A Constraint Satisfaction Approach to Testbed Embedding Services

Jeffrey Considine John W. Byers Ketan Mayer-Patel

***Abstract*— Today's networking community is becoming increasingly skeptical of the significance of research results founded wholly upon experimental results conducted in simulation. Now, with the availability of wide-area distributed testbeds such as PlanetLab, it is feasible to move beyond evaluation by simulation, and to perform wide-area experiments across the Internet as an alternative. However, while use of a distributed testbed affords much greater realism than a network simulation, there is a significant downside, as tight control over one's experiments is relinquished.**

We argue that providing services for distributed testbeds that capture aspects of the specifiable, repeatable behavior implicit in simulation and emulation will be an integral component of next-generation testbeds. As a case study, we consider the following problem: given a desired end-system topology consisting of a set of pairwise constraints (such as upper and lower bounds on bandwidth and delay), locate a representative subtopology within a wide-area testbed that satisfies those constraints. Previous work on Netbed addresses wide-area embedding problems of this form using an optimization framework, whereas we employ constraint satisfaction. We discuss the relative merits of these approaches, outline the theoretical foundations of a distributed service that provides a synergy between adaptive network measurements and the embedding process, and report on preliminary experimental results conducted on PlanetLab.

I. Introduction

Networking researchers primarily employ three experimental methodologies: simulation, emulation, and observation of live deployments across the wide-area Internet. While the current state of the art in simulation and emulation admits elaborate scenarios and has advanced rapidly over the past decade, the current best practice in conducting Internet-wide experiments is rather primitive in comparison. One of the strengths of both simulation and emulation is the degree of control which a researcher can exert over an experiment. As a result, one can prescribe a specific set of test conditions, can run an experiment repeatedly (and repeatably!), and can be confident that results are not distorted by external factors. But the main weakness of these approaches is closely related to their main strength: control is not easily relinquished to a realistic model of the Internet, since such a model does not exist [3], [2]. Thus, simulated scenarios are often open to criticism for being overly artificial or simplistic. It is this point which compels networking researchers to perform wide-area experiments whenever feasible, since wide-area experimentation enables them to test their methods "in the wild" rather than against a model.

Historically, obtaining access to a large set of distributed nodes has been a barrier to Internet experimentation on any significant scale. The community has responded to this challenge with a number of efforts currently in progress to build large-scale, widely deployed, Internet testbeds such as PlanetLab [8]; and in developing emulation environments that incorporate connections across the wide-area, such as Netbed [12]. Deploying a large-scale networking testbed, however, is only a starting point. In particular, the wide-area network conditions between end-systems reflect instantaneous Internet usage, and unlike simulation or emulation environments, these conditions cannot currently be configured, customized, nor tightly controlled. This lack of control can leave researchers longing for the comforts of simulation, such as specifiable, predictable and repeatable behavior.

The contrasting merits of simulation and emulation versus the use of testbeds motivate us to consider whether the best features of each can be combined. We view simulation and emulation environments as offering researchers a "blank slate" on which they can craft their experimental set-up down to fine details. In contrast, many relevant parameters of a testbed configuration are fixed in advance and are outside the experimenter's direct control. However, all is not lost, as the experimenter often has the ability to determine or estimate parameters of a wide-area testbed by conducting measurements.[1] Also, it seems likely that testbeds will soon scale to sizes that could be orders of magnitude larger than a typical (small) experiment. Therefore, in the near future, a researcher may well have the ability and flexibility to *select* a subset of nodes from the testbed that satisfy desired parameters of a given test configuration.

Currently, the de facto selection practice on a platform such as PlanetLab is to hand-pick nodes and links with the "right" characteristics, a time-consuming and tedious practice. An experimenter on this platform must hope that

J. Considine and J. Byers are with Boston University, Dept. of Computer Science. Supported in part by NSF grants ANI-0093296 and ANI-9986397. E-mail: {jconsidi,byers}@cs.bu.edu.

