Inferring Link Weights using End-to-End Measurements

Ratul Mahajan Neil Spring David Wetherall Tom Anderson

University of Washington

Abstract—**We describe a novel constraint-based approach to approximate ISP link weights using only end-to-end measurements. Common routing protocols such as OSPF and IS-IS choose least-cost paths using link weights, so inferred weights provide a simple, concise, and useful model of intra-domain routing. Our approach extends router-level ISP maps, which include only connectivity, with link weights that are consistent with routing. Our inferred weights agree well with observed routing: while our inferred weights fully characterize the set of shortest paths between 84-99% of the router-pairs, alternative models based on hop count and latency do so for only 47-81% of the pairs.**

I. Introduction

Over the past several years, Internet mapping technology based on end-to-end measurements has matured to the point where realistic router-level maps are now becoming available [3, 6, 11, 19]. These maps are in turn being used to study the structure of the Internet (e.g., node outdegree distribution [7, 21]). However, we note that the current maps do not support studies based on Internet paths. This is because topology alone does not determine which paths will be selected. Rather, the routing model, along with the topology, determines the paths that are taken in practice.

In this paper, we study the problem of determining a routing model that characterizes the intra-domain forwarding paths used within an ISP. Our model is based on link weights, since commonly used intra-domain routing protocols such as OSPF and IS-IS use link weights to compute least cost or shortest paths. Somewhat surprisingly, we find that we are able to use a constraint-based method to approximate ISP link weights based solely on end-to-end measurements. The key observation behind the solution is that the weight of the path taken by a packet is at most the weight of any other path between the same nodes; otherwise another path would have been taken. We represent this rule for all observed paths as a set of constraints on link weight values, and solve the resulting constraint system to infer the weights. Simple extensions to the constraint system deal with noise in the data, such as longer paths being taken due to transient failures.

Email: {ratul,nspring,djw,tom}@cs.washington.edu

We applied our approach to six diverse ISP maps collected by Rocketfuel [19], including Ebone (Europe), Sprint (USA) and Telstra (Australia), using the traceroute data collected as part of topology measurement itself. We find that the inferred weights are an excellent model for observed routing: across the six ISPs, the sets of paths between 84-99% of the router-pairs were fully characterized by our weights. We also find that simpler alternatives such as minimum hop count or latency are poor models of intra-domain routing: they described paths between only 47-81% of the router-pairs.

To our knowledge, our technique is the first presented method for approximating ISP link weights. It complements inter-domain routing policy inference based on observed BGP paths [10, 20]. Since link weights are the simplest mechanism currently available to ISPs for implementing traffic engineering [9], our technique provides a starting point for an external study of traffic engineering practices.

In our ongoing work, we are focusing on understanding the correspondence between the inferred and actual link weights. The inferred weights are valuable in that they characterize the observed routing. Yet they may not be an exact match for the actual ISP link weights because the set of weights that achieves a given set of routes is not unique. The weights may differ by a scaling factor, and, more importantly, each weight may vary within a small range. This does not affect the observed paths but may reduce the ability to correctly predict the paths selected during failures. Ideally, we hope to be able to predict the paths used during failures, and so are investigating the "realism" of our inferred weights.

The rest of the paper is organized as follows. Section II explains our approach in detail, and Section III evaluates the effectiveness of the inferred link weights in modeling intra-domain routing. We discuss related work in Section IV, and conclude in Section V.

II. Constraint-Based Approach

In this section, we explain our approach to infer link weights. As input we use a network map and a set of observed paths through it, both of which are obtained by various mapping efforts using traceroutes. We start with the abstract problem we are trying to solve.

A. Abstract Problem

Consider a network with vertices (nodes) and directed edges (links). Each edge has a positive weight, and the weight of a path is the sum of the weights of its edges. Assume that weighted shortest path routing is used in the network, so the least weight path, or paths in case of a tie, between two vertices are chosen. All common intra-domain routing protocols such as OSPF, IS-IS, and RIP compute weighted shortest paths.

