Representing the Internet as a Succinct Forest

Jim Gast [1], Paul Barford [2]

University of Wisconsin - Platteville, University of Wisconsin - Madison

Abstract

Effective placement of resources used to support distributed services in the Internet depends on an accurate representation of Internet topology and routing. Representations of autonomous system (AS) topology derived solely from routing tables show only a subset of the connections that actually get used. However, in many cases, missing connections can be discovered by simple traceroutes. In addition, the differences between customer-to-provider links, peer-to-peer links, and sibling-to-sibling links are useful distinctions. We first apply a clustering algorithm to simplify the AS graph to an AS forest. Then we use two complementary mechanisms to improve accuracy of an AS forest as a predictor of packet paths. One mechanism uses recent insights that packets flow unidirectionally across customer-provider inter-AS links. Annotations are added to the AS forest to indicate links that appear to be peering versus those that appear to be customer-provider links. The other mechanism provides links between trees by remembering the most recently seen similar traceroute.

1 Introduction

Deploying services broadly across the Internet can provide additional capabilities, improve response times, reduce network congestion and reduce load over specific, crowded links. Effective placement of resources such as performance measurement nodes, proxy caches, application layer multicast distribution points, and anomaly detection engines would provide more accurate and more detailed services.

In this paper, we address the problem of effective resource deployment using client clustering at the AS level. We argue that deployment at the AS level is

[1] Research supported by the Anthony C. Klug fellowship in Computer Science
[2] Research supported by NSF ANI-0085984

sufficient and appropriate for most services since routing within ASes is typically very efficient, and consideration of this problem at the router level makes it intractable for both the purpose of understanding topology and application of resource placement algorithms. We present a new method of clustering that generates a forest of trees of ASes. Branches of the tree form progressively smaller clusters, where each branch consists of ASes that are topologically near and over which packets are most likely routed. Building and refining the AS forest consisted of three steps.

1.1 Step 1 - Building a representation of the Internet at the AS level

In step 1, we construct an initial AS forest based on Border Gateway Protocol (BGP) data. We felt it was important to treat the highly inter-connected core of the Internet differently than the small ISPs on the edges. A few core ISPs have connections to hundreds of lower-tier ISPs and to many of the other core ISPs. It is appropriate to model them as a clique we will call the *forest floor*. The floor provides extremely stable routing with professionally managed fault tolerance and very high bandwidth. This forest floor is the centroid analogous to the strongly-connected portion of the Internet reported by Broido and Claffy [1]. From each of those nodes we grow a tree consisting of the regional, local, and leaf ISPs that are served by that core ISP. The presumed path of long distance packets would be from a leaf to the forest floor and then across the heavily connected floor and through the destination tree to the destination leaf.

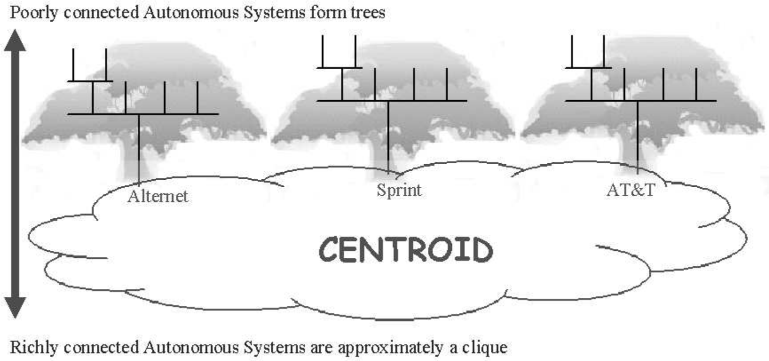
Poorly connected Autonomous Systems form trees
Alternet
Sprint
AT&T
CENTROID
Richly connected Autonomous Systems are approximately a clique

Fig. 1. Global Internet viewed as a clique and trees

1.2 Step 2 - Calibrating the relative depths

To assign accurately a depth to each branch of each tree, we needed a mechanism for comparing trees. Using 200,000 traceroutes collected from public traceroute servers, we adjust the depth of each AS so that packet flows seen in actual traceroutes would adhere to a pattern noticed by Gao [2] in which packets flow only uphill, then laterally, then only downhill. Whenever a packet takes an unexpected hop between trees, we use the opportunity to compare those depths.

