Scalable and Accurate Identification of AS-Level Forwarding Paths

Z. Morley Mao, *University of Michigan*, *zmao@eecs.umich.edu*
David Johnson, Jennifer Rexford, Jia Wang, *AT&T Labs–Research*, {*dsj,jrex,jiawang*}*@research.att.com*
Randy Katz, *UC Berkeley*, *randy@cs.berkeley.edu*

***Abstract*— Traceroute is used heavily by network operators and researchers to identify the IP forwarding path from a source to a destination. In practice, knowing the Autonomous System (AS) associated with each hop in the path is also quite valuable. In previous work we showed that the IP-to-AS mapping extracted from BGP routing tables is not sufficient for determining the AS-level forwarding paths [1]. By comparing BGP and traceroute AS paths from multiple vantage points, [1] proposed heuristics that identify the root causes of the mismatches and fix the inaccurate IP-to-AS mappings. These heuristics, though effective, are labor-intensive and mostly *ad hoc*. This paper proposes a systematic way to construct accurate IP-to-AS mappings using dynamic programming and iterative improvement. Our algorithm reduces the initial mismatch ratio of 15% between BGP and traceroute AS paths to 5% while changing only 2.9% of the assignments in the initial IP-to-AS mappings. This is in contrast to the results of [1], where 10% of the assignments were modified and the mismatch ratio was only reduced to 9%. We show that our algorithm is robust and can yield near-optimal results even when the initial mapping is corrupted or when the number of probing sources or destinations is reduced. Our work is a key step towards building a scalable and accurate AS-level traceroute tool.**

I. Introduction

Traceroute is widely used to detect routing problems, characterize end-to-end paths, and discover the Internet topology. Traceroute sends a series of TTL-limited probes toward a target destination, and reports the interfaces on the forwarding path and the round-trip time for each hop. In Figure 1, the first column shows the output of the traceroute to CNN's web site. This is invaluable to network operators and researchers. For example, network operators use traceroute to identify forwarding loops, blackholes, routing changes, unexpected paths through the Internet, and the end-to-end latency. Upon detecting a routing or performance anomaly, operators need to identify the Autonomous System (AS) responsible for the problem. The second and third columns of the Figure 1 denote the AS information of the routers along the forwarding path. Inaccurate information about the ASes along the path leads to delays in identifying and correcting the problem. In addition, research studies based on AS paths or graphs derived from traceroute depend on having an effective way to map the traceroute data to an AS-level forwarding path.

However, determining the AS-level forwarding path is an inherently difficult problem, due to the operational realities of today's Internet. Conventional approaches have many limitations. First, the AS path advertised via BGP (Border Gateway Protocol) could be used as an estimate of the AS-level forwarding path. However, the AS path traversed by BGP update messages may differ from the forwarding path due to route aggregation and routing anomalies such as deflections. Network operators want to know when these kinds of differences occur in practice. Second, each IP-level hop in the traceroute path could be mapped to an AS number by using an Internet routing registry (*e.g.*, "NANOG traceroute" [2] and prtraceroute [3]). However, the registries are often out-of-date or incomplete. A third alternative is to use the *origin AS*—the AS that initially announced the prefix—extracted from BGP routing tables. Though this information is more accurate and complete, the approach also has limitations such as multiple origin AS's (*MOAS's* [4]), route aggregation, and unannounced address blocks. For instance, for the fifth hop of the traceroute example in Figure 1, both the *whois* address registry and the BGP table return AS25 as the owner AS. However, we will show later that this hop is an exchange point actually belonging to AS11423.

Based on extensive measurements, previous work [1] discovered that a large fraction (around 15%) of the traceroute paths did not match the corresponding BGP paths. They found that most discrepancies between the BGP and traceroute AS paths stemmed from inaccuracies in the IP-to-AS mapping applied to the traceroute data. They proposed heuristics to identify the root causes of the mismatches and fix inaccurate IP-to-AS mappings, based on the comparison of a large collection of BGP and traceroute paths from multiple vantage points.

1	169.229.62.1	AS25	Berkeley
2	169.229.59.225	AS25	
3	128.32.255.169	AS25	
4	128.32.0.249	AS25	
5	128.32.0.66	AS11423	Calren
6	209.247.159.109	AS3356	Level3
7	64.159.2.65	AS3356	
8	64.159.1.46	AS3356	
9	209.247.9.170	AS3356	
10	66.185.138.33	AS1668	GNN
11	66.185.147.208	AS1668	
12	66.185.136.17	AS1668	
13	64.236.16.52	AS5662	CNN

Fig. 1. Example traceroute output with AS information to www.cnn.com.

This work was conducted while Morley was doing her internship at AT&T Labs–Research.

The earlier work concentrated on a set of "explanations," each of which could account for the differences between many pairs of BGP and traceroute AS paths. These explanations built on an understanding of common operational practices, such as the presence of *Internet eXchange Points* (IXP's), where multiple AS's connect to exchange BGP routes and data traffic. Note, however, we do not expect the BGP path to *always* match the traceroute path, as they may differ due to reasons such as routing anomalies.

The heuristics proposed in [1] have several limitations. First, they are labor-intensive. They rely on collecting a large set of traceroute probes, relevant BGP update messages, BGP routing tables, and reverse DNS lookups. Because most of the heuristics were based on the fan-in and fan-out counts of the mismatched AS's, the accuracy of the results can be greatly affected by the number and diversity of data collection locations, as shown in the comparison between 3 vs. 8 vantage points.

Second, the heuristics depend on certain operational realities which arguably could change over time. For instance, incomplete paths cannot be resolved using the heuristics proposed in [1]. However, incomplete traceroute paths are not uncommon, as some routers do not return ICMP replies. Currently, the number of such paths is 6-15% of the total, but this could increase.

Third, the rules in [1] for correcting mismatches are very liberal. Technically an IP-to-AS mapping assigns a *set* of AS's to a given prefix, although that set is often a singleton. In the heuristics of [1], no AS is ever deleted from the original assignment to a prefix. In addition, prefixes identified as IXP's are mapped to the set of *all* AS's, which means that we allow the IXP's to match *any* AS. Also, if two AS's are identified as "siblings", then if either one is assigned to a given prefix then both are. These rules thus do not always yield accurate and realistic mappings, and can gloss over distinctions that might otherwise be deduced, such as the precise set of AS's that connect to an IXP.

In contrast to previous work, we take a more systematic approach to correcting inaccurate IP-to-AS mappings. We retain the assumptions of [1] that (i) the AS-level routing and forwarding paths *usually* match and (ii) the IP-to-AS mapping derived from the BGP routing tables is *mostly* correct. Our algorithm identifies a small number of modifications to the initial IP-to-AS mapping that significantly improve the match rate. Compared to the heuristics in [1], we use a more powerful notion of "matching" and attempt to derive information from pairs of traceroute and BGP paths even when some mismatches are unavoidable under the current mapping. The *unavoidable errors* (defined later) are due to inaccuracy in IP-to-AS mappings or legitimate reasons such as routing anomalies or aggregation. To determine the unavoidable mismatches, we use dynamic programming to compute, for each pair of a traceroute path and the corresponding BGP path, a matching between elements of the two paths that *minimizes* the number of mismatches for that pair. Based on summary information from the set of optimal matchings, we then change the mapping so as to reduce the number of unavoidable errors, subject to constraints intended to keep the mapping realistic and force the process to converge when applied iteratively.

