provide a view set that has adequate diversities in terms of both geographical locations and Internet hierarchical positions. Based on the BGP tables of these ASs, we construct the AS-prefix graph, infer the AS relationships, and then perform our algorithms. The AS relationship inference algorithm in [4] is adopted in this paper.

We analyze the statistics of the sure paths in the routing tables of those ASs in Table I in 10/10/2004 data set. It is found that the number of distinct paths in these routing tables is $5,687,631$ while the number of the derived sure paths is $7,379,993$. Surprisingly, the number of sure paths expands by 1.3 times only. In addition, except those origin ASs of the prefixes, the number of base ASs is 1081 only out of the totally 18382 ASs. The numbers suggest that the Internet have a core composed of a few ASs and the paths of these intermediate ASs be frequently used by the edge ASs to reach the other part of the Internet. This observation is confirmed with the findings of [30]. We also examine the number of sure paths between the any pair of source AS and destination prefixes. We found that 99.2% of them have single path only and 8% have two or more paths, which shows our assumption that there is one path for a pair of AS and destination holds in most cases.

We mainly use the routing tables from the other two sources rather than ROUTEVIEWS for the purpose of validation. These ASs are listed in Table II in ascending order of AS number. The information pertaining to the geographical locations and the Internet hierarchical positions is also listed in the table. In addition, based on the known paths of the ASs in Table I, Figure 7 shows the average unsure length and the average percentage of unsure part of the paths for all the prefixes for each of the examined ASs in the table. Intuitively, the average unsure length measures the average distance of

TABLE I

ROUTING TABLES FOR AS PATH INFERENCE

	ASN	IP Address	Name/Country	T/D/R
0	286	134.222.85.45	KPN/NL	2/264/26
1	293	134.55.200.1	ESN/US	3/111/62
2	1221	203.62.252.26	Telstra/AU	3/86/78
3	1239	144.228.241.81	Sprint/US	1/1737/3
4	1299	213.248.83.240	TeliaNet/SE	1/288/22
5	1668	66.185.128.48	AOL/US	2/102/66
6	2493	206.186.255.223	Sprint/CA	5/1/12702
7	2497	202.232.0.2	JPNIC/JP	2/200/38
8	2828	65.106.7.139	XO Comm/US	1/450/16
9	2905	196.7.106.245	UUNET/ZA	3/27/267
10	2914	129.250.0.11	Verio/US	1/637/7
11	3257	213.200.87.254	Tiscali/DE	2/256/30
12	3277	194.85.4.55	RUSNet/RU	5/80/86
13	3292	195.249.0.135	TDC/DK	2/276/25
14	3303	164.128.32.11	Swisscom/CH	1/526/14
15	3333	193.0.0.56	RIPE/NL	3/127/55
16	3356	4.68.0.243	Level3/US	1/1189/4
17	3549	208.51.134.254	GlobalCrossing/US	1/598/9
18	3561	206.24.210.26	Savvis/US	1/614/8
19	4513	195.66.224.82	Globix/US	2/532/13
20	5056	167.142.3.6	IowaNetS/US	3/13/594
21	5459	195.66.232.254	London IX/GB	3/159/43
22	5511	193.251.245.6	France Telecom/FR	1/185/39
23	5650	208.186.154.35	Elec. Lightwave/US	1/185/40
24	6079	207.172.6.227	RCN/US	3/127/56
25	6395	216.140.8.59	Broadwing/US	1/311/18
26	6453	195.219.96.239	Teleglobe/CA	1/261/28
27	6461	209.249.254.19	Metromedia/US	3/588/10
28	6509	205.189.32.44	Canarie/CA	3/36/190
29	6539	216.18.63.137	Group/CA	3/263/27
30	6939	216.218.252.145	Hurricane/US	2/310/19
31	7018	12.0.1.63	AT&T/US	1/1929/2
32	7660	203.181.248.233	APAN/JP	3/28/261
33	7911	64.200.151.12	Williams/US	1/284/24
34	8001	209.123.12.51	Net Access Corp/US	2/225/36
35	8075	207.46.32.32	Microsoft/US	3/210/37
36	9942	203.194.0.12	COMindico/AU	2/146/48
37	11537	198.32.8.196	Abilene/US	3/77/93
38	11608	207.246.129.14	Accretive/US	3/98/69
39	12956	213.140.32.146	Telefonica/ES	2/227/35
40	15290	216.191.65.126	Allstream/CA	2/163/42
41	16150	217.75.96.60	Port80 AB/SE	3/36/196

these ASs to the sure paths. It shows that on average the examined ASs are $0.3 \sim 2.9$ AS hops away from the base ASs and have a percentage of unsure length of $7 \sim 52\%$. Given the high diversities exhibited in terms of both the locations and the unsure length statistics of these ASs, we have some confidence of having a representative sample of ASs in the Internet. We believe that these ASs will not lead to a biased evaluation of the performance of our algorithms.