K. Mayer-Patel is with the University of North Carolina, Chapel Hill, Dept. of Computer Science. Supported in part by NSF grant ANI-0219780. E-mail: kmp@cs.unc.edu

[1] Arguably it is the job of an underlay service, and not the individual experimenter, to conduct these measurements, as articulated in [6].

an appropriate selection of nodes currently exists, locate it, and conduct the experiment (while it still exists!). This illustrates one advantage of the simulation and emulation approaches — given a sufficiently large blank slate, almost any experiment may be crafted. Indeed, an automated version of mapping and selection tools are already an integral part of the Netbed emulation environment [12] (discussed below), giving a Netbed experimenter the convenience of "just specifying a scenario". Our work advocates a similar approach to that implemented in Netbed, albeit with a different methodology for specifying constraints, conducting measurements, and directing search.

In building a framework for automated *embedding services*, we decompose the issues into three categories: topology specification, testbed characterization, and embedding discovery. First, a researcher must concisely specify the relevant experimental parameters, e.g. the allowable values for metrics applied to each end-to-end path. Second, the current conditions of the testbed must be sufficiently well characterized to determine whether or not a specific embedding meets the specified parameters. Finally, the resulting characterization of the testbed must be searched for an embedding matching the specification. While these are conceptually different steps, we envision a synergistic relation between them in practice. In particular, our framework allows for the specification together with a search in progress to guide additional measurements for characterization. Simultaneously, the current characterization can guide discovery towards embeddings that may be realized without additional measurements.

Individually, each of these categories has been considered in its own right. Extensive experience with simulation and emulation (e.g. ns2 [7] and Netbed [12] respectively) has developed sophisticated interfaces and tools for topology specification. We advocate either reusing or extending these methods for our study as appropriate. The metric-induced network topology (MINT) framework [1] provides another tool for describing and characterizing topologies by abstracting away low-level path details while still capturing the characteristics of end-to-end paths and their interactions with each other.

On the measurement side, there is considerable literature about how to perform measurements to characterize an entire network (e.g. [10]), but relatively little about extracting key details and providing those details as part of a service. The recent "underlay" proposal [6] motivates shared infrastructure for issues such as routing decisions in overlay networks. This same approach is equally useful for testbed characterization, especially when shared across multiple experiments running on the testbed.

The problem of identifying suitable embeddings of test topologies in emulation environments has been a primary focus of Netbed [12], [11], [9]. Ricci, Alfeld and Lepreau describe the design of a solver for the problem of mapping a (potentially very large) desired experimental scenario onto a testbed of smaller scale [9]. Variants of this solver (`wanassign` for wide-area networks in [12] and wireless networks in [11]) are directed at the same general problem we consider: mapping a test topology with a set of user-specified wide-area constraints onto a set of physical nodes. Their solver uses optimization methods to identify a topology minimizing an appropriately chosen distance function between the attributes of the desired topology and the selected topology. This *optimization* approach contrasts with the *constraint satisfaction* approach we employ, which we argue affords a number of advantages for this application. First, an optimization approach requires the user to choose the relative importance of each link attribute to define an appropriate notion of distance between topologies. In contrast, with a constraint satisfaction approach, an experimenter can simply realize a binary yes-no guarantee on a set of experimental conditions in which the error tolerance is prescribed in advance. A second advantage of constraint satisfaction is that it readily allows statistical *sampling* from the space of satisfiable embeddings, which can be useful in assessing representative outcomes. Finally, a constraint satisfaction approach can admit a more parsimonious use of measurements than optimization, as it may be relatively easier to conduct fewer total measurements to find a satisfactory embedding than to find the "best" embedding.

The remainder of this paper proceeds as follows. We first develop a constraint-based problem statement in Section II and discuss both the computational complexity of the embedding discovery problem and practical heuristics for reducing and managing complexity. Section III then examines testbed characterization and methods for reducing the number of measurements performed. Finally, Section IV presents the results of several proof-of-concept experiments conducted on PlanetLab.

II. Embedding Experimental Topologies in Testbeds

We now introduce our topology embedding framework and sketch methods for practical implementation.