When applied to the initial IP-to-AS mapping derived from BGP tables, our algorithm reduces the fraction of mismatched paths from 15% to 5%, while only changing 2.9% of the assignments in the initial IP-to-AS mapping. This is in contrast to the results of [1], where 10% of the assignments were modified and the fraction of mismatched paths was only reduced to 9%. We show that our algorithm is robust and can yield near-optimal results even when the initial mapping is corrupted or when one reduces the number of probing sources or destinations and hence has fewer pairs to work with. This is due to the fact that we modify the mappings to correct the errors in the initial mappings by directly modeling the errors into the problem. For instance, when we reduce the set of destinations randomly so that 90% fewer probes are needed, we obtain a mapping that is almost as good as the one obtained using the full set. This is because our initial measurements cover a large number of destinations with similar AS paths. Moreover, our algorithm produces more fine-grained mappings than the one derived in [1], for instance retaining more singleton assignments and providing a clearer picture of the precise set of AS's connected to a given IXP.

The rest of the paper is organized as follows. Section II overviews our data collection methodology. In Section III, we formalize the problem of finding the number of unavoidable mismatches under a specified IP-to-AS mapping for a given path pair, and observe that a matching having only this number of mismatches can be found efficiently using dynamic programming. (The details of the algorithm are presented in Appendix A.) Section IV gives our procedure for improving an IP-to-AS mapping once we have computed the optimal matchings for all the pairs of traceroute and BGP paths. We thoroughly evaluate the effectiveness and accuracy of this approach in Section V, which also examines its robustness and proposes a technique to reduce the number of traceroute probings needed to construct IP-to-AS mappings. In Section VI we discuss how the changes introduced by our constructed mappings can at least be partially validated using publicly available information and local configuration data. Finally, we discuss related work in Section VII and conclude in Section VIII with a summary of our contributions and a discussion of ongoing research.

II. Data Collection

Designing a useful "AS-level traceroute" tool depends on having an accurate way to map the IP addresses of network equipment to the administering AS's. In this work, we focus on systematically constructing an accurate IP-to-AS mapping which can be used as input of AS-level traceroute tool.

We reuse the traceroute probes, BGP update messages, BGP routing tables, and reverse DNS lookups from eight vantage points (Table I) that were collected in [1]. The sites were chosen based on their topological diversity and the ability to collect both traceroute and BGP update data.

The data collection methodology is summarized here, while the details can be found in [1]. Starting with a list of routing table entries, we first identify the prefixes that cover the

Organization	Location	Dates in 2003	Upstream Provider (AS Number)	Compared Prefixes
AT&T Research (AS 6431)	NJ, USA	June 6-9	UUNET (701), AT&T (7018)	118345
UC Berkeley (AS 25)	CA, USA	June 6-8	Qwest (209), Level 3 (3356)	112120
PSG home network (AS 3130)	WA, USA	April 30 - May 8	Sprint (1239), Verio (2914)	117195
Univ of Washington (AS 73)	WA, USA	June 4-8	Verio (2914), Cable & Wireless (3561)	121660
ArosNet (AS 6521)	UT, USA	May 1-6	UUNET (701)	117323
Nortel (AS 14177)	ON, Canada	May 1-6	AT&T Canada (15290)	117966
Vineyard.NET (AS 10781)	MA, USA	June 4-9	UUNET (701), Sprint (1239), Level 3 (3356)	113114
Peak Web Hosting (AS 22208)	CA, USA	May 1-8	Level 3 (3356), Global Crossing (3549), Teleglobe (6453)	110388

TABLE I
EIGHT VANTAGE POINTS TO COLLECT TRACEROUTE AND BGP PATHS.

	Extracted Prefixes	Origin AS's	MOAS Prefixes
AS 6431	120997	15105	0
AS 25	124202	15213	0
AS 3130	121054	15086	0
AS 73	123583	15194	0
AS 6521	121096	15099	0
AS 14177	121135	15104	0
AS 10781	121669	15103	0
AS 22208	125050	15136	0
RouteViews	134095	15294	860
RIPE(00–08)	128960	15328	3400
SingAREN	6744	862	25
Potaroo	142348	16112	211
Verio	105381	13778	116
AT&T	128411	15171	109
Combined	203698	16367	8827

TABLE II
BGP TABLES FOR IP-TO-AS MAPPING AROUND MAY 2003

routable address space and then select two IP addresses within each prefix for traceroute probing. For each IP address we measure the forwarding path with traceroute and extract the BGP AS path from the routing table of the border router. We also collected BGP updates in MRT format through a BGP session with the same border router. We preprocessed the traceroute and BGP paths by discarding paths with BGP routing changes, private AS numbers, apparent AS loops, and Null AS paths. These were fewer than 0.6% of the total.

The eight traceroute and BGP data sets were collected between May and June in 2003. Table I provides details on these sets. The number of prefixes observed from each vantage point varies from 110K to 120K.

We construct an original mapping by combining BGP routing tables from multiple locations (listed in Table II) and extracting the last hop in the AS path (the "origin AS") for each entry. There are over 200K prefixes, along with over 16K origin AS's. The mapping assigns to each prefix the origin AS for it, or the set of origin AS's, if there is more than one. About 5% of the prefixes mapped to multiple origin AS's (MOAS's), with as many as 5 origin AS's assigned to some prefixes.

III. EVALUATING IP-TO-AS MAPPINGS

In this section we discuss our approach to evaluating a given mapping of IP prefixes to AS's by measuring how consistent it is with our given set of traceroute-BGP path pairs.

Based on the discussion in the previous section, a *mapping* A can be described by a set P_A of IP prefixes and, for each $x \in P_A$, a non-empty set $A(x)$ of AS's. Most such sets will be of size 1, although larger sets are possible if x corresponds to a MOAS. The set P_A is assumed to be complete in that every IP address has at least one prefix in P_A. For any IP address I, let $P_A(I)$ be the longest prefix in P_A that matches I. To simplify our notation in what follows, we extend the definition of A to IP's by letting $A(I)$ stand for $A(P_A(I))$.

Consider a traceroute-BGP path pair (p, q), where $p = (p_1, p_2, \ldots, p_n)$ is a sequence of IP's and $q = (q_1, q_2, \ldots, q_m)$ is a sequence of AS's. A function $a : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, m\}$ is a *matching* for p and q if $a(i) \leq a(i+1)$, $1 \leq i < n$. (Implicitly, we also view a as matching p_i with $q_{a(i)}$.) Given a mapping A, the *error* of a given matching a for p and q is

$$E_A(a, p, q) = \left| \left\{ i \leq n : q_{a(i)} \notin A(p_i) \right\} \right| + a(n) - |\{a(i) : 1 \leq i \leq n\}|$$

Note that there is a penalty of 1 both for matching an IP p_i to an AS not in $A(p_i)$ and for failing to match any IP to an AS q_j that precedes the last matched AS in q. We do not impose a penalty for failing to match AS's after the last matched AS since traceroute paths can often be truncated due to timeouts, etc.