Given the routing as a set of chosen paths, our goal is to assign weights to the edges such that the least weight paths between all vertex-pairs match the paths that are chosen. There is not a unique solution to this problem for two reasons. First, scaling all weights by the same factor does not affect routing. Second, changing the weight of a link within bounds may have no impact on routing. Our method below thus produces one of the possible solutions.

Section II-B describes the basic solution to the problem, and Section II-C modifies it to reduce the number of constraints. In both these sections, we assume that all chosen paths between all vertex-pairs are known. Later in Section II-D we describe extensions to deal with shortcomings in the routing collected using traceroute.

B. Basic Solution

The key observations in our solution are: i) the weight of the chosen path(s) is less that than of other possible paths between the same vertices; and ii) if multiple chosen paths exist between two vertices, they have equal weight. These observations can be translated into a constraint system that can be solved to obtain the weights.

For example, consider the network in Figure 1. Assume that the chosen paths between A and G are ADG and $ABEG$, which means that they have equal weight, and are shorter than the alternate paths ACG, $ACFG$, $ABDG$, $ADEG$ and $ABDEG$. We represent these facts using the constraints shown below.

1. $w_{ad} + w_{dg} = w_{ab} + w_{be} + w_{eg}$
2. $w_{ad} + w_{dg} < w_{ac} + w_{cg}$
3. $w_{ad} + w_{dg} < w_{ac} + w_{cf} + w_{fg}$
4. $w_{ad} + w_{dg} < w_{ab} + w_{bd} + w_{dg}$
5. $w_{ad} + w_{dg} < w_{ad} + w_{de} + w_{eg}$
6. $w_{ad} + w_{dg} < w_{ab} + w_{bd} + w_{de} + w_{eg}$

We consider only simple (non-circular) alternate paths as every circular path has a shorter simple path. This makes it sufficient to constrain the chosen path to be shorter than the corresponding simple path.

Similar constraints are set up for chosen paths between all vertex-pairs. Because the number of both vertex pairs and simple paths is finite, there are a finite number of constraints. We solve the constraint system using linear programming [15] to obtain a set of weights consistent with the given routing. As long as the given routing is shortest path with (unknown) positive link weights, we obtain a consistent (solvable) constraint system.

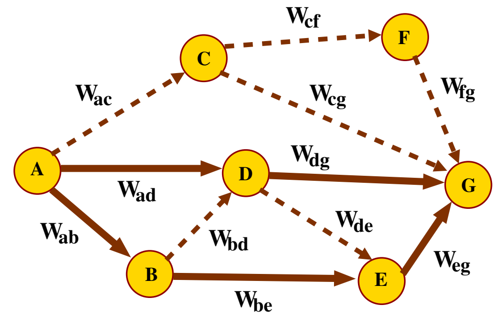

Fig. 1. The chosen paths, ADG and $ABEG$, are shown with solid lines, and the alternate paths with dashed lines.

C. Reducing the Number of Constraints

The basic solution leads to an exponential number of constraints because a constraint is set up for every simple alternate path. This makes it intractable for large networks. To address this problem, we use a simple criterion to identify and remove redundant constraints.

Let SP_{sd} be a chosen path between v_s and v_d, and $AP_{si}{\cdot}AP_{id}$ be an alternate path with an intermediate vertex v_i. Let $W(P)$ denote the weight of path P. The constraint $W(SP_{sd}) < W(AP_{si}{\cdot}AP_{id})$ is redundant if at least one of the two conditions is true for any intermediate vertex v_i in the alternate path: *i)* AP_{si} is not a chosen path between v_s and v_i; or *ii)* AP_{id} is not a chosen path between v_i and v_d.

It is easy to see why this criterion, which is similar to path relaxation in Dijkstra's algorithm, identifies redundancy. Let SP_{si} and SP_{id} be chosen paths between their respective endpoints. The concatenation of these two leads to a valid alternate path $SP_{si}{\cdot}SP_{id}$. So there exists the constraint $W(SP_{sd}) < W(SP_{si}{\cdot}SP_{id})$. If one of the above conditions is true, $W(SP_{si}{\cdot}SP_{id}) < W(AP_{si}{\cdot}AP_{id})$ so the constraint $W(SP_{sd}) < W(AP_{si}{\cdot}AP_{id})$ is redundant.