A. Problem Statement

Suppose we have a desired embedding consisting of k nodes and their incident links. Starting from the same formulation considered in `wanassign` [12], we specify the desired embedding as a $k \times k$ *constraint matrix* $C = \{c_{i,j}\}$ where $c_{i,j}$ defines a set of constraints on the interconnection between i and j. In practice, constraints are likely to be vectors characterizing overlay link attributes across a set of different dimensions. Our methods apply to complex, multi-attribute constraints, and can also incorporate the placement of additional constraints on nodes rather than links. For simplicity in the technical discussion however, we will discuss constraints taking the form of an upper and lower bound on a single metric, measured round-trip time (RTT). The constraint

$c_{i,j} = [l_{i,j}, h_{i,j}]$ in such a specification requires that we embed nodes i and j as nodes x and y in the testbed such that $l_{i,j} \leq d(x, y) \leq h_{i,j}$, where $d(x, y)$ is the measured RTT. If the RTT between a pair of nodes is not important, one may use the entry $[0, \infty]$ to match any latency.

Complementing our specification is a matrix characterizing the n-node testbed topology. We describe the result of measurements that have been conducted in an $n \times n$ *inference matrix* M. M is composed of entries $m_{i,j}$, each of which is again a range $[l'_{i,j}, h'_{i,j}]$, but each $m_{i,j}$ bounds the *measured* RTT between *real* nodes i and j. In the event we can conduct end-to-end measurements between a pair of nodes i and j, we assume that we can use these to specify the entry $m_{i,j}$ tightly, i.e. such that $l'_{i,j} = h'_{i,j}$. However, in many situations, conducting measurements for all pairs may be impractical or infeasible, so tight bounds on $m_{i,j}$ may not be known. Later, in Section III we discuss methods (specifically applicable to delay constraints), that enable us to infer bounds on many $m_{i,j}$ entries from a much smaller set of measurements. This allows us to be selective with respect to conducted measurements. Since our embedding service is compatible with an inexact characterization of the underlying topology, we need only conduct a set of measurements that characterize the topology *sufficiently well* to locate a desired embedding. The connection between constraints and inferences drawn from measurements is captured in the following definition, which motivates our problem statements.

Definition 1—Feasible Embedding: Given a constraint matrix C, and an inference matrix M associated with a graph G, a *feasible embedding* is a one-to-one mapping f from the k specification nodes to k of the vertices in G such that for all i and j, $l_{i,j} \leq l'_{f(i),f(j)} \leq h'_{f(i),f(j)} \leq h_{i,j}$.

We now consider the following natural problems.

Problem 1—Locating a Feasible Embedding: Given a constraint matrix C and an inference matrix M on G, search for a feasible embedding of C onto G.

Problem 2—Sampling from Feasible Embeddings: Given a constraint matrix C and an inference matrix M on G, select an embedding of C onto G uniformly at random from the space of feasible embeddings of C onto G.

Before we proceed, we confirm the reader's suspicion that the problems we have motivated have high worst-case complexity. Indeed, in the simple case in which the inference matrix and constraint matrix consist of 0/1 values and represent adjacencies and non-adjacencies, then finding a feasible embedding is equivalent to the subgraph isomorphism problem, a well-known NP-complete problem [4]. Also, from well-known connections between sampling and counting, the second problem we consider is $\sharp$P-complete. However, in practice, we find that many interesting problem instances do not lie close to the boundary of solubility and insolubility, and for these instances, the heuristics we describe can easily scale up to experiments involving hundreds of nodes. For example, large constraint matrices will typically have both recursive structure and sparsity, which we exploit in Section II-B to speed up search.

B. Search Strategies

We now discuss search strategies for discovering embeddings. Since the embedding problem is NP-complete, we cannot guarantee worst-case performance substantially better than brute force search. Nevertheless, in practice, our methods are effective for testbed topologies up through at least hundreds of nodes.

A naive brute force approach to discovering embeddings considers each of the $\binom{n}{k}$ sets of k real nodes, and compares each permutation to see whether it satisfies the entries in the constraint matrix. Such a method takes $O\left(\binom{n}{k} k! k^2\right)$ time, While this is clearly impractical by itself, our proof of concept implementation is based on such a simple enumeration using depth-first search. Coupled with various pruning methods, this enumeration-based approach can be practical for discovering feasible embeddings on current testbeds as will be demonstrated in Section IV.