B. Upper bounds on inference accuracy

Before the validation, we first use the data collected on 10/10/2004 to investigate how the inaccuracy of the topology information of the resulting AS-prefix graph impact the performance of the AS path inference algorithm.

We first use the BGP tables of those ASs in Table I to extract the AS-prefix graph and infer the AS relationships. Then we check the AS paths in the tables of those in Table II to see how many paths can be found in the resulting AS graph. There are two possible reasons that a path cannot be found in the graph. The first is that this path contains the links that are invisible to the vantage points in Table I. We categorize this kind of paths as the *missing-link paths*; the other reason is that this path is not valley-free in the resulting AS-prefix graph. We categorize this kind of paths as the *invalid-policy paths*.

TABLE II

ROUTING TABLES FOR VALIDATION

#	ASN	IP Address	Name/Country	T/D/R
0	513	192.65.184.3	CERN/CH	3/16/471
1	1103	195.69.144.34	SURFnet/NL	3/35/199
2	1853	193.203.0.1	ACOnet/AT	3/25/292
3	2116	194.68.123.149	Catch Comm./NO	3/17/449
4	3741	168.209.255.2	Internet Solution/ZA	4/44/151
5	4538	202.112.60.251	CERNET/CN	3/14/549
6	4608	202.12.29.64	APNIC/AU	5/1/12885
7	4777	202.12.28.190	APNIC NSPIXP2/AU	3/3/2699
8	5392	217.29.66.5	TELNET/IT	4/3/2709
9	5417	195.69.144.99	Demon/GB	3/8/977
10	6762	195.69.144.196	Telecom/IT	2/79/89
11	8251	195.69.144.52	Cistron IP/NL	3/10/796
12	8434	194.68.123.66	Telenor AB/SE	3/67/104
13	9177	212.47.190.1	SOLPA AG/CH	4/8/1009
14	12779	217.29.66.65	ITGATE.Net/IT	4/8/1033
15	12793	193.203.0.52	eTel/AT	3/18/440
16	12859	195.69.144.200	BIT/NL	3/25/304
17	13030	198.32.160.103	Init Seven/CH	3/28/264
18	13129	212.20.151.234	Cogent/DE	2/57/118
19	13237	195.69.144.212	LambdaNet/FR	2/135/53
20	15837	80.81.192.126	Tele GmbH/DE	5/16/493
21	20854	194.153.154.35	Mega Provider/NL	3/3/3825
22	20932	192.65.185.142	SIG/NL	3/9/937
23	25232	195.69.144.121	Rokscom/NL	5/19/417
24	25560	80.81.192.106	rh-tec/DE	4/8/1069
25	29686	195.69.145.56	Probe/DE	2/17/469
26	31477	193.111.172.55	DUOCAST/NL	5/2/11863

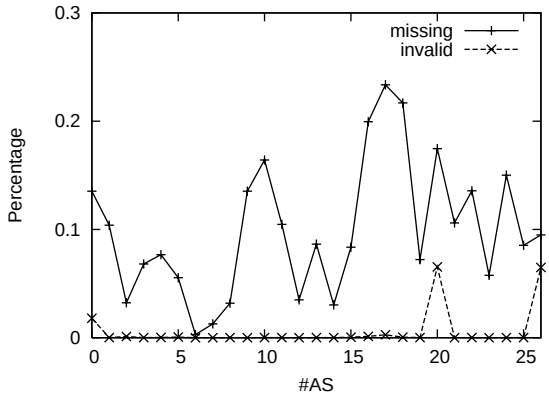


Fig. 8. Impact of the inaccuracy of topological information on the AS path inference performance

These two kinds of paths are referred to as *uninferable paths* (In contrast, the other paths are *inferable* for this AS-prefix graph.). Note that these two categories might be overlapped since a path cannot be found due to both missing links and policy violations. The percentage of the uninferable paths for each examined tables is shown in Figure 8. It shows that on average about 10.5% of paths cannot be found in the given AS-prefix graph. Although [5] found that the results of AS relationships inference could impact the performance of AS path inference, our finding shows that there are only a few invalid-policy paths in most of the tables except those of AS15837 and AS31477. But even if the AS relationships are inferred with 100% accuracy, there still remain average 10% of paths uninferable due to the missing links. The existence of these uninferable paths provides an *upper bound* to the AS path inference accuracy given that the inference is performed over the given AS-prefix graph.