The *unavoidable error* $E_A(p, q)$ for mapping A and pair (p, q) is defined to be

$$\min \left\{ E_A(a, p, q) : a \text{ is a matching for } (p, q) \right\}.$$

A mapping A is said to be *consistent* with a pair (p, q) if $E_A(p, q) = 0$. This suggests two metrics for evaluating the quality of a mapping A:

1) $|\{(p, q) : E_A(p, q) = 0\}|$, the total number of pairs with which A is consistent, and
2) $\sum_{(p,q)} E_A(p, q)$, the total number of unavoidable errors for A.

In this paper we shall consider both metrics. Their evaluation is facilitated by the fact that we can compute $E_A(p, q)$ efficiently using dynamic programming. In particular, given A, p, and q, we can in $O(nm^2)$ time find a matching a such that $E_A(a, p, q) = E_A(p, q)$. The details of the algorithm are

provided in Appendix A. When there is more than one optimal matching a, we break ties in favor of the one that matches IP's to the last possible AS in the path. This is motivated by the observation that mismatches that occur towards the end of the AS path due to aggregation should be resolved by mapping the IP address in question to the customer network, *i.e.,* to the AS later in the path.

The $O(nm^2)$ running time bound is quite small in our context, given that the average values of m and n are 3.8 and 7.4 respectively, with the maximum values encountered being 11 and 25. The algorithm takes only 0.2 milliseconds to complete for each traceroute-BGP pair on a 900 Mhz Sparc processor. However, to consider the entire data set – 1,860,448 pairs, it takes more than 6 hours. Running the algorithms on more vantage points and experimenting with different threshold values of the rules require even longer running time. This provides a strong motivation for reducing the data needed to obtain accurate results, as discussed later in Section V.

As an illustration of the effectiveness of the dynamic programming approach, consider what happens when we apply it to the initial mapping obtained from the BGP tables (in Table II) with the entire data set of about 1.8 million pairs. Whereas the heuristics in [1] was only able to identify roughly 79% of these pairs as consistent, we find that 85.3% of them are. (A major factor in this improvement was our success in handling pairs with truncated traceroute paths, which the heuristics of [1] could not deal with at all.) When we apply our approach using the heuristically optimized mapping based on the heuristics presented in [1] to the full data set, we find that 90.7% of the pairs are consistent. This figure is much closer to the figure reported in [1], roughly 91%, but that figure ignored the results for several classes of defective pairs due to problems in traceroute, while we include them because our dynamic programming algorithm allows us to deal with the defects, even though most continue to have unavoidable errors. For the first mapping, the total number of unavoidable errors was 410,357 while for the second it was 265,459, which in both cases was roughly 1.5 errors per inconsistent pair.

IV. Improving IP-to-AS Mappings

Our dynamic programming algorithm, by producing an optimal matching for each pair, can also be used to *improve* the mapping by helping us identify places where the mapping is accurate. For example, the mapping from the BGP tables assigns prefix 154.54.10.0/24 to AS2149, but in the 7972 pairs in which an IP address appears for which this is the longest prefix in P_A, our optimal matchings never match it to AS2149, but match it to AS174 91.8% of the time (even though there is a penalty involved). This suggests that the actual mapping should have been to the latter AS. Indeed, if we change the mapping accordingly, the optimal matchings for the relevant pairs now match 154.54.10.0/24 with AS174 100% of the time!

In this paper, we consider a simple scheme for improving mappings by systematically exploiting the information from our optimal matchings. Other schemes are possible, but the current results already show significant improvement over the previous approach and provide a starting point for other more sophisticated rules. Suppose we are given a mapping A, the results of computing optimal matchings under A for all pairs (p, q), and a subset of the prefixes designated as *confirmed MOAS's* (typically confirmed by external data). We "improve" A as follows.

Say a pair (p, q) is *good* if $E_A(p, q) \leq 2$. We restrict attention to those prefixes x that fail to match with a member of $A(x)$ in at least one good pair. (There is no need to change $A(x)$ for the other prefixes.) For each prefix $x \in P_A$, let S_x denote the set of all IP addresses whose longest prefix in P_A is x, and let n_x denote the number of good pairs involving members of S_x. For each AS y, let $n_x(y)$ be the number of good pairs in which our dynamic programming algorithm matches a member of S_x with y. Let z_0 be the AS currently assigned to x that has the smallest value of $n_x(y)$ (ties broken arbitrarily), and let $r_0 = n_x(z_0)/n_x$, the fraction of the good pairs in which x is matched to z_0. Similarly, let z_1 be the AS not assigned to x that has the largest value of $n_x(y)$, and let $r_1 = n_x(z_1)/n_x$, the fraction of the good pairs in which x is matched to z_1. Apply the first of the following four rules that is relevant (and only that rule) to x:

- **Rule 1: Delete from MOAS-pair**
  If $|A(x)| = 2$, x is not a confirmed MOAS, and $r_0 < 0.1$, delete z_0 from $A(x)$, making it a singleton.
- **Rule 2: Replace a singleton**
  If $A(x)$ is a singleton and $r_1 \geq 0.55$, set $A(x) = z_1$.
- **Rule 3: Create a MOAS-pair**
  If $A(x)$ is a singleton and $0.2 < r_1 < 0.55$, add z_1 to A(x), making it a pair.
- **Rule 4: Add to MOAS**
  If $|A(x)| \geq 2$ and $r_1 > 0.1$, add z_1 to $A(x)$.

Note that all the rules that add items to $A(x)$ must of necessity reduce the number of unavoidable errors. The rule that deletes items can increase the number of errors, but is included so as to reduce the probability of creating fictitious MOAS's. Similarly, the threshold ratios for additions to $A(x)$ restrain the creation and expansion of MOAS's. (If for every x we let $A(x)$ be the set of all AS's, we would get a mapping with no unavoidable errors, but this would provide no insight into the true correspondence between prefixes and AS's.) The particular thresholds in the rules were chosen with the above goals in mind and after looking at the results from preliminary matchings. Although the precise values of the thresholds are somewhat arbitrary, the results obtained using them indicate that they are quite effective. An obvious question for future research is whether fine-tuning these thresholds can lead to even better matchings.

The above scheme can only increase the size of $A(x)$ by one. This is motivated in part by the conservative arguments of the previous paragraph, but also by the fact that we envision applying the scheme iteratively, with corrections made in one step helping to prevent mistakes in later ones. Here is the iterative procedure we use:

1) Let A be our initial mapping.
2) Repeat until done:

a) Compute optimal matchings for the current mapping A and all traceroute-BGP path pairs.
b) Apply the improvement scheme to A.
c) If A remains unchanged, we are done.

3) For each prefix x with $|A(x)| > 2$, add to $A(x)$ all those AS's that were matched to x more than 2% of the time and more than 10 times.

Note that, given the rules in the scheme, it is unlikely that more than 10 iterations of the inner loop will be needed. This is because the last rule is the one that is most likely invoked over multiple iterations, and it cannot be invoked more 10 times given the threshold value, unless any deletion occurs. The final step is designed to expand the identified MOAS's of size greater than 2 to include most of the likely candidates for membership. This is still far more restrictive than the heuristics used in [1], which assumed that for every prefix x identified as a MOAS, $A(x)$ contained *all* the AS's.