Based on this criterion, Constraints 3 through 6 in the example above are redundant if the chosen paths between directly connected vertices is the edge connecting them, and the shortest path between B and G is BEG.

A simple analysis shows that we are now left with a number of constraints that is polynomial in n, the number of nodes. The maximum number of alternate paths for a vertex pair (v_s,v_d) is n, because each vertex v_i can appear in at most one alternate path – the concatenation of chosen paths SP_{si} and SP_{id}. Even if there are multiple chosen paths between v_s (or v_d) and v_i, only one constraint needs to be set up as all of them have equal weight. With n^2 vertex pairs, each with at most n alternate paths, the number of constraints is $O(n^3)$. The actual number of constraints is much lower than n^3 since most non-redundant alternate paths would contain multiple intermediate vertices.

Our implementation does not enumerate all alternate paths before pruning out the redundant paths. Instead, we use a dynamic path growing algorithm that produces only non-redundant alternate paths.

D. Dealing with Shortcomings in Measurement Data

When the chosen path information is collected using traceroute measurements, it is likely to have two problems: $i)$ some paths may be longer than the stable shortest paths due to transient events such as failures; and $ii)$ not all chosen paths may be observed. We now extend our solution to address both these problems.

If every observed path is considered a chosen (shortest) path, it may lead to an inconsistent constraint system. Our extension to deal with this *noise* is based on constraint hierarchies, which provide an innovative way to set up constraints such that all of them may not be satisfied [2]. The main idea is to associate error variables with constraints, and minimize the weighted sum of errors. In our approach, we use an error variable per observed path.

In Figure 1, assume that two paths, ADG and $ABEG$, from A to G were observed. It is not necessary that both (or any) of these paths are shortest. We associate with them the error variables e_{adg} and e_{abeg}, which represent the excess weight (beyond that of the true shortest path) of the path. The modified constraints are:

1. $w_{ad} + w_{dg} - e_{adg} = w_{ab} + w_{be} + w_{eg} - e_{abeg}$
2. $w_{ad} + w_{dg} - e_{adg} < w_{ac} + w_{cg}$

Similar constraints are set up for all vertex-pairs. We use the simplex algorithm [15] to solve the constraint system and minimize the weighted sum of the error variables, using the number of times the path was observed as the weight. The error variables of non-shortest paths have a non-zero value in the solution.

Traceroute data may not contain chosen paths between all vertex-pairs. We define *pair-wise completeness* as the fraction of vertex-pairs between which at least one path was observed. Poor pair-wise completeness blunts the constraint reduction criterion in Section II-C: when chosen path information between v_s (or v_d) and v_i is not available, some alternate paths between v_s and v_d with v_i as intermediate vertex cannot be eliminated (some can still be eliminated based on other intermediate vertices in the path). In principle, this means that our solution is no longer polynomial in the number of nodes, but if pair-wise completeness is high, the constraint system should still be tractable. The measurement methodology of our data source (Section III-A) helps in achieving high pair-wise completeness.

Pair-wise completeness can be improved using two techniques. First, due to *reverse path symmetry*, a consequence of both directions of a link having equal weight, the reverse path of a shortest path is shortest. We confirmed reverse symmetry for all the ISPs: the most common path from A to B is the reverse of the most common path from B to A.[1] Second, the *optimal substructure* property – any subset of a shortest path is shortest – gives us shortest paths between any pair of vertices in a given shortest path.[2]

Another measurement artifact is that while we may have a path between two vertices, we are not guaranteed to have seen all the chosen paths between them. This means that the weight of the chosen path is not more than, as against strictly less than, the alternate paths in the network. We reflect this by changing the strict inequalities ('$<$') in our constraints to '$\leq$'.

E. Applicability

Our approach is applicable for any network with weighted shortest path routing with a single setting of link weights: the preferred path between two nodes is dependent only on the weights, and not on, for instance, the destination address of the packet (beyond the destination address determining the destination node).