1.3 Step 3 - Adding missing links and an alternate parent

Our traceroute data shows many links that were not present in the BGP table data. This is consistent with the observations of Jamin et al. [3] and others since BGP contains only sparse information about non-local links. We again applied Gao's observations (but in this case to the traceroute results) and differentiated customer-to-provider links from sibling-to-sibling links. This allowed us to add an extra link to each node in each tree for an alternate parent.

The breakthrough that allows us to dramatically improve the forest is, ironically, additions that make it no longer a forest of trees. We add up to one extra link from each customer to an alternate provider based on the preponderance of traceroutes in our training set. Now that tier-n nodes can have up to 2 parents, trees are now mini-graphs. There are links that connect mini-graphs to other mini-graphs and we depend on unidirectional notation to avoid cycles. The result is that our graph correctly classifies 91% of the traceroutes in the test set.

2 Related Work

The need to separate the Internet into two very different regions, the highly interconnected core versus the periphery of hierarchical trees, was clearly stated by Broido and Claffy in [1]. They call the core portion the *giant component* and include inside it a large number of IP nodes.

An AS-level view of the Internet has much less detail, but would retain its value as an operational tool. And an AS-level view can be obtained from publicly available BGP tables like those at Oregon Route-Views [4]. Early researchers assumed that two ASes were linked if their AS numbers were adjacent in an AS path. Jamin et al. [3] showed that there are many actual links that do not show up in BGP. Then Gao and Rexford [5] made a substantial improvement in AS-graph accuracy when they noticed that customer-to-provider links create a hierarchy. Gao [2] went on to identify peer-to-peer and sibling-to-sibling relationships between Autonomous Systems and proposed a mechanism for inferring the relationships from the AS paths in BGP Tables. Subramanian et al. [6] propose a 5-level classification of AS's that apply the inter-AS relationships to discover which AS's are dense core, transit core, outer core, regional, or customer. We find these classifications to be very helpful in understanding the flow of traffic through the Internet. We expand on their insights by constructing a topology that does not just depend on routing data. In effect, our traceroutes simulate using far more vantage points.

Initial work on clustering clients and proxy placement was done by Cuñha [7]. That work described a process of using *traceroute* to generate a tree graph of client accesses (using IP addresses collected from a Web server's logs). Proxies were then placed in the tree using three different algorithms and the effects on reduction of server load and network traffic were evaluated. More recent work by Krishnamurthy and Wang in [8] provide new mechanisms for clustering of clients at the router level.

A number of recent papers have addressed the issue of proxy placement based on assumptions about the underlying topological structure of the Internet [9–11]. Li et al. [9] describe an optimal dynamic programming algorithm for placing multiple proxies in a tree-based topology.

Jamin et al. [10] examine a number of proxy placement algorithms under the assumption that the underlying topological structure is not a tree. Their results show quickly diminishing benefits of placing additional mirrors (defined as proxies which service all client requests directed to them) even using sophisticated and computationally intensive techniques. Qiu et al. [11] also evaluate the effectiveness of a number of graph theoretic proxy placement techniques. They find that proxy placement that considers both distance and request load performs a factor of 2 to 5 better than a random proxy placement. They also find that a greedy algorithm for mirror placement (one which simply iteratively chooses the best node as the site for the next mirror) performs better than a tree based algorithm.

3 Topologically-guided Clustering

3.1 Finding the centroid

To determine the strongly connected component, we used a composite BGP table from Oregon Route-Views [4] and computed the portion of the AS list reachable by AS_n in h hops, varying h from 2 to 5. Almost all of the ASes (94%) can reach the bulk of the Internet (90% of the other ASes) in 5 hops. The 4 hop results were also not helpful. Many nodes high on the 4 hop list gained their height solely by virtue of having 2 well-connected neighbors. Finally, we found a useful gap in the 3 hop list in which 25 ASes can see 80% of the other ASes. Since we could visualize 25 trees on a single screenful of information, we declared those 25 ASes to be our (somewhat arbitrary) backbone. Broido and Claffy [1] did their study at IP-level rather than AS-level and found a *giant component* containing 8.3% of IP nodes. We chose our backbone to be small so that our service placement algorithm would contain more detail. Other applications of the forest representation may find the Broido and Claffy characterization more appropriate.