In the next section we will discuss how the above procedure performs given various choices for the starting mapping and the set of traceroute-BGP path pairs.

V. Experiments

A. Experiment selection and design

In this section, we thoroughly evaluate the effectiveness and accuracy of the dynamic-programming-based approach on the data set in Table I. The results of several experiments are discussed to show its robustness against the changes in the initial mapping as well as the reduction in the number of BGP and traceroute path pairs. The former analysis is important as it may be difficult to collect BGP tables from a large number of vantage points to construct a complete and accurate initial mapping. The latter evaluation is critical for a practical AS-level traceroute tool, as one cannot afford to do large amounts of probing.

Table III illustrates the experimental design space along the dimensions of (1) initial mapping and (2) set of traceroute-BGP path pairs. In the base case for the initial mapping, we use the mapping obtained from the large collection of BGP tables shown in Table II and described in Section II. The second option for the initial mapping is the "heuristically optimized" mapping obtained in [1] using ad-hoc heuristics. The third option is obtained by randomly deleting the assignments for 10% of the prefixes in the BGP-table-based mapping and instead initially assigning these to a non-existent AS, thus simulating the effect of added noise and inaccuracy in the initial mapping.

The base case for traceroute pairs contains all 1.8 million pairs from our original set. To study the robustness against smaller data sets, our second and third cases use smaller sets obtained by deleting all pairs with certain sources/destinations. More details on the construction of these subsets will be presented when we describe the relevant results.

The prefixes in the initial BGP tables have varying lengths. Among the 200K prefixes, more than 60% them have length of 24 or longer. For the purpose of applying our dynamic programming algorithm and improvement scheme, we subdivide all prefixes that are in the initial mapping and encountered in the traceroute data into /24's with one exception described below. The imposed limit at length 24 is motivated by the fact that most ISP's filter out prefixes longer than /24. This finer-grained partition of the IP-address space is motivated by the hope of allowing the mapping to account for the effects of address aggregation. A large supernet, for instance the one corresponding to a /16 prefix, might be assigned to a single AS by our initial mapping while it matches multiple AS's when we correlate traceroute paths with the BGP AS paths. If we divide the /16 into the corresponding /24's, the conflicting assignments may be significantly reduced without having to create new MOAS's – each /24 might match to a single AS in most of the pairs (although different /24's might match to different AS's).

	traceroute-BGP pairs		
Initial mapping BGP Tables –	Full data set	Omitting Sources	Omitting Destinations
Unmodified	DP-BGP	DP-OS	DP-OD
Heuristically Optimized	DP-HO	-	-
Omitting Assignments	DP-OM	-	-

TABLE III
Experiment design space and selected experiments

The initial mapping for the newly created /24's is inherited from the mapping for the parent supernet. Any MOAS prefix in the original table is considered to be a "confirmed MOAS" in the sense of Section IV since the origin AS's in the BGP tables are deemed to be reasonably accurate in this regard. However, a newly created /24 of such a MOAS prefix is not considered to be a "confirmed MOAS," thus allowing for potential deletions from the assigned set when we apply our improvement rules. For ease of programming, we carry out the subdivision into /24's for all encountered prefixes except those that already contain a subnet with length 24 or longer. Our results might improve if we carried out the subdivision for all prefixes, but very few prefixes and pairs are affected by this exception. The total number of prefixes in our experiment after subdivision into /24's is 105,068, more than half of which are newly created.

Recall that our improvement rules ignore traceroute-BGP path pairs with more than 2 unavoidable errors. Based on the initial BGP table mapping, 85.30% of the pairs have no errors, 10.67% have a single error, and 2.28% have 2 errors. The remaining pairs all have more than 3 unavoidable errors but make up only 1.75% of the pairs. After going through the dynamic programming based optimization, only 0.55% of the pairs have more than 2 errors. Most likely these represent defective pairs where the two paths fail to correspond due to forwarding path changes during traceroute probing, BGP routing changes, or other causes.

B. DP evaluation: DP-BGP and DP-HO

The dynamic programming algorithm performs optimization on each pair of traceroute and BGP AS path. Subsequently, we re-optimize the assignment for each IP prefix by tabulating the results of the derived matchings for the prefix and applying the four rules of Section IV. We then iterate the procedure.

Table IV summarizes the results for all five experiments: DP-BGP, DP-HO, DP-OM, DP-OD, and DP-OS. It shows the benefit of iterative optimization. Each row is for one iteration of the improvement procedure. Iteration 0 evaluates the initial mapping. In general, Iteration i evaluates the mapping created in Iteration $i-1$. Recall that the final iteration simply expands all MOAS's with more than 2 elements to contain all AS's that match the prefix at least 2% of the time.

We now describe the meaning of each column in the table. The first column is the iteration number, 0 being the starting point. The "Mismatch" column indicates the percentage of traceroute-BGP path pairs with at least one unavoidable error. "Error count" is the total number of unavoidable errors in all the pairs. An error is either a mismatch between an AS in the BGP path and the assigned AS of a traceroute IP address using the dynamic programming algorithm or a skipped AS in the middle of a BGP path. Note, we do not penalize for AS's skipped toward the end of BGP path as traceroute may not reach the destination AS. Most pairs with unavoidable errors have 1 or 2 errors. The column labeled "Matched prefixes" presents the percentage of prefixes encountered in the data set that do not have conflicting matchings. The number of changed assignments in this iteration is shown in "New maps" column. The "Rule" columns contain the number of errors that are expected to be corrected by application of the corresponding rule during the current iteration. The values in parentheses indicate the number of prefixes to which the rule is applied during the current iteration. Only the first rule can increase the number of errors as it deletes an AS from an AS pair in the mapping. The total number of errors to be corrected is shown in the "Expected gain" column. The last column, marked by "Actual gain," is the actual reduction in the total number of unavoidable errors compared to the previous iteration. Note, this number is at least as large as the value in the "Expected gain" column in the previous row. It is often significantly larger in the first several iterations, because the dynamic program re-optimizes the matchings at each iteration and the reassignment of one IP prefix may help us correct other errors as well.

We now focus on the results of the first experiment: DP-BGP. All 5 experiments conducted converge within 6 iterations. The last iteration is where the final step (4) of the iterative procedure adds likely AS's to some of the MOAS assignments. DP-BGP uses the initial IP-to-AS mapping based on 200K prefixes from BGP tables in Table II. Initially, only 14.70% of the pairs have unavoidable errors, indicating that the BGP tables provide a fairly good starting point. Most modifications to the mapping are performed in the first iteration, as is the case in the other four experiments. Typically the most common change in assignment is replacement (Rule 4). For example, close to $2,000$ prefixes have their assignments replaced in DP-BGP during the first iteration. The replacement rule usually corrects for the effects caused by sharing of addresses between sibling AS's, inaccuracy of ICMP source IP address, and routing anomalies. This provides a good indication that the dynamic programming algorithm is performing fine-grained correction. The previous study [1] did not consider replacement as a way to correct IP-to-AS mappings, as it did not optimize over all the paths systematically. Our algorithm is a significant improvement in this regard. We observe that the "Actual gain" is almost always much larger than the "Expected gain" during the first several iterations. For instance, the gain obtained in Iteration 1 is $197,477$, which is 35% more than the expected gain of $146,086$. This shows that value of reoptimizing and iterating.