The first pruning method we considered aims to reduce the number of sets of real nodes enumerated. Here, a simple idea is quite powerful in practice - if the partial set of nodes chosen so far cannot be embedded into the constraint matrix, then prune the search at that point. These partial embeddings can be computed incrementally and are essentially smaller instances of the original embedding problem.

The second pruning method is based upon the observation that many simple constraint matrices have a regular structure that allows multiple equivalent mappings of constraints onto the same set of real nodes. By reducing these sets of equivalent mappings between constraints and the nodes in the testbed to a single representative, the search space can be substantially reduced. The pruning method we apply leverages concepts from group theory, namely automorphism groups. Briefly, an automorphism is a renaming of nodes which preserves the structure, which in this case is the constraint matrix C. By computing the automorphism group of C, we can identify equivalent nodes within C and reduce the search space looking for embeddings into C accordingly. While it is not known whether automorphism groups can be computed in polynomial time (it is well known to be equivalent to graph isomorphism), the fastest practical graph isomorphism programs can handle graphs with thousands of nodes. One of these, Nauty [5], already uses automorphisms as a means of pruning its search tree.

III. Inferences from Testbed Measurements

As suggested in Section II-A, our search methods do not require the precise values of metrics on overlay paths.

If we can infer sufficiently tight bounds on a path's actual value from a set of other measurements, we can avoid performing measurements along that path. Therefore, an important research direction related to search is that of minimizing the number of measurements needed to locate a candidate topology. Here, we formalize and motivate a set of such inference problems that are independent of the underlying metric, and then consider inference methods specific to delay constraints.

A. *Optimizing Measurement Orderings*

We first assume that conducting a measurement over a given overlay edge (i, j) gives us the value $d_{i,j}$ and enables us to set $h_{i,j} = l_{i,j} = d_{i,j}$. In practice, this notion of conducting a measurement might entail sending a number of probes. Next, we define the following quality measure on inference matrices, noting that many other alternative measures are also reasonable.

Definition 2: For a given inference matrix M, we say that the *mean range* of M is

$$\mu(M) = \sum_i \sum_j \frac{h_{i,j} - l_{i,j}}{n^2}.$$

Definition 3: Given a set of measurements and a set of deterministic inference methods, the *minimal inference matrix* is the unique inference matrix with minimal mean range that is consistent with the measurements and inference methods.

This motivates the following problem in choosing a measurement ordering, which we state informally, as we do not present any theoretical results. Experiments which demonstrate the effectiveness of employing heuristics for this problem with respect to latency are described in Section IV.

Problem 3: Given a set of inference methods and a set of already conducted measurements $d_1, d_2, \ldots d_x$, efficiently select measurement d_{x+1} such that the resulting minimal inference matrix $\hat{M}$ minimizes the expectation of $\mu(\hat{M})$.

B. *Inference Techniques*

We now sketch deterministic inference methods specific to latency constraints to motivate our ability to derive good bounds on inference matrix entries while conducting far fewer than $\binom{n}{2}$ measurements (assuming symmetry and no self measurements). These methods assume that latency follows the triangle inequality, $d_{i,j} \leq d_{i,k} + d_{k,j}$.

Our measurements indicate that the percentage of triples of PlanetLab end-systems that violate the triangle inequality is approximately 4.4%, a value that is consistent with other recent measurements. The methods we present later in this section generalize to use various relaxations of the triangle inequality. One such relaxation is to assume that $d_{i,j} \leq a(d_{i,k} + d_{k,j}) + b$, for constants $a \geq 1$ and $b \geq 0$. For example, the number of violations on PlanetLab is less than 1% of all triples with either $a = 1.15$ and $b = 1$ ms or $a = 1.09$ and $b = 5$ ms. However, we also note that there a handful of individual triples that badly violate the triangle inequality, and these might have to be monitored and handled separately.