3.2 Definitions

The clustering algorithm uses neighbor sets and a distance function that acts as the length of a link.

The following definitions are used throughout the rest of this section:

- $Depth_n$ is the shortest distance from AS_n to a centroid node measured in AS hops.
- AS_n is a *neighbor* of AS_m if it immediately follows or precedes AS_m in any AS path.
- The *set of neighbors* of AS_n is denoted by N_n.
- $outdegree(n)$ of AS_n is $|N_n|$.
- AS_m is a candidate parent of AS_n if $m \in N_n$ and $Depth_m = Depth_n - 1$. The set of candidate parents of AS_n is denoted C_n.
- The *Hamming distance* between AS_n and AS_m is the number of neighbors exclusive to only one of them.

$$dist(n, m) = |N_n \cup N_m| - |N_n \cap N_m|$$

3.3 Clustering ASes using BGP routing data

For each node, we first compute $Depth_n$ for each AS_n using a variation of Prim's [12] Algorithm. Over 51% of the nodes are directly connected to a backbone node ($Depth_n \leq 1$) and over 91% of the nodes have $Depth_n \leq 2$.

Each AS with $Depth_n > 0$ chooses the nearest parent:

$$nearest(n) = \min_{m \in C_n} \{dist(n, m)\}$$

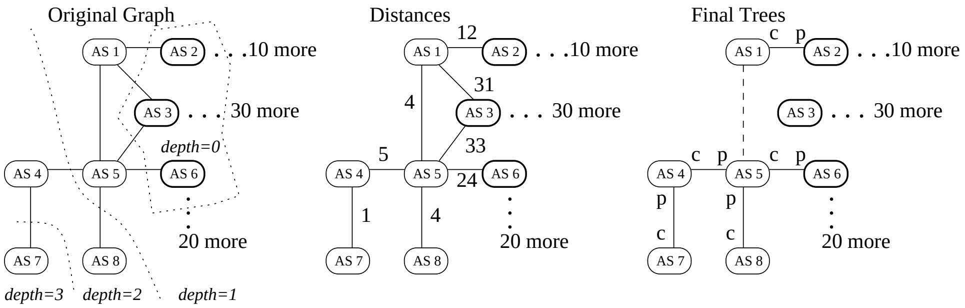

Fig. 2. Walk-though of the clustering algorithm

3.4 Initial annotations

The remaining neighbors in N_n consist of nodes that are either one hop farther from the backbone, the same $Depth_n$, or one hop closer. Here we explicitly assume that edges that take a packet closer to the backbone are uphill edges. We eliminate uphill edges that were not the nearest candidate parent. Links to nodes at the same $Depth_n$ are initially assumed to be siblings. Links that are one hop farther away from the backbone we (initially) assume to be customers. The initial annotations were very inaccurate. As we will see later, some inaccuracies stem from BGP's inability to see links and from mistaken inferences about which side of a link is a customer versus a provider. Gao [2] tried to further refine the downhill links by comparing the outdegree of the parent to that of the child. If the outdegree of the parent was a ratio, R, bigger than the child, she labeled the link a customer-to-provider link. We used traceroute data to discover peering, instead.

A demonstration of the algorithm will clarify it. Figure 2 shows Autonomous Systems numbered 1 through 8. In this example, AS2, AS3, and AS6 have already been designated as belonging to the backbone and declared to have $depth = 0$. When links from those 3 labeled nodes are followed, the newly discovered nodes are labeled with $depth + 1$ and moved from the unlabeled set to the labeled set. When all nodes have been labeled, AS7 has been found to be 3 hops from the backbone.

Now we remove redundant uphill links. The candidate parents for AS1 are AS2 or AS3. The Hamming distance between AS1 and AS2 is 12, since AS2 has 10 neighbors not also linked to AS1 and AS1 has 2 neighbors not also linked to AS2. This is the closest candidate parent, so the AS1-to-AS2 link is labeled $c \rightarrow p$ and the link from AS1 to AS3 is removed. This reflects the assumption that AS1-to-AS2 is the true link and AS1-to-AS3 is merely a backup link. Similarly, AS5 is assigned to be a customer of AS6. The link from AS1 to AS5 is now a sibling or peer link since they are at the same depth. Since this link does not take packets any closer to the backbone, it is not included in the tree. If subsequent evidence (see section 4.4) tells us it is an important link,