Based on the results of DP-BGP, we conclude that the dynamic programming algorithm is very effective in reducing the mismatches between traceroute and BGP AS paths, while making a relatively small number of corrections in the mapping. As shown, the number of paths with errors is reduced by more than 60% from 14.7% to 5.23%. Similarly, the total number of errors is cut by close to 65%. The number of prefixes with errors is reduced by 37%. All this is accomplished by making changes to the assignments for only about $3,000$ prefixes. A close look at the distribution of the changes made based on each rule indicates that about 59% of the corrections made are replacements, 33% involve creating a pair of AS's, 8% involve adding an AS to a MOAS, and a negligible number involve deletion from a pair of AS's.

To compare our algorithm with the previous study [1], look first at the final line for DP-BGP and the first line for DP-HO, which evaluates the heuristically obtained mapping from that study. There are 9.27% mismatched pairs initially as shown in Iteration 0 for DP-HO, as compared to the 5.23% for the final DP-BGP mapping. Moreover, our approach, when applied to the heuristically obtained mapping, reduces the number of mismatched pairs even further, to 3.08%. This further improvement can probably be attributed to the fact that the heuristically obtained mapping is more liberal in its assignment of MOAS's than is the BGP-table-based mapping. It allows identified IXP's to map to any AS and any prefix that maps to one of a pair of sibling AS's is assumed to map to both. It is noteworthy that during Iteration 0 of DP-HO, 165 prefixes are affected by Rule 1 which deletes an AS from an AS pair. This is a significantly higher number than that for DP-BGP and hence an indication that the liberality of the heuristically obtained mapping is not totally warranted.

Table V compares the distribution of mapping sizes for all the 105,068 prefixes encountered in the data set for DP-HO and DP-BGP. The mapping size is the number of AS's a prefix maps to. A small number of prefixes in DP-HO map to all AS's as they are identified to be IXP's. We note that significantly more prefixes map to a single AS in DP-BGP than in DP-HO – 93.23% compared to 84.31%, as DP-BGP provides more fine-grained correction and is less liberal in adding AS's to mappings.

C. Robustness in initial IP-to-AS mapping: DP-OM

The third experiment DP-OM in Table IV illustrates the resilience of our algorithm with respect to errors in the initial IP-to-AS mapping. The errors are introduced by assigning a randomly chosen 10% of the prefixes to a single non-existent "dummy" AS. The effect is quite obvious as more than 50% of the pairs contain unavoidable errors in Iteration 0, and the number of errors is more than four times that for the initial BGP-table-based mapping! Remarkably, the iterative

i	Mis-match	Error count	Matched prefixes	New maps	Rule 1 gain (map)	Rule 2 gain (map)	Rule 3 gain (map)	Rule 4 gain (map)	Expected gain	Actual gain
DP-BGP: using initial BGP-table-based mapping, 1,860,448 traceroute measurements										
0	14.70%	410357	90.82%	2853	-8 (6)	74071 (1850)	43371 (862)	28652 (135)	146086	0
1	8.17%	212880	93.77%	442	-17 (3)	8908 (120)	3697 (221)	10271 (98)	22859	197477
2	7.42%	187121	94.23%	47	0 (0)	16 (2)	1961 (16)	7010 (29)	8987	25759
3	6.96%	177297	94.26%	10	0 (0)	6 (2)	4 (1)	451 (7)	461	9824
4	6.94%	176833	94.26%	2	0 (0)	0 (0)	4 (1)	2 (1)	6	464
5	6.94%	176827	94.26%	1	0 (0)	0 (0)	0 (0)	2 (1)	2	6
6	6.94%	176825	94.26%	27	0 (0)	0 (0)	0 (0)	32535 (27)	32535	2
7	5.23%	143697	94.27%	-	-	-	-	-	-	33128
DP-HO: using Heuristically Optimized mapping, 1,860,448 traceroute measurements										
0	9.27%	265459	91.67%	2539	-122 (165)	27286 (1298)	17347 (592)	46891 (484)	91402	0
1	4.96%	142129	94.19%	323	0 (5)	6691 (68)	1566 (149)	7112 (101)	15369	123330
2	4.54%	125688	94.53%	29	0 (1)	12 (2)	119 (7)	3552 (19)	3683	16441
3	4.36%	121945	94.55%	3	0 (0)	0 (0)	4 (1)	114 (2)	118	3743
4	4.35%	121827	94.55%	1	0 (0)	0 (0)	0 (0)	2 (1)	2	118
5	4.35%	121825	94.55%	1	0 (0)	0 (0)	0 (0)	2 (1)	2	2
6	4.35%	121823	94.55%	64	0 (0)	0 (0)	0 (0)	24428 (64)	24428	2
7	3.08%	96786	94.59%	-	-	-	-	-	-	25037
DP-OM: omitting 10% of randomly selected BGP-table-based mapping, using 1,860,448 traceroute measurements										
0	54.11%	1700492	81.70%	9153	-4 (5)	728816 (8025)	83330 (1018)	17197 (105)	829339	0
1	17.98%	476275	91.49%	1866	0 (9)	35649 (995)	45904 (483)	41894 (379)	123447	1224217
2	12.01%	289402	93.30%	509	0 (1)	6582 (226)	4888 (74)	23019 (208)	34489	186873
3	10.15%	247750	93.73%	97	0 (0)	131 (21)	237 (16)	4176 (60)	4544	41652
4	9.88%	242295	93.79%	23	0 (0)	39 (6)	82 (3)	232 (14)	353	5455
5	9.86%	241919	93.81%	1	0 (0)	0 (0)	0 (0)	12 (1)	12	376
6	9.86%	241907	93.81%	159	0 (0)	0 (0)	0 (0)	60911 (159)	60911	12
7	6.57%	175977	93.88%	-	-	-	-	-	-	65930
DP-OD: omitting traceroute probing destinations, using 242,836 traceroute measurements										
0	13.10%	46653	91.37%	1169	0 (0)	7448 (775)	4566 (337)	3406 (57)	15420	0
1	7.73%	25728	93.97%	198	0 (2)	895 (49)	492 (83)	1786 (64)	3173	20925
2	6.89%	22265	94.33%	42	0 (0)	1 (1)	102 (8)	1034 (33)	1137	3463
3	6.46%	21081	94.39%	16	0 (0)	1 (1)	11 (3)	146 (12)	158	1184
4	6.41%	20917	94.42%	4	0 (0)	0 (0)	6 (2)	58 (2)	64	164
5	6.38%	20853	94.42%	1	0 (0)	0 (0)	0 (0)	1 (1)	1	64
6	6.38%	20852	94.42%	9	0 (0)	0 (0)	0 (0)	3986 (9)	3986	1
7	4.79%	16749	94.42%	-	-	-	-	-	-	4103
F	7.12%	190511	91.93%	-	-	-	-	-	-	-
DP-OS: omitting traceroute probing sources, using 938,377 traceroute measurements										
0	18.45%	264925	91.56%	2721	-4 (4)	35063 (1753)	26137 (835)	19322 (129)	80518	0
1	11.61%	149943	94.51%	436	-9 (3)	7683 (135)	8422 (207)	7935 (91)	24031	114982
2	9.88%	124373	94.98%	56	-2 (3)	7 (2)	431 (19)	6340 (32)	6776	25570
3	9.20%	116817	95.02%	13	0 (0)	14 (4)	476 (4)	85 (5)	575	7556
4	9.14%	116217	95.03%	1	0 (0)	4 (1)	0 (0)	0 (0)	4	600
5	9.14%	116213	95.04%	17	0 (0)	0 (0)	0 (0)	31783 (17)	31783	4
6	5.94%	84194	95.04%	-	-	-	-	-	-	32019
F	6.34%	165630	94.09%	-	-	-	-	-	-	-

TABLE IV

DP-BGP, DP-HO, DP-OM, DP-OD, DP-OS: ITERATIVE OPTIMIZATION USING DYNAMIC PROGRAMMING.