Most ISP networks fit the above model, but there are some exceptions. BGP confederations can lead to non-shortest-path routing. With OSPF, some paths may not be least weight when using area aggregates [17], or when a lower weight external path exists between nodes in the same area (since paths between nodes in the same area are restricted to that area). The latter would happen only when intra-area weights are high, making it unlikely in practice since intra-area weights are usually low. If a network uses circuit technologies such as MPLS, the routing policy is not visible at the IP layer. Our approach would yield little insight into the routing policy of such networks.

Even in a network that does not use weighted shortest path routing exclusively, there may be a set of link weights that achieve the same routing. Our analysis will infer such weights, even though they are not used.

III. Evaluation

In this section we evaluate the link weights inferred using our approach. Our input data is described in Section III-A. We describe several alternate link cost models in Section III-B for comparison with our inferred weights in Section III-C, where we show that these these alternate models are inadequate to describe ISP routing.

[1] In contrast, inter-domain routes are frequently asymmetric.

[2] Our current implementation uses these derived paths for both setting up shortest path constraints and eliminating redundant constraints (Section II-C), but we have since realized that using them only for reduction would increase robustness to noise.

AS	Name	Rtrs	Links	Paths	Pairs
1221	Telstra (au)	115	153	20K	88%
1239	Sprint (us)	323	972	214K	54%
1755	Ebone (eu)	88	161	15K	57%
3257	Tiscali (eu)	164	328	9K	66%
3967	Exodus (us)	80	147	27K	68%
6461	Abovenet (us)	145	376	88K	63%

Fig. 2. ISPs with the number of backbone routers and links, the number of traceroutes, and pair-wise completeness.

We use three criteria to evaluate how well a particular cost model characterizes observed routing.

1. *The fraction of observed paths that had least cost.* With an accurate routing model, most observed paths should be least cost. This is a metric for overall data fit; each path is weighted by the number of times it was seen.

2. *The fraction of dominant paths that had least cost.* A dominant path is the most commonly seen path between two nodes, and thus is most likely to be the stable path between them. Uncommon paths, such as those taken during failures, are weeded out. With a good model, most dominant paths should be least cost. All dominant paths are weighted equally, so unlike the previous evaluation, the results are not biased by a few common paths.

3. *The fraction of node-pairs between which the routing was fully characterized.* If a model predicts multiple least cost paths between two nodes, they should all be observed due to load balancing across equal cost paths. Thus, complete characterization of routing between node-pairs means that not only must most paths taken between them be least cost, but also that paths not seen must be costlier. A coarse model that assigns the same cost to many paths would fare well in the first two evaluations by overstating the number of least cost paths, but not in this one unless all those paths were actually used.

In Section III-D we study the predictive power of the routing model – how much data is needed to derive weights that predict ISP paths that were not directly observed.

Paths that are longer than the least-cost paths predicted by the inferred weights can be present in the measurement data due to: i) transient events such as failures; ii) the network's routing model not being weighted shortest path; or iii) link weight or topology changes while measurements are being taken. In the future, we intend to distinguish between the three causes behind long paths based on a lifetime based analysis of the observed paths.

A. Data Source

The six ISPs we study are listed in Figure 2. The maps were collected by Rocketfuel [19]. The pair-wise completeness in the table has been computed after application of the two techniques described in Section II-D. We infer weights of the links that connect backbone routers. Rocketfuel extracts the backbone using the information present in the DNS names and connectivity structure of the routers. Traceroute data obtained as part of topology measurement itself were used as input. Rocketfuel is very good at collapsing the interfaces on the same router (alias resolution), leading to more accurate maps. It also uses many (over 600 in our data set) public traceroute servers as vantage points, thus providing a fair view of an ISP's routing.

B. Alternate Metrics

We compare the inferred weights with three other metrics – *latency*, *hops*, and *hoplat*. *Latency* uses the link propagation delay rounded up to the nearest millisecond as cost. We used DNS-based heuristics to map routers to cities [16]. Link delay is approximated as the time to traverse the distance between those cities at the speed of light in fiber.[3] The actual fiber path may not be direct and circuit switching may create a link between cities that does not physically exist. However, we believe geographic distance is a reasonable approximation, since fiber paths are likely to be close to the direct path and circuit switching is not used extensively in the ISPs we studied [19].