C. Validation

In this section, we are going to validate the performance of our inference algorithms. The AS paths of the ASs in Table II are inferred on the basis of the AS-prefix graph extracted from the routing tables of the vantage points in Table I.

Because each run of the algorithm is able to infer the AS paths of all the ASs to a destination prefix, we need to run around $150,000$ to check the general performance of the algorithms for all the prefixes. However, many prefixes shared identical AS paths [35]. By clustering the prefixes that share the identical AS paths, we get only about $27,000$ prefix clusters. For each prefix cluster, we pick only one prefix as the destination prefix and feed it to the inference algorithms. The resulting AS paths are used for all the prefixes belonging to this prefix cluster. Thus the actual running time for the validation is significantly reduced by 7 times. We took the validation experiments on a desktop equipped with Pentium IV 2.8 GHz CPU and 1 GB RAM. In order to get the results for all the prefixes, we need to run the algorithms around 24$\sim$48 hours, i.e. it takes around 4$\sim$8 seconds to infer the AS paths of all the ASs to one particular destination prefix.

In order to evaluate the performance of the algorithm, we use the similar metrics defined in [5] for evaluation. For an examined AS u and a particular prefix p, by comparing the path lengths of the inferred path and the actual path, the result can be *longer*, *shorter* or *length match*; by checking whether the actual path is the best path or just shows up in the rib_in, the result can be *exact match* (the inferred best path is the actual path) or *one match* (one of the paths in the rib_in matches with the actual path). Besides these metrics, we also define a metric *(one) match in* K which indicates whether the actual path is in the first K paths in the rib_in. The metrics *longer*, *shorter*, *length match* and *exact match* are used for the evaluation for the single AS path inference problem and the metrics *one match* and *match in* K are for the potential AS path inference problem. In order to show the inference upper bound imposed by the inaccuracy of the given AS-prefix graph, we use metric *upper bound* to refer to the percentage of the inferable paths, which can be directly derived from the ratios of uninferable paths shown in Figure 8.

We mainly checked the performance of the algorithms on the data set collected on 10/10/2004. We ran KNOWNPATH that uses COMPAREPATHSPF and COMPAREPATHLUF respectively and the multiple path version MULTIPATH algorithm with parameter $M = 3$. For the sake of comparison, we also ran the basic Bellman-Ford algorithm that incorporates neither known paths nor multiple potential paths, which is the similar approach adopted in [5].

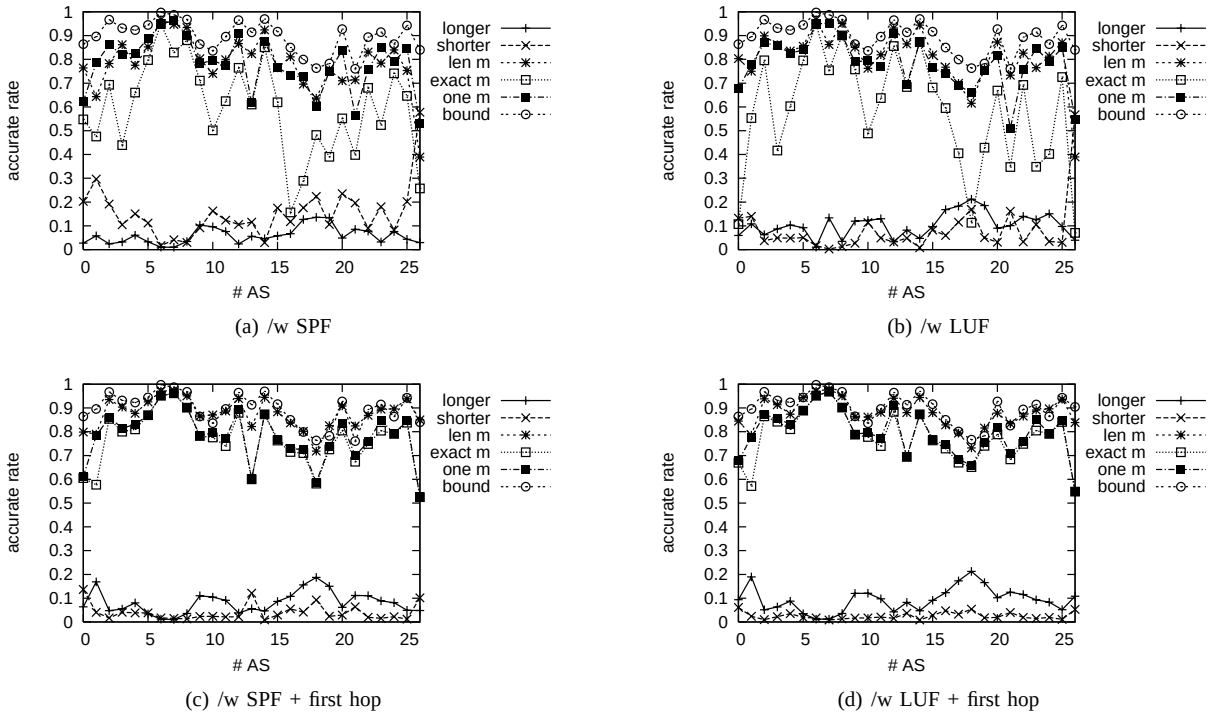
In [5], the authors developed a novel technique to detect the first AS hop that a source AS will use to reach a destination. They showed that if the first AS hop information had been provided, their inference accuracy of length match could be improved by up to 15% to around 70%$\sim$88%. In this paper, we also check how the performance of the algorithms will be improved by incorporating the first AS hop information assuming that the information had been available beforehand.