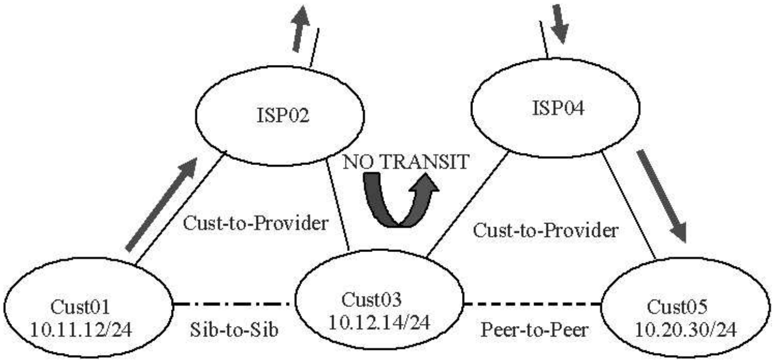

Fig. 3. Customers do not provide transit between providers.

it will be added back.

4 Annotating the AS Forest

To test our AS forest we ran traceroutes. From the list of 882 traceroute servers, route servers, and looking glass sites from www.traceroute.org, we chose 135 servers, each in a different AS, 2 or more hops from the centroid.

4.1 Choosing traceroute destinations

For the traceroute destinations, we chose a representative IP address for every AS. Even when the last hop fails to reach a working IP address, if a prior hop already shows the desired AS, it is a usable AS trace. In our case, 11% of our traceroutes did not end in the intended AS.

Over the week of March 11, 2002, we performed 200K traceroutes. Although this number is comparable to other studies [13,14] and much smaller than one study [1], our study did not need repetitions of the same routes.

4.2 Noting the relationship between AS's

The pattern we were hoping to see was the one identified by Gao [2]: each packet should flow uphill customer to provider, $c \rightarrow p$, (or laterally, sibling to sibling, $s \leftrightarrow s$) until it reaches the highest point needed to reach an AS (or a sibling or peer of an AS) upstream of the destination. Then the packet should flow only downhill provider to customer, $p \rightarrow c$, until it reaches the destination.

Figure 3 shows the phenomenon Gao described. Customer Cust03 does not want to pay money to ISP02 to receive packets destined for ISP04. In this annotated graph, the link between Cust03 and Cust05 is completely unknown by all other nodes. Assume ISP02 would like to send a packet to ISP04. If Cust03 advertised a willingness to take that packet, Cust03 would be paying both ISP02 and ISP04 to transit a packet that did not start or end at Cust03. We accept Gao's distinction [2] between a peer and a sibling because this makes the terms accurate, precise, and useful.

- ASes u and v have a peering relationship iff neither u transits traffic for v nor v transits traffic for u.
- ASes u is a provider of AS v iff u transits traffic for v and v does not transit traffic for u.
- ASes u and v have a sibling relationship iff both u transits traffic for v and v transits traffic for u.

The purpose of the peering relationship between Cust03 and Cust05 is to save the cost (and latency) of passing packets through ISP04. So Cust03 does not advertise Cust05's IP addresses (10.20.30/24) to anyone else. It is important that ISP02 be unaware of the link. If ISP02 gave packets addressed to 10.20.30/24 to Cust03, Cust03 would be paying the delivery cost and Cust05 would get a free ride.

The link between Cust01 and Cust03 illustrates a sibling relationship. In this case, Cust03 advertises that he will gladly accept packets for 10.11.12/24. If ISP04 wanted to send a packet to one of Cust01's IP addresses the packet would correctly find the path through Cust03. Our mechanisms for tracing the route of packets cannot distinguish between sibling relationships and peering relationships, so we treat all AS connections at the same depth as sibling relationships.

4.3 Translating IP Traceroute to AS Traceroute

For each line of each traceroute, we translated the router link IP address to an AS number using the centralized BGP table. Our results sometimes skip over an AS because packets were lost, we got no response from the router, or because the router's interface had an IP address that belongs to the AS at the other end of the link. Because of route aggregation and other practical limitations of BGP, our translation from IP address to AS could be wrong as well. Finally, ISP's need not use globally-routable IP addresses for links inside their own domain. If we miss seeing the ingress into the AS, we might completely miss seeing the AS. Thus, our translated AS path might understate the length of the true AS path.