Hops assigns the same cost to all the links, making minimum hop count the routing criteria. ISP backbones are highly meshed [19], so there are many paths with equal hop count but vastly different latencies. Network administrators may prefer to distinguish between such paths. *Hoplat* models this preference by choosing the least latency path(s) among minimum hop count paths.

While other measures such as capacity may describe routing more accurately, such information is not currently available in ISP maps derived using traceroute. Of course, if routing is dependent on an unknown topology property (such as link capacities), our methodology may help to infer that characteristic.

C. Routing Characterization

We now evaluate how well each metric – latency, hops, hoplat, and inferred weights – models observed routing.

1. **All Observed Paths**

Figure 3 shows the fraction of observed paths that were least cost using each metric. Observed paths agree best with inferred weights: while weights fit 87-100% of the paths across the six ISPs, the next best metric (hops) fits only 67-92%.

2. **Dominant Paths**

Figure 4 shows the fraction of dominant paths that were least cost. As before, inferred weights are significantly more successful compared to the other metrics. While the inferred weights fit 76-98% of the dominant paths, the best

[3] In the future, we intend to extract link latencies from traceroutes.

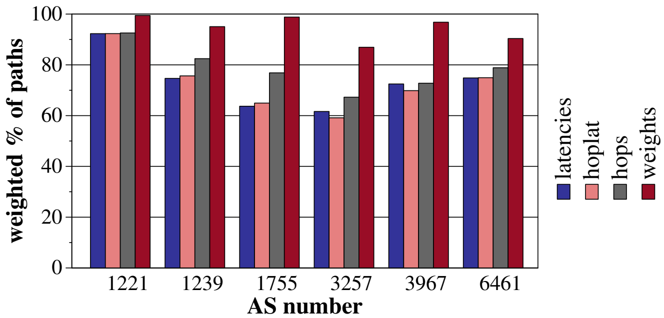

Fig. 3. Percentage of observed paths that were least cost.

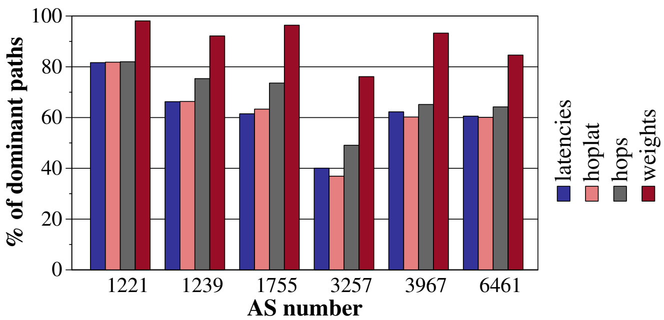

Fig. 4. Percentage of dominant paths that were least cost.

alternate metric (hops) fits only 49-82%. Nine out of ten dominant paths that were not fitted by weights were seen five or fewer times, increasing the probability of the dominant path not being the stable path.[4]

3. **Routing Between Pairs of Nodes**

To capture how well routing between node-pairs is characterized, we partition them into the following classes:

- *full:* each least cost path between the pair was seen;
- *partial:* some, but not all, least cost paths were seen;
- *none:* no least cost path was seen.

Figure 5 shows the fraction of node pairs in the full and partial classes, with the remaining being in none. We can see that hops is a very coarse metric; it partially characterizes as many as 4-20% of the pairs, which means that it predicts more least cost paths than are actually present in the network, thus overestimating the extent of multi-path routing. The success of hops in earlier evaluations compared to latency and hoplat can be attributed to this property; all the alternate metrics fully describe routing only for 47-81% of the node-pairs. On the other hand, inferred weights fully describe 84-99% of the node-pairs, and partially characterize only 1-3%. Note that some degree of partial fitting would also arise from not having observed enough paths to see all the least cost paths.

[4] To reduce measurements required to collect an ISP map, Rocketfuel minimizes the number of paths measured between the same pair of nodes. Multiple measurements of infrequent paths over a period of time would help eliminate this problem.