Graphs in Figure 9 show the performance of KNOWNPATH in difference scenarios. The x axis in the graphs represents the indexes of the examined vantage points in Table II and y axis indicates the value of each performance metric.

Figure 9(a) shows the performance of KNOWNPATH that leverages the SPF path comparison procedure and Figure 9(b) shows that of the LUF path comparison procedure. In the case of the SPF strategy, the percentage of the shorter paths is apparently more than longer paths while the percentages of the shorter and longer paths are nearly comparable in the later case. We find that the average performances of the other metrics are approximately the same in two figures. But for the individual ASs, in terms of exact match, the two strategies yield distinct performances for different vantage points. For instance, the SPF strategy yields less than 40% exact match paths for AS12859, AS13030, AS13237, AS20854 and AS31477; the worst case (AS12859) is only 16% while the best case (AS4608) is up to 95%. In contrast, the LUF strategy yields less than 40% exact match paths for AS513, AS13129, AS20854, AS25232 and AS31477; AS31477 is the worst case (7%) and AS4608 is the best case (95%). Such various performances of the two path comparison procedures for different vantage points suggests that the routing policies adopted by each individual AS are highly diverse and we cannot apply an uniformly defined path selection procedure to all the ASs. This observation also suggests a possible future direction for the AS path inference problem, i.e. to apply different routing selection procedure for particular ASs.

It is worthy of mentioning that in the experiments, we also tried several other path comparison procedures. For example, we tried the typical routing preference strategy, i.e. the customer routes are superior to the peer and provider routes no matter how long the paths are. We also tried other path comparison procedures by exchanging the priorities of each criterion in the procedures COMPAREPATHSPF or COMPAREPATHLUF. However, we found that on average the SPF and the LUF strategies significantly outperform the other path comparison strategies even the strategy that employees the typical routing preference. The observation might suggests that most of the ASs in the Internet not utilize the local preference to set the routing policies but still rely on the shortest path first strategy as default for most of the prefixes. Another interesting observation is that although the authors in [23] observed route oscillations in the AS graph with inferred AS relationships if they adopted the typical routing preference path comparison strategy, we did not observe route oscillations when we use the same path comparison procedure in our KNOWNPATH algorithm. Note that [23] found that the oscillations are caused by the provider-to-customer cycles in the AS graph. Our observation suggests that the introduction of the known paths into the graph be able to separate the provider-to-customer cycle in the AS graph and prevent route oscillations.

We find that by incorporating the first AS hop information into AS path inference, the performance of the KNOWNPATH algorithm is improved significantly no matter which path comparison procedure is used, as shown in Figure 9(c) and 9(d). Moreover, the algorithms produce comparable inference accuracy for different vantage points. The observation again suggests the importance of the local routing policies to the AS



(a) /w SPF

(b) /w LUF

(c) /w SPF + first hop

(d) /w LUF + first hop

Fig. 9. Performance of the KNOWNPATH algorithm

TABLE III
AVERAGE PERFORMANCE OF THE ALGORITHMS

(%)	longer	shorter	length match	exact match	one match
NOPATH	5	25	69	33	49
NOPATH + first hop	7	16	78	51	52
KNOWNPATH w/ SPF	6	15	78	60	78
KNOWNPATH w/ LUF	10	8	81	58	79
KNOWNPATH w/ SPF + first hop	8	4	88	77	78
KNOWNPATH w/ LUP + first hop	9	3	88	78	80
MULTIPATHS w/ LUF	8	10	81	59	86
MULTIPATHS w/ LUP + first hop	10	3	87	77	86
NOPATH for ASs in Table I	0	24	76	40	73

path inference.