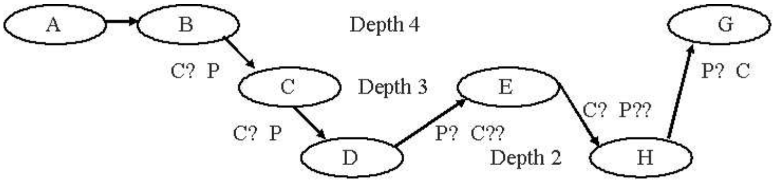

Fig. 4. Seemingly folded path implies one or more depths are incorrect

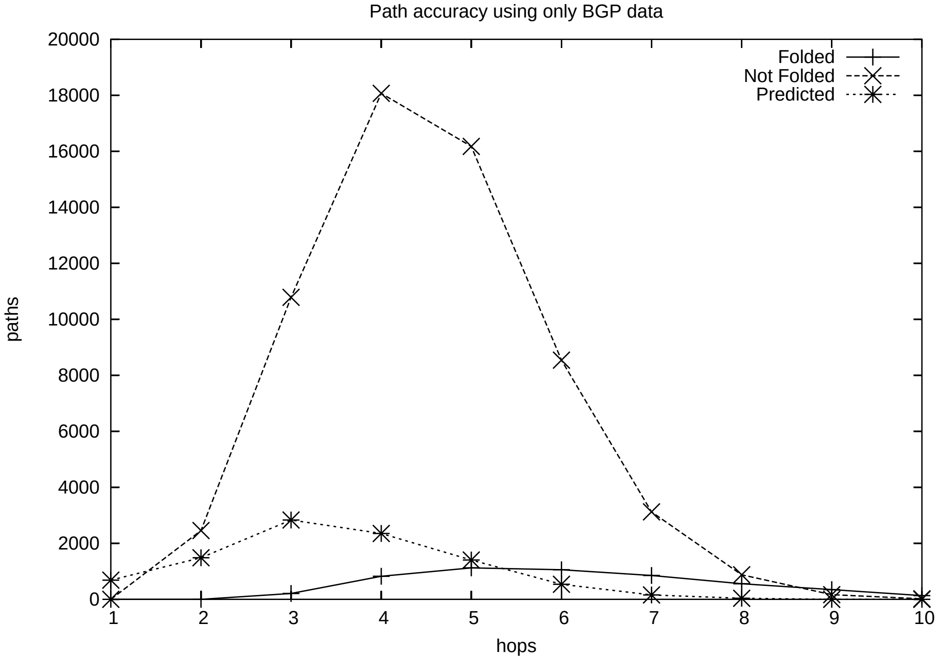

Fig. 5. Our early forest predicted only a tiny portion of the non-folded routes seen by traceroute.

A *folded trace* is a traceroute result that appeared to flow uphill after having taken a downhill hop. At that point, our AS forest had only provisional labels to categorize each link as a customer-provider link or a sibling link. So each traceroute gives us an opportunity to improve the annotations on our AS forest. Figure 4 shows evidence of a fold when node H is uphill from node E after that path had already started downhill.

As other researchers previously noted [3,2], a significant number of AS connections are hidden from most BGP tables. Figure 5 shows the results of the 74,963 unique complete traceroutes when applied to the AS forest derived solely from BGP information. The majority of the paths were from 3 to 6 AS hops long. A small number of paths were as long as 12 AS hops and a small number of IP addresses found routing loops at the inter-AS level.

The *Folded* traces are the AS paths that appeared to flow uphill after having taken a downhill hop. These traces indicate that our annotations of one or more hops are incorrect. The *Not Folded* traces are the paths that did not violate the uphill-to-downhill laws but contained links not in our AS forest.