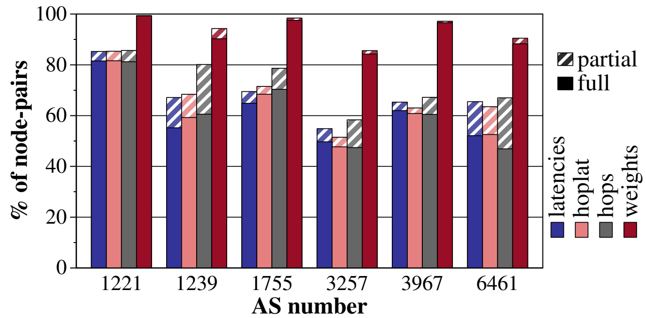

Fig. 5. Percentage of node-pairs in the full or partial classes.

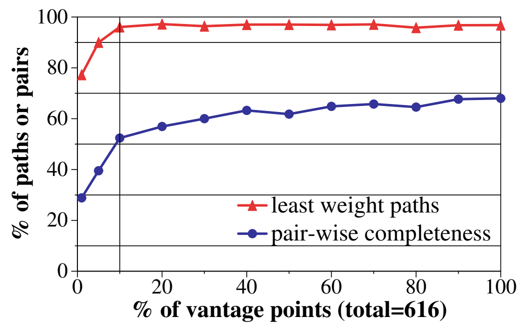

Fig. 6. Predictive power of routing model for Exodus.

D. *Predictive Power of the Routing Model*

We next investigate how many measurements are needed to derive a good model for overall ISP routing, predicting paths that were not directly observed. We refer to this as the predictive power of the model. Since we do not measure all ISP routes, the closest experiment we can perform is to infer weights from a fraction of our measurements and assess how well those weights predict the entire set of route observations. To take a fraction of the measurements, we randomly varied the number of vantage points used to collect the traceroute data, in subsets of size 1%, 5%, and 10-100% in steps of 10%.

Figure 6 shows for Exodus (3967) the percentage of paths in the total dataset that were least cost using the weights inferred from the subset of measurements. We see that the weights derived from 10% of the vantage points describe routing almost as well as weights derived using all of them. For comparison, the figure also shows pair-wise completeness as a function of the percentage of vantage points. The fit does not improve as we move from 50% to almost 70% pair-wise completeness. This is encouraging as it suggests that 50% completeness is sufficient to predict the paths between node-pairs that were not directly observed.

We ran this experiment for other ISP maps[5] and obtained similar results in terms of the pair-wise completeness needed for a fit with good predictive power: 80%

[5] We could not do this experiment for Sprint (1239): as we reduced pair-wise completeness, few constraints could be eliminated (see Sections II-C and II-D). The resulting constraint system was intractable.

for Telstra (1221), 40% for Ebone (1755), 50% for Tiscali (3257), and 45% for Abovenet (6461). The high completeness percentage required for Telstra is an artifact of its simple topology, in which the pair-wise completeness was 75% even with just 1% of the vantage points.

IV. Related Work

Our work extends the Internet topology maps [3, 6, 11, 19] with link weights that are consistent with observed routing. Like researchers studying other network properties [5, 12, 13, 14], we use only end-to-end measurements to infer link weights. As we move towards understanding intra-domain routing, we complement various inter-domain routing analyses [10, 18, 20, 22].

Our approach is close to that of Fortz & Thorup [9] and Wang et al. [23]. These works formulate optimal routing as a constraint satisfaction problem. Their algorithms use traffic demands as input to "engineer" link weights that satisfy some policy goal such as to minimize utilization, while our approach uses measurements of observed paths to "reverse-engineer" weights.

The abstract problem in Section II-A is an inverse shortest paths problem (ISPP). In an ISPP, the goal is to assign edge weights that are consistent with a given routing characteristic such as end-to-end distances or paths. Various ISPPs have been studied [4, 8]. We solve a specific instance, which to our knowledge has not been studied before, and extend the solution to handle inconsistent input.