Our methods first decompose the inference matrix M into two separate upper and lower bound matrices H and L, composed of entries $h_{i,j}$ and $l_{i,j}$ respectively. We perform inferences on these matrices separately. Inferences on the H matrix are straightforward: given measurements $d_{i,j}$ and $d_{j,k}$, we can apply the triangle inequality directly to set $h_{i,k} = d_{i,j} + d_{j,k}$, if this reduces the value of $h_{i,k}$. Similarly, recursive applications of this rule allow us to set $h_{i,k} = h_{i,j} + h_{j,k}$ if this improves $h_{i,k}$. In fact, all of the upper bounds in the minimal inference matrix can be obtained efficiently by running an algorithm for the All Pairs Shortest Paths (APSP) problem on the H matrix. To do so, we initialize the $h_{i,j}$ entries to $d_{i,j}$ when $d_{i,j}$ is known and ∞ otherwise, then use these $h_{i,j}$ entries as link weights in the APSP computation. Upon completion, the APSP matrix is our new upper bound matrix H.

Our method for deriving bounds on L is somewhat less obvious. As with the H matrix, we initialize entries in L where we have conducted measurements to those measured values. The remaining entries are set to zero. We then note that by the triangle inequality, for all i, j, k:

$$\begin{aligned} d_{i,j} + d_{j,k} &\geq d_{i,k} \\ d_{i,j} &\geq d_{i,k} - d_{j,k} \geq l_{i,k} - h_{j,k} \\ d_{i,j} &\geq \max_k (l_{i,k} - h_{j,k}) \end{aligned}$$

Therefore, both $\max_k (l_{i,k} - h_{j,k})$, and by a similar argument, $\max_k (l_{k,j} - h_{k,i})$, serve as new lower bounds for $d_{i,j}$. So we can assign $l_{i,j}$ to the larger of these two quantities. If we rewrite these formulas in terms of $-l_{i,j}$, then we get $-l_{i,j} = \min_k ((-l_{i,k}) + h_{j,k})$ or $-l_{i,j} = \min_k (h_{k,i} + (-l_{k,j}))$. This is now very similar to the matrix multiplication formulation of all pairs shortest paths. In fact, using a known variant of matrix multiplication in which we substitute operators $(\min, +)$ for $(+, \cdot)$, we can compute all of the lower bound inferences with two matrix multiplications.

While we have simple heuristics for applying these inference methods to optimize measurement orderings, a provably robust solution is an open problem that we leave for future work. We briefly sketch the power of using inferences to minimize the number of measurements conducted in Section IV.

IV. Experimental Results

As a proof of concept, we prototyped a centralized version of the embedding service. We used a snapshot of PlanetLab pair-wise latencies from July 12, 2003. Starting from the production node list, nodes not responding to pings were first removed. As measurements were

Clique Size	2	3	4	5	6	7	8	9	10	11	12
0-10 ms cliques	8/403	5/936	9/1475	17/1645	8/1327	8/771	8/315	3/86	3/14	1/1	0/0
1-10 ms cliques	35/325	61/501	84/387	60/142	20/20	0/0	0/0	0/0	0/0	0/0	0/0

TABLE I

EMBEDDING LOW LATENCY CLIQUES (NUMBER OF MAXIMAL CLIQUES/TOTAL NUMBER OF CLIQUES).

performed, nodes with full file systems and CPU loads over 2.0 (measured with uptime) were also removed from the list, leaving 118 nodes for our experiments. All the searches we report on below took less than 30 seconds on a 1GHz Pentium 3 processor.

First, we demonstrate the feasibility of embedding topologies within PlanetLab in Section IV, along with some intriguing results regarding the lack of certain embeddings. We then show the results of preliminary heuristics for adaptive probing.

A. Finding Low Latency Cliques

Our first set of experiments searches for variable size cliques with at most 10 ms RTT between each node (i.e. $\forall_{i \neq j}(c_{i,j} = [0, 10])$). A set of nodes matching such a topology might be ideal for a tightly synchronized distributed protocol requiring quick response times from peers. We note that finding cliques is also a well-known NP-complete problem [4], but due to the irregular nature of the measured PlanetLab topology and the comparatively small clique sizes, we find it is actually quite easy to find such cliques in practice (the longest searches took less than 30 seconds).