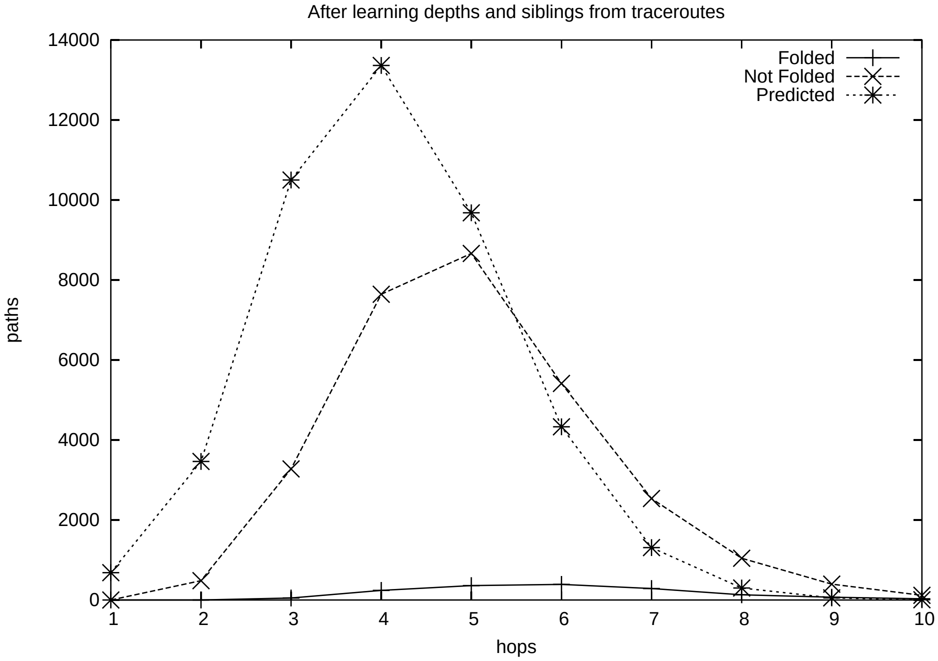

Fig. 6. Adjusting the annotations in the graph reduced the number of folded (implausible) paths and improved prediction.

This is also an opportunity for us to adjust the depth annotations. For a hop from AS_m to AS_n we compare $Depth_m$ to $Depth_n$ in cases where the AS forest did not have a link at (m, n). Finally, the *Predicted* traces are paths that only contained AS hops in the AS forest.

Figure 6 shows the same paths after the $Depth_n$ values have been refined. In this case, we pause for learning each time a traceroute shows an uphill hop after the packet had already reached a pinnacle. We used a *Current Best Hypothesis* algorithm [15] to test each hop of the traceroute. Imagine a trace $(k, l, ..., m, n)$ in which l was thought to be downhill from k, but n was thought to be uphill from m. This folded trace violates one or more of the annotations we have made. At least one of the links between k and m was annotated $A(k, l)$ as a $p \rightarrow c$ link. Choose k and l to be the closest instance of a $p \rightarrow c$ link. On the evidence of this traceroute, that could be a false positive. Alternatively, $A(m, n)$ was $c \rightarrow p$, preventing us from using it on the downhill side (a false negative). A special case where $l = m$ is easily handled.

To choose the appropriate generalization or specialization, we select the link most refuted by the evidence. That is, we track the failure count $F(m, n)$ and success count $S(m, n)$ of each annotation. If the total evidence $E = F(k, l) + S(k, l) + F(m, n) + S(m, n)$ exceeds a a learning rate threshold, α, we assume that we have seen enough cases to render a judgment. Each link, (k, l), has an error proportion $Err(k, l) = F(k, l)/(F(k, l) + S(k, l))$. If $Err(k, l) > Err(m, n)$ we change (m, n) to $s \leftrightarrow s$ by setting $Depth_m = Depth_n$. Alternatively, if the downhill link was more probably incorrect, we

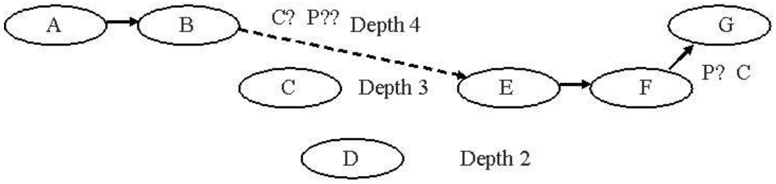

Fig. 7. Some traceroute results showed links not in BGP

set $Depth_n = Depth_m$. Since we have changed the depth of an AS, we correct all of the annotations of the links to that AS.