V. Conclusions

We described a novel, constraint-based approach to infer link weights using only end-to-end measurements. Our approach is efficient and deals effectively with noisy data. We used it to infer the backbone link weights for six ISP networks, and found that the inferred weights characterize the ISP routing much better than alternate metrics based on hop count and latency: inferred weights fully characterized routing between 84-99% of the router-pairs across the six ISPs, while alternate metrics could do so for only 47-81% of them.

Our results enhance ISP topologies with link weights, making them useful for simulations and routing studies. Obtaining link weights is also a step towards characterizing intra-domain routing and traffic engineering; we intend to use them to study properties such as the extent of multipath routing, hop and latency inflation in least weight paths, routing load distribution, and failure resilience.

Acknowledgements

We thank Alan Borning for help with constraint solving, Ashish Sabharwal for help with reducing the number of constraints, and Christophe Diot and Jennifer Rexford for useful discussions.

We also thank the developers of *lp_solve* [1] and SoPlex [24], the linear program solvers used in this work.

This work was supported by DARPA under grant no. F30602-00-2-0565.

References

[1] M. Berkelaar. lp_solve: linear programming code. ftp://ftp.ics.ele.tue.nl/pub/lp_solve/.

[2] A. Borning, B. Freeman-Benson, and M. Wilson. Constraint hierarchies. *Lisp and Symbolic Computation*, 5(3), 1992.

[3] H. Burch and B. Cheswick. Mapping the Internet. *IEEE Computer*, 32(4):97–98, 102, 1999.

[4] D. Burton and L. Toint. On an instance of the inverse shortest paths problem. *Mathematical Programming*, 53, 1992.

[5] R. L. Carter and M. E. Crovella. Measuring bottleneck link speed in packet-switched networks. Technical Report TR-96-006, Boston University Computer Science Department, 1996.

[6] k. claffy, T. E. Monk, and D. McRobb. Internet tomography. In *Nature*, January 1999.

[7] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. In *ACM SIGCOMM*, 1999.

[8] S. P. Fekete, W. Hochstättler, S. Kromberg, and C. Moll. The complexity of an inverse shortest path problem. *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, 49, 1999.

[9] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *IEEE INFOCOM*, 2000.

[10] L. Gao. On inferring autonomous system relationships in the Internet. In *IEEE Global Internet Symposium*, 2000.

[11] R. Govindan and H. Tangmunarunkit. Heuristics for Internet map discovery. In *IEEE INFOCOM*, 2000.

[12] V. Jacobson. Pathchar. ftp://ftp.ee.lbl.gov/pathchar.

[13] V. Jacobson. Traceroute. ftp://ftp.ee.lbl.gov/traceroute.tar.Z.

[14] K. Lai and M. Baker. Nettimer: A tool for measuring bottleneck link bandwidth. In *USITS*, 2001.

[15] K. G. Murty. *Linear Programing*. John Wiley & Sons, 1983.

[16] V. N. Padmanabhan and L. Subramanian. An investigation of geographic mapping techniques for Internet hosts. In *ACM SIGCOMM*, 2001.

[17] R. Rastogi, Y. Beribart, M. Garofalakis, and A. Kumar. Optimal configuration of OSPF aggregates. In *IEEE INFOCOM*, 2002.

[18] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *ACM SIGCOMM*, 1999.

[19] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *ACM SIGCOMM*, 2002.

[20] L. Subrmanian, S. Agarwal, J. Rexford, and R. H.Katz. Characterizing the Internet hierarchy from multiple vantage points. In *IEEE INFOCOM*, 2002.

[21] H. Tangmunarunkit, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Network topology generators: Degree-based vs. structural. In *ACM SIGCOMM*, 2002.

[22] H. Tangmunarunkit, R. Govindan, and S. Shenker. Internet path inflation due to policy routing. In *SPIE ITCom*, 2001.

[23] Y. Wang, Z. Wang, and L. Zhang. Internet traffic engineering without full mesh overlaying. In *IEEE INFOCOM*, 2001.

[24] R. Wunderling. SoPlex: The sequential object-oriented simplex class library. http://www.zib.de/Optimization/Software/Soplex/soplex.php.