Mapping size	DP-HO	DP-BGP
all AS's	56 (0.05%)	0
1	88583 (84.31%)	97958 (93.23%)
2	16278 (15.49%)	6762 (6.44%)
3	135 (0.13%)	289 (0.28%)
4	13 (0.01%)	42 (0.04%)
5	3 (0.00%)	4 (0.00%)
6-15	0	13 (0.01%)

TABLE V

DISTRIBUTION OF MAPPING SIZE: NUMBER OF AS'S EACH PREFIX MAPS TO.

algorithm steadily reduces the errors in the mapping and converges with only 6.57% of the pairs having unavoidable errors, fairly close to the final result in DP-BGP. We observe that, as with DP-BGP, the replacement rule (Rule 2) is the most frequently invoked, although here with much higher frequency. Typically, the rule when invoked replaces the dummy AS with the correct mapping.

D. Robustness in the set of pairs used: DP-OD, DP-OS

So far all our analysis is based on the entire traceroute and BGP data set collected in the previous study [1], where an extensive set of traceroute measurements were performed as a first attempt to study techniques needed for an accurate AS-level traceroute tool. In that study, traceroute targets were selected based on the local BGP table; two IP addresses were

probed from each prefix, yielding at least 200,000 IP addresses from each vantage point chosen. From all eight vantage points, we have over 1.8 million traceroute results. Clearly, if one can reduce the number of probes without sacrificing the accuracy of the resulting mapping, it becomes more practical to perform the probing regularly, as will be necessary if we want to keep our mapping up to date. Minimizing the number of probes is also an important requirement for the scalability and efficiency of our desired AS-level traceroute tool. Previous work [5] suggested that increasing probing sources has less benefit compared to adding probing destinations for the purpose of discovering network topology. We study the effect of reducing both probing sources and destinations on our final mapping.

Each traceroute measurement provides the correlation between a prefix level path and its corresponding BGP path from the local BGP table. Some quick analysis shows that there is significant redundancy in our overall set of pairs, and so smaller sets might suffice. One way to reduce redundancy is to simply probe one destination IP address for each unique BGP path at a given source, even if that path corresponds to multiple prefixes. We simulate this by randomly selecting just one of our pairs for each BGP path at each source. This reduces our work: DP-OD uses only 242,836 pairs – 13% of the measurements in the full data set. However, it sacrifices coverage of the address space, since many measurements sharing the same BGP path to different IP addresses differ in their prefix-level paths. Applying our dynamic-programming-based approach to this set of pairs and the BGP-table-based mapping (DP-OD in Table IV) we get surprisingly good results. The first eight rows indicate iterative improvement in reducing errors and mismatched paths similar to that for DP-BGP. However, we should focus our attention on the last row, marked with "F." This shows the result of taking the final mapping obtained from the reduced set of pairs and evaluating it against the entire data set of all 1.8 million traceroute-BGP pairs. The result is quite close to the final result in DP-BGP with only slightly more errors and mismatched pairs.

DP-OD focuses on reducing the probing destinations. It is equally important to understand the impact of probing sources on the usefulness of measurement data. To study this impact, we picked four vantage points (AT&T Research, Univ of Washington, Nortel, and Peak Web Hosting) out of the eight locations in Table I, based on their diversity in topology and network connectivity. This roughly halved the numbers of measurements and pairs. The results are included under DP-OS in Table IV. Once again, the final matching, when evaluated against the full set of pairs, is almost as good as that obtained using the full set. Indeed, it is slightly better than that obtained for DP-OD, perhaps reflecting the fact that here we used a bigger subset of pairs. Both results together, however, help make the case that our approach is robust with respect to the set of input pairs we use, although at least a small price must be paid for using less complete sets. Note also that in both cases we still get substantially fewer mismatches than did the heuristically optimized mapping of [1], even though the latter used the full set of pairs.

Mapping	Identical	Disjoint	Comparison target
DP-BGP	97.10%	1.76%	BGP
HO-BGP	90.00%	0.00%	BGP
DP-HO	90.28%	0.04%	DP-BGP
DP-OM	95.84%	3.40%	DP-BGP
DP-OD	97.80%	1.16%	DP-BGP
DP-OS	99.56%	0.10%	DP-BGP

TABLE VI
COMPARING THE SIMILARITY OF NEWLY CREATED IP-TO-AS MAPPINGS.

Category	Existing prefixes	Created /24's
Rule 1: Delete from MOAS-pair	0.00%	0.23%
Rule 2: Replace a singleton	12.92%	46.00%
Rule 3: Create a MOAS-pair	10.69%	25.34%
Rule 4: Add to MOAS	1.77%	3.05%

TABLE VII
DP-BGP: DISTRIBUTION OF FINAL 3050 MAPPING CHANGES.

E. Comparing all experiments

We now compare the mappings resulting from all five experiments (DP-BGP, DP-HO, DP-OM, DP-OD, and DP-OS), the initial BGP-table-based mapping (BGP), and the heuristically optimized mapping of [1] (HO-BGP). Table VI shows for a given combination of mapping and comparison target, how many assignments are the same and how many are completely disjoint. Some of the mappings may overlap and constitute the third case. Let us first look at how DP-BGP and HO-BGP compare to BGP, the mapping from which they are both derived. Surprisingly DP-BGP modifies the BGP assignments for less than 3% of the prefixes compared to 10% for HO-BGP. This comparison shows that our dynamic programming based algorithm makes significantly fewer changes to BGP, while reducing the number mismatched pairs by a significantly larger amount. (There are no disjoint prefixes between HO-BGP and BGP, as HO never deletes any AS's from an assignment.)

The rest of the table shows comparisons with DP-BGP, to illustrate the effect of perturbing the initial mapping or reducing the set of pairs. The mappings from all four experiments – DP-HO, DP-OM, DP-OD, and DP-OS agree with that for DP-BGP in over 90% of the assignments. DP-HO is the worst, as it starts from HO-BGP rather than BGP. DP-OS does remarkably well, as the agreement is close to 100%. This explains why the mapping works so well over the entire data set. In the future, we plan to develop a scheme combining DP-OD and DP-OS to further reduce probing.