For comparison, the averages of the performance metrics in different scenarios are listed in Table III. We use term NOPATH to refer to the basic algorithm that neither incorporates known AS paths nor propagates multiple potential paths. We are able to observe the influence of different factors on the performance. By leveraging known paths, there is a 25$\sim$27% improvement in terms of exact match from 33% to $57 \sim 59$% and a 10$\sim$12% improvement in terms of length match from 69% to $78 \sim 81$%. Furthermore, if the first AS hop information is given, there is another 20% gain in terms of exact match to 78% and another 7% improvements in terms of length match to 88%. In addition, since the MULTIPATHS algorithm adopts the same path comparison procedure as KNOWNPATH, there is no apparent differences in the performance metrics pertaining to the single AS path inference problem. On the other hand, in terms of the performance metrics related to the potential AS path inference problem, the MULTIPATHS algorithm yields the average accuracy as high as 86% for one match, which is about 95% of all *inferable* paths for the examined vantage points.

Furthermore, for the basic algorithm NOPATH, we also check its inference accuracy for the ASs in Table I. The routing tables of these ASs are used to construct the AS-prefix graph for the inference. The results are shown in the last row of Table III. It shows that the algorithm only yields an average accuracy of 40% in terms of exact match and 73% in terms of one match even for the ASs whose AS paths have been known. In contrast, our algorithms are able to guarantee 100% inference accuracy for these ASs.

Figure 10 shows the distribution of average value of the performance metric *match in* K against the value of K for different algorithms and scenarios. For the curve of the KNOWNPATH algorithm, we observe a sharp increase in the first 5 paths. Later, the improvement of the performance metric of match with K becomes almost negligible. Therefore, if we lower the objective of the single AS path inference problem to solve the easier potential AS path inference problem, the accuracy can be increased. This also suggests the intuition behind the MULTIPATHS algorithm. Figure 10 shows that there is an 8% improvement to 86% in terms of one match if we allow multiple paths advertised to the neighbors. We can also

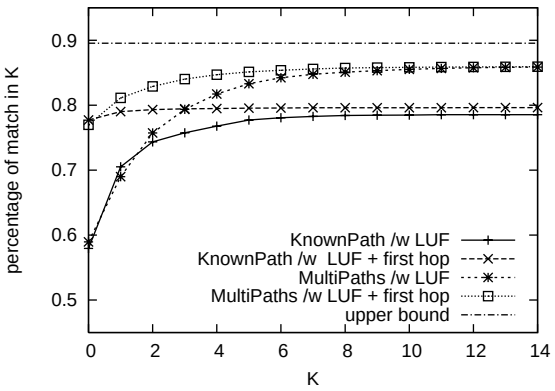


Fig. 10. Distribution of metric *match in* K v.s. K

observe the similar abrupt increase of the inference accuracy in the first few paths. It shows that we can find nearly 86% paths within the first 10 paths. In addition, we found that the incorporation of first hop AS information does not improve the performance of one match but it do help the algorithm identify the best paths.

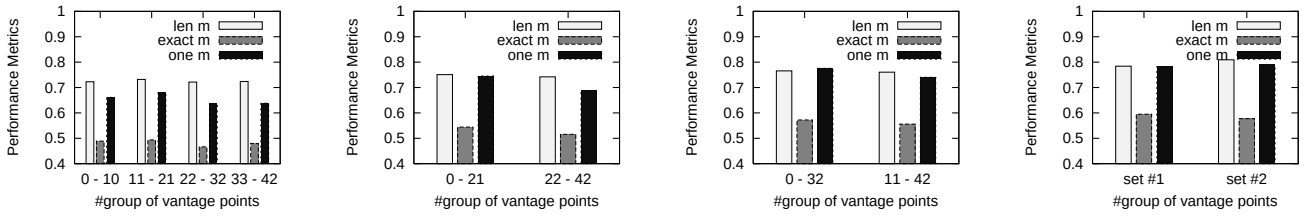
When we employ the MULTIPATHS algorithm, it is found that the first AS hop information improve the accuracy of exact match. But the improvement is approximately the same degree as that to the KNOWNPATH algorithm. There still remain about 10% paths that have been correctly inferred and stored in rib_in but cannot be identified as the best path. The observation suggests that we might need some other additional information to identify the rest of the best paths. We tried several approaches. At first, we tried to exploit the information in IRR. We extracted the AS relationships according to the routing registry information of RIPE NCC with the similar methodology introduced in [26]. By applying the calibrated AS relationships, some paths in rib_ins are eliminated due to the violation of the valley-free property. An improvement of about 0.5% is observed. In addition, [36] reported that there exists a Global Education and Research Network (GERN) in the Internet which is composed of several continental and national educational and research backbones, such as Abilene in US, GÉANT in Europe and APAN in Asia & Pacific. The paths between the ASs in GERN most likely stay in GERN instead of going out through commercial backbones. By applying the heuristic that two ASs in GERN prefer paths within GERN, we get another 0.3% improvement of inference accuracy. The improvement led by these heuristics seems negligible, but they suggest that we might be able to improve the inference accuracy by applying some specific routing policies.