The first row of Table I shows the number of cliques of each size found, where each entry a/b means that there are a maximal cliques of that size and b is the total number of cliques of that size (a maximal clique is not a subgraph of a larger clique). Despite 10 ms being a low RTT value for the Internet as a whole, we find many such cliques since PlanetLab nodes tend to be well connected. However, the number of maximal cliques is much smaller than the total number of cliques – most of the smaller cliques are part of larger clusters.

The unique clique of size 11 found in Table I only contained nodes from six distinct institutions. To avoid using co-located nodes, we introduce a lower bound of 1 ms, so $\forall_{i \neq j}(c_{i,j} = [1, 10])$. The results of applying this restriction to the search are shown in the second row of Table I. Now, the largest clique size is 6 and there are twenty feasible embeddings. However, five institutions are represented in each of the twenty feasible embeddings, and only two other institutions fill in the remaining position. One interpretation of these results is that there is less clustering in the PlanetLab graph if we look at institutions instead of nodes.

Our next experiments extend the search to find multiple low latency cliques, keeping the 1ms lower bound, and adding the requirement that nodes in different cliques have between 20 ms and 50 ms latency so that the cliques are physically separated, but not too far apart. Figure IV-A gives an example constraint matrix C for three groups of two nodes each. Such a scenario is natural (although artificially easy) for evaluating the effectiveness of topological optimizations in an overlay routing protocol.

Number of Cliques	Clique Size 2	3	4	5	6	7
1	325	501	387	142	20	0
2	6898	6238	1004	0	0	0
3	12950	0	0	0	0	0
4	0	0	0	0	0	0

TABLE II

EMBEDDING MULTIPLE CLIQUES.

Table II shows the number of embeddings for various numbers of cliques and clique sizes. Curiously, there are very few ways to embed multiple separate cliques within PlanetLab. This may have significant implications for PlanetLab researchers who value such embeddings and could inform future node placements in order to widen this space of embeddings.

B. Informed Probing

Our final experiments show the potential of some simple heuristics for adaptive probing. The initial measurement matrix was constructed by min-filtering a set of three ping measurements along each end-to-end path to limit the impact of traffic bursts and other outlying measurements. A few entries in the resulting matrix were missing due to measurement failures, so the corresponding nodes were removed. We then removed the small set of triangle inequality violations from the dataset in two steps. Most violations were minor and were eliminated simply by quantizing the measured RTTs (rounding up RTTs to multiples of 5 ms). Nodes which continued to violate the triangle inequality on at least one neighboring

$$\begin{array}{|cccccc|}
[0,0] & [1,10] & [20,50] & [20,50] & [20,50] & [20,50] \\
[1,10] & [0,0] & [20,50] & [20,50] & [20,50] & [20,50] \\
[20,50] & [20,50] & [0,0] & [1,10] & [20,50] & [20,50] \\
[20,50] & [20,50] & [1,10] & [0,0] & [20,50] & [20,50] \\
[20,50] & [20,50] & [20,50] & [20,50] & [0,0] & [1,10] \\
[20,50] & [20,50] & [20,50] & [20,50] & [1,10] & [0,0]
\end{array}$$

Fig. 1. A clustered constraint matrix for three cliques of two nodes.

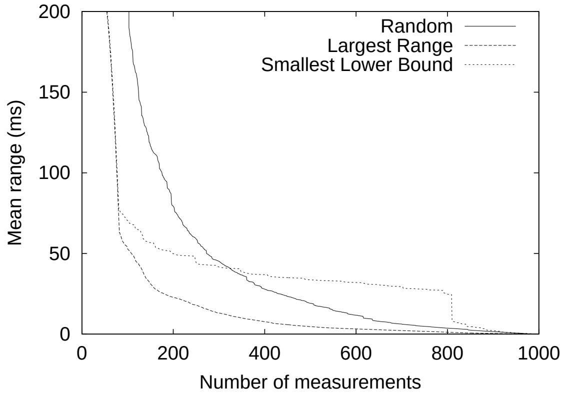

Fig. 2. Benefits of inference and adaptive probing.

triple were thrown out of the dataset. The resulting matrix had complete all-pairs data for 45 nodes.