VI. VALIDATION

In general, validating an IP-to-AS mapping is hard due to the lack of access to information in network configurations. We must rely on information obtained from other sources, both public and internal, which themselves are incomplete. Also, since our approach is different in nature from previous ones, different sorts of validation are needed. An important distinction between our new algorithm and previous work is that the changes to the IP-to-AS mappings are all based on

prefixes and what our dynamic program matches them to, and not directly based on inferred relationships between AS's or on inferred IXP's. Therefore, our validation is based on finding explanations for the changes rather than on confirming the inferred relationships. Typically, one type of change can have several explanations.

As in [1], we first use router configuration data from AT&T's network, AS 7018, and the $whois$ entry for the corresponding AS's (using organization names) to validate changes of IP-to-AS mappings where AS 7018 is involved. We found a total of 54 such cases. Recall that the IP-to-AS mappings resulting from the dynamic-programming-based approach could have two categories of prefixes: prefixes existing in the original IP-to-AS mapping and newly created /24 prefixes. Changes can be applied to either an existing prefix or a new /24 prefix.

As seen in Table VII, replacement is the most common change applied to an IP-to-AS mapping. Mismatches are fixed by replacing the AS in the original mapping, commonly by its sibling or customer AS. Sibling AS's are owned and managed by a single organization. They may share address spaces, with one AS numbering some of its equipment using part of an address block originated by another. In addition, an AS does not necessarily announce the addresses assigned to its equipment via BGP. Sometimes these addresses fall into larger address blocks originated by its sibling or provider.

In our validation, we consider the replacement 7018 by another AS X to be valid if AS X is a sibling or a customer of AS 7018. Using the configuration data, we are able to verify 18 out of 22 such cases; they are all customers of AS 7018. The replacement of another AS Y by AS 7018 is valid if AS Y is a sibling or customer of AS 7018. We can verify 7 out of 10 such replacements. The cases we were unable to validate could be due to errors in the inference, but are most likely due to incomplete or not up-to-date router configuration data.

Our algorithm could also produce new MOAS prefixes. These are commonly due to sibling AS's, customer AS's, or IXP's. In all the cases that correspond to customers of AS 7018, the prefix was specified in a static route associated with one or more access links to a customer. This means AS 7018 originated the route to this prefix on behalf of a customer; thus the prefix referred not to the equipment inside the backbone but rather to addresses in the customer's network. We found 16 newly identified MOAS prefixes involving AS 7018, 14 of them correspond to siblings or customers of AS 7018. There are also 6 modified MOAS prefixes, 2 of them are IXP's connected with AS 7018 and the rest are siblings or customers of AS 7018. Overall, we confirm 45 out of 54 changes (83%) to the IP-to-AS mappings. Though we are not able to validate all the cases, we did not find any evidence that contradicts the changes that the dynamic programming algorithm applied.

For the rest of the inferred MOAS prefixes that are not verified using local configuration data, we queried $whois$ using the AS number or prefix to see if the description contained the words "exchange point" or "Internet exchange". This check succeeded for 24 of our 1246 MOAS inferences. Then, we compared our results against a list of known IXP's that was derived from [6] and [7]. This confirmed 24 of the inferences, including well-known IXP's over the entire world such as Internet Exchange in Japan, Vienna, Paris, Hong Kong, New York, and San Jose. For example, the fifth hop in the traceroute example in Figure 1 is verified to be the Calren exchange point. Comparing to the results in [1], the new approach identified 11 additional IXP's among the MOAS prefixes. Most of them are in Europe or of small size in the U.S., among them are IXP's in Vienna, Switzerland, France, London, San Diego, New York, Chicago, and Miami. In the previous approach, the fan-in and fan-out AS count for these prefixes is less than 2, causing them to be missed. Using both methods, we identify 38 of the inferred MOAS prefixes as IXP's. We also randomly sampled 10 out of the remaining 1208 MOAS prefixes. We found 4 that appear to have similar names and are likely siblings, 3 that appear to have a customer provider relationship since one of the AS's is a tier-1 ISP and the address space has been divided into smaller blocks, and the last 3 cases are unclear based on registry name information.

With more complete data sources, a more thorough validation might be possible, but even the limited amount presented here, together with our mismatch statistics, supports the hypothesis that our dynamic-programming-based approach produces an IP-to-AS mapping of substantially improved quality and can be quite effective in the context of an AS-level traceroute tool.

VII. Related Work

Recent measurement studies have quantified the differences between BGP and traceroute AS paths. The analysis in [8] showed that these differences have a significant impact on the characterization of the Internet topology. The work in [9] used publicly-available data (such as *whois*, lists of known IXP's, and other Web sites) to test the hypothesis that many of the mismatches stem from IXP's and siblings. To improve the accuracy of AS graphs derived from traceroute, the work in [10] proposed techniques that identify border routers between AS's to correct mistaken AS mappings; this is an alternate approach that handles some of the inaccuracy introduced by IP-to-AS mappings derived from BGP tables. Traceroute data have been used in other studies that measure router-level topologies and map routers to AS's [11], [12]. Except for handling certain traceroute anomalies such as unmapped IP addresses, these studies did not focus on improving the accuracy of the IP-to-AS mapping derived from the BGP routing tables.

Focusing solely on BGP AS paths, the work in [13], [14] presented algorithms for inferring AS-level commercial relationships, including siblings; however, these studies did not consider the influence of sibling AS's on the accuracy of traceroute AS paths. Previous work [1] relied on traceroute paths and BGP updates collected from multiple vantage points to a large number of destinations throughout the Internet and proposed heuristics for *identifying* IXP's, siblings, and other causes of mismatches to improve the IP-to-AS mapping applied to the traceroute paths.

VIII. Conclusions

This work attempts to provide a systematic solution to the problem in [1]. In contrast to previous work, we formulate

the problem of evaluating an IP-to-AS mapping in terms of a collection of optimization problems over the set of path-pairs, problems that can be solved efficiently by dynamic programming. We also provide an iterative scheme for improving a mapping, based on the solutions to all the path-pair matching problems. Together, these yield a robust algorithm for constructing near-optimal IP-to-AS mappings and identifying accurate AS-level traceroute paths. The techniques presented in this work can exploit all the path-pairs including those with incomplete paths. They do not rely on the understanding of common operational practices or depend on additional data sources, although based on our limited validation studies, the mappings they automatically produce seem to be effective at revealing relationships that previously could only be inferred by detailed study of network data.

Our ultimate goal is to build a scalable and accurate AS-level traceroute tool. This would help network operators to diagnose routing and performance problems. Researchers could also use it to build a more accurate Internet topology map and to help deduce Internet path properties in network tomography, for example.

In our ongoing research, we are addressing the following issues. First, our algorithm assumes that the IP-to-AS mapping is stable. We are investigating how frequently the mapping changes and how to update the mapping upon discovering such changes. Second, we are investigating the impact of changing the set of rules we use to improve IP-to-AS mappings. Although our experimental results show that the current set of rules work very well, we are exploring additional or alternative rules that might further reduce the mismatch ratio of our final mapping. Third, we are continuing to investigate the trade-off between the nature and number of probings made and the quality of the mappings derived from the resulting set of path-pairs, since the major effort in this whole process remains the collecting of pairs.

REFERENCES

[1] Z. Morley Mao, Jennifer Rexford, Jia Wang, and Randy Katz, "Towards an Accurate AS-level Traceroute Tool," in *Proc. ACM SIGCOMM*, September 2003.