D. Factors influencing the inference accuracy

Because of the limitation of the available data sources, we are able to evaluate the performance of our algorithms for a limited number of ASs. It seems unconvincing to conclude that our algorithms could yield the similar inference accuracy for the rest of the ASs. However, we argue that our inference algorithm could guarantee the similar performance for most of the ASs in the Internet. The reasons are as follows. First, we have shown that the selection of the ASs for validation exhibit adequate diversity in terms of not only the geographical and hierarchical positions but also the distribution of unsure lengths. The second, we performed intense experiments to check the impact of the numbers and the selections of the vantage points on the inference accuracy and the stability of the inference accuracy over time using various combinations of the available BGP routing tables. The experiments are described as follows. In these experiments, we only discuss the performance of the KNOWNPATH algorithm with the SPF path comparison strategy.

The impact of the vantage points on the performance of the algorithm: In order to check the impact of the selection of vantage points on the performance of the algorithm, we tried as many the combinations of the known vantage points as possible to perform our algorithms. We group the 42 vantage points in Table I and II into several groups and incorporate the AS paths of each group into the inference process to examine whether the selection of the known vantage points will influence the performance. First, we group the ASs into 4 groups, which include the ASs in Table I with index $0 \sim 9$, $10 \sim 20$, $21 \sim 30$ and $31 \sim 42$ respectively. Each group has the tables of around 10 vantage points. The inference results for the ASs in Table II are shown in Figure 11(a). Then we divided the top 21 and the bottom 21 ASs into two groups as inference basis. The performance of the algorithm are shown in Figure 11(b). Third, we grouped the top 31 and the bottom 31 ASs into two groups. The inference results based on these ASs for the ASs in Table II are shown in Figure 11(c). Finally, we use the last 15 ASs in Table I and the 27 ASs in Table II to compose another 42 ASs to construct an AS graph to infer the AS paths of the first 27 ASs in Table I. This AS set is referred to as *set #2* and the AS set in Table I is referred to as *set #1*. We compare the results of this experiment based on the ASs in the set #2 with the previous results based on the set #1 in Figure 11(d). According to the charts in Figure 11, it is find that if the numbers of the known vantage points are approximately the same, the influent of the selection of vantage points on the average performance of the algorithm seems minor. For instance, the accuracy in terms of exact match is around 47%$\sim$49% if the number of vantage points is around 10; it is around 52%$\sim$54% if the number is 21; it is around 56%$\sim$57% if the number is 31; and the accuracy is around 58%$\sim$60% if the number is 42. But it is clear that the selection of vantage points do influence the inference performance. It is still worthy of investigating how to select the vantage points in the most efficient way such that it achieve the best inference accuracy.

The impact of the number of vantage points on the performance of the algorithm: In addition, based on the previous experiments, we are able to examine the impact of the number of vantage points on the performance of the algorithm. We compare the performance metrics of the KNOWNPATH algorithm that incorporate the AS paths of different number of vantage points. We checked the cases that the known paths of the first 10, 21 and 31 vantage points in Table I are incorporated. The results are shown in Figure 12. We find that by incorporating the AS paths of more vantage



(a) 10 (b) 20 (c) 30 (d) 40

Fig. 11. Impact of the selection of the vantage points on the performance of the algorithm

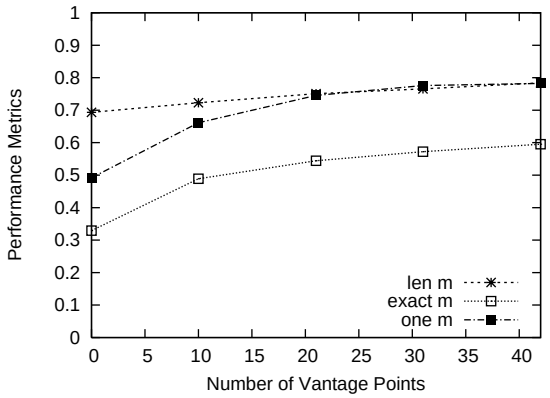


Fig. 12. Impact of the number of the vantage points on the performance of the algorithm