Some traceroute results showed links between Autonomous Systems that were not found in the BGP tables. Local BGP speakers (other than the ones used in RouteViews) choose not to tell other BGP speakers about private peerings and exchange points that do not have financial arrangements to provide transit. When discovered, these links give us an opportunity to compare the depths between trees. The algorithm found exchange points like the Russian Universities Federal Network (AS3267) quickly. $Depth_{3267}$ went from 9 hops from the backbone to 1. Others like the Milan Interconnection Point (AS16004) rose 4 times. Whenever a $Depth_n$ changes, other links become $c \rightarrow p$ or $p \rightarrow c$.

Figure 6 shows the results of learning depths. Each datapoint shows the average of 10 runs over the same traceroutes using 10-fold cross-validation with $\alpha = 6$. Higher values of α would require a larger data set.

Since this fixed many of our mistakenly labeled customer-provider paths, previously folded paths were now non-folded. We were surprised to see that the number of correctly predicted paths also improved. We can not yet explain some of the improvement. Our algorithm had reversed some customer-provider pairs. Also, there were improvements when unidirectional customer-provider links were upgraded to bidirectional sibling links. In future work, we will compare Gao's static (BGP-based) annotations with our dynamic annotations based on traceroutes.

4.4 Adding learned relatives

In many cases, the traced routes showed links that were not present in our BGP-based AS forest or even the BGP-based AS graph. We decided to add the most recent *alternate parent* to each AS whenever a trace showed an unexpected uphill hop from that AS. In the example in Figure 7, a traceroute showed AS E immediately after AS B going uphill. This was so common in the case of exchange points that we felt compelled to incorporate it into our representation. We limited the learning to identifying a single alternate parent for each AS. If we saved all of the alternate parents, the program would eventually have learned all of the routes seen, but the number of "correct"

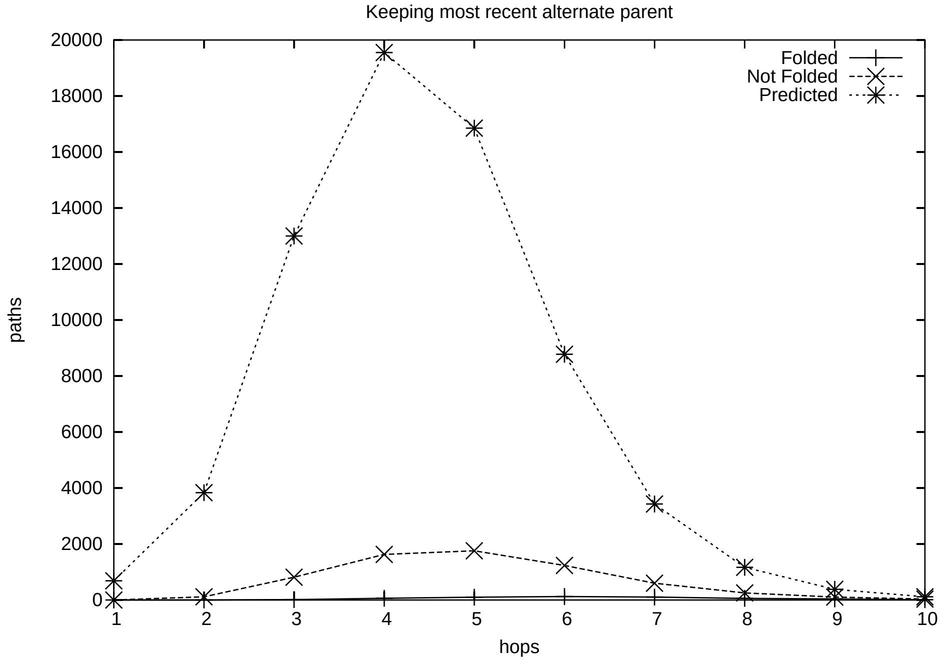

Fig. 8. Results with final AS forest

paths from one AS to another would grow too fast. This would have made our subsequent service placement algorithm ineffective. We placed no limit on the number of learned siblings at the same $Depth_n$.

Figure 8 shows the results of allowing each node in the AS forest a list of siblings and a single, alternate uphill link. We considered more sophisticated techniques for discovering the best of the discovered links, but were satisfied that the simplest technique (saving the most recent) was effective and reacted well dynamically. Again, the results are the average of 10-fold cross validation with training sets of 67,467 traces and test sets of 7,496 traces. Links with 5 or more AS hops had noticeably higher error rates than shorter paths. Over 91% of the test set traces correctly followed the uphill-then-downhill pattern and were composed only of links contained in our AS graph.