We compared various simple heuristics against performing all $\binom{n}{2}$ measurements in random order and show that these heuristics significantly lower $\mu(M)$. We simulated adaptive probing by starting with a blank inference matrix and filling in measurement results one entry at a time. As a baseline, a random permutation of the $\binom{n}{2}$ pairs of nodes was chosen. For each pair of nodes, the corresponding measurement was then entered into the inference matrix and inferences were performed as discussed in Section III. If the corresponding measurement had already been derived, i.e. $h_{i,j} = l_{i.j}$, then the measurement was skipped. We considered the following heuristics:

Largest Range. Measurements on edges with the largest range ($h_{i,j} - l_{i,j}$) were given precedence, breaking ties in favor of smaller lower bounds.

Smallest Lower Bound. Measurements on edges with the smallest lower bounds were given precedence, breaking ties in favor of larger ranges.

Figure IV-B shows the results of this experiment, plotting $\mu(M)$ as the measurements progress. Both of our heuristics perform significantly better than random choices for the first hundred or so measurements. However, *Smallest Lower Bound* falters thereafter and is eventually surpassed by the random choice method. *Largest Range* consistently achieves the lowest $\mu(M)$ values, and has two abrupt transitions in its slope. We conjecture that these changes may be related to the distribution of RTTs, but this remains a topic for further study.

V. Summary and Future Directions

As use of wide-area Internet testbeds matures as an experimental methodology, services for performing predictable and repeatable experiments will be required. In this paper, we focused on the theoretical foundations of a topology embedding service based on constraint satisfaction. Our approach allows us either to select a feasible embedding of an experimental specification in a wide-area testbed or to sample from the set of feasible embeddings. Although these problems are intractable in the worst-case, the heuristic methods we propose are practical for problem instances of moderate size.

An effective embedding service can also exploit connections between searching and conducting measurements. Our proposed inference techniques reduce the number of measurements required to discover a particular embedding by allowing the search to inform how and when measurements are made. Meanwhile, measurements performed can guide the search towards regions where feasible embeddings are likely to be found with the fewest measurements.

A number of important implementation issues remain for future investigation. These challenges include dealing with large topologies, developing distributed mechanisms for finding feasible embeddings, developing constraint specifications and search methods that allow approximate results, and integrating with existing testbed tools and services.

Acknowledgments

The authors thank Azer Bestavros, Kevin Jeffay, Jasleen Kaur, and Richard West for extensive and helpful discussions on the topics in this paper. The authors are also grateful to Jay Lepreau and Robert Ricci for providing valuable feedback on our work and for detailing the various embedding methods used in Netbed.

References

[1] A. Bestavros, J. Byers, and K. Harfoush. Inference and Labeling of Metric-Induced Network Topologies. In *Proceedings of IEEE Infocom '02*, New York, NY, June 2002.

[2] S. Floyd and E. Kohler. Internet research needs better models. In *Proceedings of Hotnets-I*, Princeton, NJ, October 2002.

[3] S. Floyd and V. Paxson. Difficulties in simulating the internet. *IEEE/ACM Transactions on Networking*, 9(4):392–403, August 2001.

[4] M. Garey and D. Johnson. *"Computers and Intractibility: A Guide to the Theory of NP-Completeness."*. W.H. Freeman, New York, 1979.

[5] B.D. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.

[6] A. Nakao, L. Peterson, and A. Bavier. A routing underlay for overlay networks. In *Proceedings of ACM SIGCOMM*, pages 11–18, August 2003.

[7] `ns`: UCB/LBNL/VINT Network Simulator (version 2). http://www-mash.cs.berkeley.edu/ns/ns.html.

[8] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proceedings of Hotnets-I*, Princeton, NJ, October 2002.

[9] R. Ricci, C. Alfeld, and J. Lepreau. A solver for the network testbed mapping problem. *Computer Communications Review*, 33(2), April 2003.

[10] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. In *Proceedings of ACM SIGCOMM*, pages 133–145, August 2002.

[11] B. White, J. Lepreau, and S. Guruprasad. Lowering the Barrier to Wireless and Mobile Experimentation. In *Proceedings of Hotnets-I*, Princeton, NJ, October 2002.

[12] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth USENIX Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, December 2002.