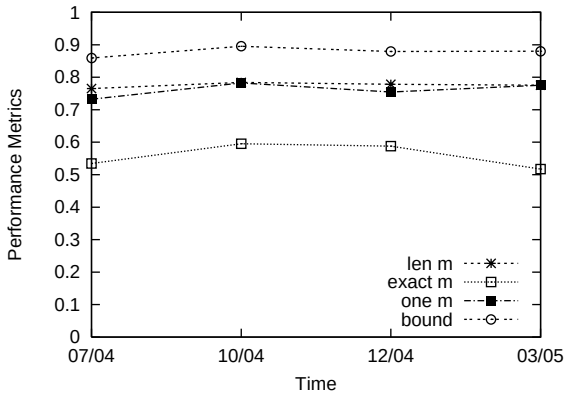


Fig. 13. Stability of the performance of the algorithm over time

points, the performance of the algorithm is improved. But the marginal utility of the vantage points decreases. The observation suggests that although it shows that the more the known vantage points, the more accurate the inference results, we are able to achieve good enough performance by exploiting the BGP routing tables of a moderate number of vantage points.

Stability of the inference accuracy over time: In order to check the stability of the algorithm's performance to different data set over time, we apply the algorithm to the data sets collected on the other three dates: 7/10/2004, 12/22/2004 and 3/20/2005. The change trends in terms of exact match, length match, one match and upper bounds are shown in Figure 13. It is found that the curves fluctuate within an acceptable range over time.

VI. Conclusions and Future Work

In this paper, we proposed several novel techniques to solve the single AS path inference problem and the potential AS path inference problem. At first, by introducing the topological information of the destination prefixes into the AS graph, we are able to infer the AS paths specific to prefixes instead of ASs. Second, by incorporating the known paths of the vantage points into inference process, our algorithms achieve an average accuracy of 60% in terms of exact match and an average accuracy of 81% in terms of length match for the single AS path inference problem. At last, by incorporating multiple potential paths in the inference process, our algorithm achieves an average accuracy of 86% in terms of one match for the potential AS path inference problem.

There still remain some open questions to answer. At first, although the number of vantage points is the major factor that influences the inference accuracy, in the case that the number of vantage points are limited, it is worthy of investigating how the selection of the vantage points impact the inference results. It has been shown that the first AS hop information help improve the inference accuracy, but the measurement approach proposed in [5] need the access to the destination network, which still seems infeasible in practice for most of the destination prefixes. Further investigations of the methodologies that detect the first AS hop information and discover other local routing policies are worthwhile for the AS path inference problem.

References

[1] "traceroute.org, http://www.traceroute.org".

[2] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[3] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos, "On Power-law Relationships of the Internet Topology", in *SIGCOMM '99*, 1999, pp. 251–262.

[4] Lixin Gao, "On Inferring Autonomous System Relationships in the Internet", *IEEE/ACM Transactions On Networking*, vol. 9, no. 6, December 2001.

[5] Zhuoqing Morley Mao, Lili Qiu, Jia Wang, and Yin Zhang, "On AS-Level Path Inference", in *Proceedings of ACM SIGMETRICS*, Banff, Canada, June 2005.

[6] Akihiro Nakao, Larry Peterson, and Andy Bavier, "A Routing Underlay for Overlay Networks", in *Proceedings of ACM SIGCOMM*, Karlsruhe, Germany, August 2003.