[2] "Nanog traceroute," `ftp://ftp.login.com/pub/software/traceroute/`.

[3] "Prtraceroute," `http://www.isi.edu/ra/RAToolSet/prtraceroute.html`.

[4] 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 *Proc. Internet Measurement Workshop*, November 2001.

[5] Paul Barford, Azer Bestavros, John Byers, and Mark Crovella, "On the Marginal Utility of Network Topology Measurements," in *Proc. Internet Measurement Workshop*, November 2001.

[6] "Packet Clearing House," `http://www.pch.net/resources/data/exchange-points/`.

[7] Elena Silenok, "Study of Internet eXchange Points," .

[8] Lisa Amini, Anees Shaikh, and Henning Schulzrinne, "Issues with inferring Internet topological attributes," in *Proceedings of SPIE*, July 2002, vol. 4865.

[9] Young Hyum, Andre Broido, and k claffy, "Traceroute and BGP AS Path incongruities," CAIDA Technical Report, March 2003. `http://www.caida.org/outreach/papers/2003/ASP/`.

[10] Hyunseok Chang, Sugih Jamin, and Walter Willinger, "Inferring AS-level Internet topology from router-level path traces," in *Proc. Workshop on Scalability and Traffic Control in IP Networks, SPIE ITCOM Conference*, August 2001.

[11] Ramesh Govindan and Hongsuda Tangmunaraunkit, "Heuristics for Internet map discovery," in *Proc. IEEE INFOCOM*, 2000.

[12] Neil Spring, Ratul Mahajan, and David Wetherall, "Measuring ISP topologies with Rocketfuel," in *Proc. ACM SIGCOMM*, August 2002.

[13] L. Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM Trans. Networking*, December 2001.

[14] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H. Katz, "Characterizing the Internet hierarchy from multiple vantage points," in *Proc. IEEE INFOCOM*, June 2002.

APPENDIX

A. Dynamic Programming Details

Let A be a mapping with associated prefix set P_A, where A assigns a set $A(x)$ of AS's to each prefix $x \in P_A$. Suppose we are given a pair (p, q), where $p = (p_1, p_2, \ldots, p_n)$ is a sequence of IP-addresses in P_A and $q = (q_1, q_2, \ldots, q_m)$ is a sequence of AS's. Our goal is to find a matching a which minimizes $E_A(a, p, q)$, as defined in Section III.

For any IP-address p_i and AS q_j, let $c(p_i, q_j)$ be the penalty for matching p_i to q_j. In our case $c(p_i, q_j)$ is 0 if $q_j \in A(P_A(p_i))$, where $P_A(p_i)$ is the longest prefix in P_A that matches p_i, and 1 otherwise. The following algorithm will however work for arbitrary choices of c. If a is a matching for (p, q) and $1 \le h \le n$, define

$$\begin{aligned} C(a,p,q,h) &= \sum_{i=1}^{h} c\Big(p_i, q_{a(i)}\Big) \\ &+ \left(a(h) - \Big|\{a(i) : 1 \le i \le h\}\Big|\right). \end{aligned}$$

This is the cost of the matching, restricted to the first h prefixes in p. Then, for $1 \le i \le n$ and $1 \le j \le m$, define $B(p, q, i, j)$ to be the minimum value of $C(a, p, q, i)$ over all matchings a with $a(i) = j$.

Note that $\min\{B(p, q, n, j) : 1 \le j \le m\}$ is the optimal cost of a matching for p and q, under the assumption that there is no penalty for failure to match AS's after $a(n)$. We can compute this minimum using straightforward dynamic programming. The initial conditions are

$$\begin{aligned} B(p,q,1,j) &= c(p_1, q_j) + j - 1, \quad 1 \le j \le m. \\ B(p,q,i,1) &= \sum_{h=1}^{i} c(p_h, q_1), \quad 1 \le i \le n. \end{aligned}$$

For the recurrence relation, we have that for $2 \le j \le m$ and $2 \le i \le n$,

$$\begin{aligned} B(p,q,i,j) &= c(p_i, q_j) \\ &+ \min_{1 \le h \le j} \Big\{ B(p,q,i-1,h) + \max(0, j-1-h) \Big\} \end{aligned}$$

These can be computed in order of increasing j, and for each j in order of increasing i. In case of ties, we choose the maximum h.

The actual algorithm for computing the matching will of course have to keep a record of which option was the minimum for each i, j:

Let $M(p, q, i, j)$ be the index of the AS assigned to the $(i-1)th$ prefix under the assignment determining $B(p, q, i, j)$. In other words, $M(p, q, i, j)$ is that h, $1 \le h \le j$, that minimizes $B(p, q, i-1, h) + \max(0, j-1-h)$.

The optimal assignment is then computed as follows:

$$\begin{aligned} a(n) &= \text{argmin}\Big\{B(p,q,n,j) : 1 \le j \le m\Big\} \\ a(i) &= M\Big(p,q,i+1,a(i+1)\Big), \quad 1 \le i < n \end{aligned}$$

$argmin\{f(j) : 1 \le j \le m\}$ is the value of j that minimizes $f(j)$. Note that the running time for this algorithm is $O(nm^2)$. A more complicated algorithm can reduce this to $O(nm)$. Given its higher constant-factor overhead and the fact that m was so small in our instances, we did not implement this version, but present its details below as they may be of interest to future researchers.

We now use two variables in our recurrence. The first is simply the variable $B(p,q,i,j)$ from our first algorithm, which we will now call $B_1(p,q,i,j)$. The second, $B_2(p,q,i,j)$, is defined to be the minimum value of $C(p,q,a,i)$ over all matchings a for which $1 \le a(i) < j$ and $a(i+1) \ge j$. The initial conditions for B_1 are the same as before, while those for B_2 are

$$\begin{aligned} B_2(p,q,1,j) &= \min_{0 \le h < j}\Big(c(p_1,q_h) + j - 1 - h\Big) \\ B_2(p,q,i,1) &= i \end{aligned}$$

for $1 \le j \le m$ and $1 \le i \le n$. The recurrence relations for B_1 and B_2 now become

$$\begin{aligned} B_1(p,q,i,j) &= c(p_i,q_j) \\ &\quad + \min\Big(B_1(p,q,i-1,j), B_2(p,q,i-1,j)\Big) \end{aligned}$$

$$\begin{aligned} B_2(p,q,i,j) &= \\ &\quad \min\Big(B_1(p,q,i,j-1), B_2(p,q,i,j-1) + 1\Big) \end{aligned}$$

In order to construct the matching we store new variables $R_1(p,q,i,j)$ and $R_2(p,q,i,j)$, where $R_h(p,q,i,j)$ is the index (1 or 2) of the function that provided the minimum in computing $B_h(p,q,i,j)$. We deduce the matching from these variables as follows:

As before, we let $a(n)$ be the j that minimizes $B_1(p,q,n,j)$. Given that $a(i) = j$ has been computed and $i > 1$, we compute $a(i-1)$ as follows.

1) If $R_1(p,q,i,j) = 1$, set $a(i-1) = j$.
2) Otherwise,
   a) Set $h = j$.
   b) While $R_2(p,q,i,h) = 2$, set $h = h - 1$.
   c) Set $a(i) = h - 1$.