5 Conclusions

In this paper we have described methods for creating an AS forest based on BGP routing data and then refined by paths learned from traceroutes. The mechanism starts with an algorithm for finding the centroid of highly-connected nodes at the backbone of the Internet. It then grows a tree from each of those nodes based on BGP information. Rather than depend on subtle clues inside the BGP table to learn the relationships along the links, our mechanism learns those links from traceroute results.

The entire mechanism is suitable for use operationally, since it does not depend on proprietary information or large collections of route data. The resulting AS forest is computationally tractable for use in service placement.

We found that the mechanism learned a more accurate AS forest than what could be gleaned from the BGP tables alone, improving the prediction accuracy to 91% when tested against traced routes.

6 Acknowledgments

The authors would like to thank Winfred Byrd for his work on ASRoute and for discussions on the limitations of using traceroute results. We would also like to thank Thomas Hangelbroek for the initial implementation of the approximate centroid algorithm in matlab and Dr. Amos Ron for the insight of using adjacency matrices to compute it. The authors are also grateful to the anonymous reviewers for helping us improve the paper.

References

[1] A. Broido, kc claffy, Internet topology: connectivity of IP graphs, Tech. rep., CAIDA, http://www.caida.org/outreach/papers/topologylocal (2001).
URL `http://www.caida.org/outreach/papers/topologylocal`

[2] L. Gao, On inferring autonomous system relationships in the internet, in: IEEE Global Internet Symposium, 2000.
URL `http://www-unix.ecs.umass.edu/~gao/ton.ps`

[3] H. Chang, R. Govindan, S. Jamin, S. Shenker, W. Willinger, Towards capturing representative AS-level Internet topologies, in: ACM SIGMETRICS, 2002.
URL `http://topology.eecs.umich.edu/archive/tr02.ps`

[4] R. Views, University of oregon, http://www.antc.uoregon.edu/routeviews.

[5] L. Gao, J. Rexford, A stable internet routing without global coordination, in: Proceedings of ACM SIGMETRICS '00, 2000.

[6] L. Subramanian, S. Agarwal, J. Rexford, R. Katz, Characterizing the internet hierarchy from multiple vantage points, in: Proceedings of IEEE INFOCOM '02, 2002.
URL `http://www.ieee-infocom.org/2002/papers/594.pdf`

[7] C. Cuñha, Trace analysis and its applications to performance enhancements of distributed information systems, Ph.D. thesis, Boston University (1997).

[8] B. Krishnamurthy, J. Wang, On network aware clustering of Web clients, in: Proceedings of ACM SIGCOMM '00, Stockholm, Sweden, 2000.

[9] B. Li, M. Golin, G. Italiano, X. Deng, K. Sohraby, On the optimal placement of Web proxies in the Internet, in: Proceedings of IEEE INFOCOM '99, New York, New York, 1999.

[10] S. Jamin, C. Jin, A. Kurc, D. Raz, Y. Shavitt, Constrained mirror placement on the Internet, in: Proceedings of IEEE INFOCOM '01, Anchorage, Alaska, 2001.

[11] L. Qiu, V. Padmanabhan, G. Voelker, On the placement of Web server replicas, in: Proceedings of IEEE INFOCOM '01, Anchorage, Alaska, 2001.

[12] R. Prim, Shortest connection networks and some generalizations, Bell System Technical Journal 36 (1957) 1389–1401.

[13] V. Paxson, End-to-end Internet packet dynamics, in: Proceedings of ACM SIGCOMM '97, Cannes, France, 1997.

[14] R. Govindan, H. Tangmunarunkit, Heuristics for Internet map discovery, in: Proceedings of IEEE INFOCOM '00, IEEE, Tel Aviv, Isreal, 2000, pp. 1371–1380.
URL `http://www.icsi.berkeley.edu/~ramesh/papers.html`

[15] J. S. Mill, A system of logic, ratiocinative and inductive: Being a connected view of the principles of evidence, and methods of scientific investigation., J.W. Parker, London (1843).