[7] Shu Tao, Kuai Xu, Ying Xu, Teng Fei, Lixin Gao, Roch Guerin, Jim Kurose, Don Towsley, and Zhi-Li Zhang, "Exploring the Performance Benefits of End-to-End Path Switching", in *Proceedings of IEEE ICNP 2004*, Berlin, Germany, Octocer 2004.
[8] Ruomei Gao, Constantinos Dovrolis, and Ellen W. Zegura, "Interdomain Ingress Traffic Engineering through Optimized AS-Path Prepending", in *Proceedings of IFIP Networking conference*, Waterloo, Canada, May 2005, pp. 647–658.
[9] Ramesh Govindan and Hongsuda Tangmunarunkit, "Heuristics for Internet Map Discovery", in *Proceedings of the IEEE INFOCOM*, March 2000.
[10] Neil Spring, Ratul Mahajan, and David Wetherall, "Measuring ISP Topologies with Rocketfuel", in *Proceedings of SIGCOMM*, 2002.
[11] "Visualizing Internet Topology at a Macroscopic Scale, http://www.caida.org/analysis/topology/as_core_network/".
[12] Z. Morley Mao, Jennifer Rexford, Jia Wang, and Randy Katz, "Towards an Accurate AS-Level Traceroute Tool", in *Proceedings of ACM SIGCOMM 2003*, 2003.
[13] Z. Morley Mao, David Johnson, Jenifer Rexford, Jia Wang, and Randy Katz, "Scalable and Accurate Identification of AS-Level Forwarding Paths", in *Proceedings of IEEE INFOCOM 2004*, Hong Kong, China, March 2004.
[14] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H. Katz, "Characterizing the Internet Hierarchy from Multiple Vantage Points", in *Proceedings of IEEE INFOCOM*, June 2002.
[15] Giuseppe Di Battista, Maurizio Patrignani, and Maurizio Pizzonia, "Computing the Types of the Relationships between Autonomous Systems", in *Proceedings of IEEE INFOCOM*, 2003.
[16] H. Tangmunarunkit, R. Govindan, and S. Shenker, "Internet Path Inflation Due to Policy Routing", in *Proceedings of SPIE ITCom*, 2001, pp. 188 – 195.
[17] Z. Ge, D.R. Figueiredo, S. Jaiswal, and L. Gao, "On the Hierarchical Structure of the Logical Internet Graph", in *Proceedings of SPIE ITCOM*, August 2001.
[18] Xenofontas Dimitropoulos, Dmitri Krioukov, and Goerge Riley, "Revisiting Internet AS-level Topology Discovery", in *Proceedings of PAM'05*, Boston, MA, March 2005.
[19] David G. Andersen, Nick Feamster, Steve Bauer, and Hari Balakrishnan, "Topology Inference from BGP Routing Dynamics", in *Proceeding of ACM SIGCOMM Internet Measurement Workshop 2002*, Marseille, France, November 2002.
[20] Xiaoliang Zhao, Dan Pei, Lan Wang, Dan Massey, Allison Mankin, S. Felix Wu, and Lixia Zhang, "An Analysis of BGP Multiple Origin AS (MOAS) Conflicts", in *Proceedings of IMW'01*, 2001.
[21] "BGP Best Path Selection Algorithm, http://www.cisco.com/warp/public/459/25.shtml".
[22] Feng Wang and Lixin Gao, "On Inferring and Characterizing Internet Routing Policies", in *ACM SIGCOMM Internet Measurement Conference 2003*, 2003.
[23] Hao Wang, Haiyong Xie, Yang Richard Yang, Li Li, Yanbin Lin, and Avi Silberschatz, "On Stable Route Selection for Interdomain Traffic Engineering", Tech. Rep., Department of Computer Science, Yale University, February 2005.
[24] Nick Feamster, Hari Balakrishnan, and Jennifer Rexford, "Some Foundational Problems in Interdomain Routing", in *3rd ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, San Diego, CA, November 2004.
[25] "Internet Routing Registry, http://www.irr.net/".
[26] Georgos Siganos and Michalis Faloutsos, "Analyzing BGP Policies: Methodology and Tool", in *Proceedings of IEEE INFOCOM*, 2004.
[27] Hyunseok Chang, Ramesh Govindan, Sugih Jamin, Scott J. Shenker, and Walter Willinger, "Towards Capturing Representative AS-Level Internet Topologies", *Computer Networks*, vol. 44, no. 6, pp. 737–755, 2004.
[28] Jennifer Rexford, Jia Wang, Zhen Xiao, and Yin Zhang, "BGP Routing Stability of Popular Destinations", in *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, Marseille, France, November 2002.
[29] "AS Path Inference Service, http://rio.ecs.umass.edu/cgi-bin/asinfer.cgi".
[30] Sanghwan Lee, Zhi-Li Zhang, and Srihari Nelakuditi, "Exploiting AS Hierarchy for Scalable Route Selection in Multi-Homed Stub Networks", in *Proceedings of IMC'04*, Taormina, Sicily, Italy, October 2004.
[31] Timothy G. Griffin and Gordon Wilfong, "On the Correctness of IBGP Configuration", in *Proceedings of ACM SIGCOMM*, August 2002.
[32] "Route Views Project Page, http://www.routeviews.org/".
[33] "RIPE Route Information Service, http://www.ripe.net/ris/index.html".
[34] "CERNET BGP VIEW project, http://bgpview.6test.edu.cn/bgp-view/index.shtml".
[35] Andre Broido and kc claffy, "Analysis of RouteViews BGP data: policy atoms", in *Network Resource Data Management Workshop*, 2001.
[36] Suman Banerjee, Timothy G. Griffin, and Marcelo Pias, "The Interdomain Connectivity of PlanetLab Nodes", in *PAM2004*, Antibes Juan-les-Pins, France, April 2004.