On the Utility of Inference Mechanisms

Ethan Blanton Sonia Fahmy Greg N. Frederickson
Department of Computer Science, Purdue University
West Lafayette, Indiana, USA
E-mail: {eblanton, fahmy, gnf}@purdue.edu

Abstract

A number of network path delay, loss, or bandwidth inference mechanisms have been proposed over the past decade. Concurrently, several network measurement services have been deployed over the Internet and intranets. We consider inference mechanisms that use $O(n)$ end-to-end measurements to predict the $O(n^2)$ end-to-end pairwise measurements among n nodes, and investigate when *it is beneficial to use them in measurement services. In particular, we address the following questions: (1) For which measurement request patterns would using an inference mechanism be advantageous? (2) How does a measurement service determine the set of hosts that should utilize inference mechanisms, as opposed to those that are better served using direct end-to-end measurements? (3) How can the answer to question 2 be efficiently computed as measurement requests arrive and terminate? Our solution is able to identify groups of hosts which are likely to benefit from inference, by utilizing a probabilistically generated spanning forest on the measurement request graph. We compare our solution to a simple heuristic that uses the number of measurements a host participates in. Results with synthetic datasets as well as datasets from a popular peer-to-peer system demonstrate that our technique identifies host subsets that benefit from inference quite accurately, and in significantly less time than an algorithm that identifies optimal subsets. The measurement savings are large when measurement request patterns exhibit small-world characteristics, which is often the case for peer-to-peer and other popular distributed systems.*[1]

1. Introduction

An important class of network inference mechanisms estimate the properties (*e.g.,* delay or loss) of a large number of end-to-end network paths by measuring some subset thereof.[2] This class of mechanisms is designed to reduce the amount of injected active measurement probe traffic and the effort required to collect a large number of measurements, typically at the expense of measurement accuracy. For example, the Azureus BitTorrent client can use inferred network delay information to select peers from which to transfer data [1]. The question of how much of a reduction in measurements the existing inference mechanisms achieve for different measurement request patterns has not been adequately studied.

A network measurement service, which provides measurement results to applications on request, is uniquely suited to utilizing network inference mechanisms. Examples of network measurement services include ScriptRoute [2], the Scalable Sensing Service (S^3) [3], iPlane [4], and the system by Calyam *et al.* [5]. Because a measurement service has knowledge of a larger number of network measurements than individual applications, it is in a position to determine *when* inference can be used to reduce the total number of measurements required to satisfy a particular demand from applications. To accomplish this, the service must quantify the measurement load required to set up and operate a network inference mechanism, and compare this load to that introduced by direct measurement of requested properties [6].

In this paper, we predict the network traffic injected by inference mechanisms, and compare it to the traffic injected by requested direct measurements. We present an efficient method for *identifying opportunities when inference induces less traffic than a given pattern of direct measurements*. Our method is *not* an inference mechanism, but a tool for deploying existing inference mechanisms dynamically on the hosts where their use is advantageous. We note that setting up an inference mechanism may incur a non-negligible cost [7]. After startup, continuing measurements typically require $O(n)$ probes to estimate properties of $O(n^2)$ paths [7], [8], [9], [10]. Our work hinges on the two observations that (1) there is a hidden constant for the $O(n)$ probes, which can be large, and that (2) oftentimes, not all the $O(n^2)$ path properties are requested.

1. This research is sponsored in part by a gift from Hewlett-Packard and NSF CAREER grant 0238294. We would like to thank Sujata Banerjee, Puneet Sharma, and Praveen Yalagandula (HP Labs) and Sriharsha Gangam (Purdue) for several helpful discussions on this work. We would also like to thank Chuan Wu and Baochun Li for sharing with us their UUSee node degree data.

2. The terms *inference* or *tomography* are also used to refer to the inference of properties of internal (router-to-router) links from purely end-to-end (i.e., host-to-host) measurements. Such mechanisms are not under discussion in this paper.

The remainder of this paper is organized as follows. Section 2 gives some definitions. Section 3 defines the problem that we are addressing. Section 4 gives our solution and analysis. Section 5 gives results from our experiments with both real and synthetic datasets. Finally, Section 6 summarizes our results and directions for future work.

2. Terminology

When we discuss *measurements*, we generally mean *active* network measurements that require injecting probe packets into the network, *e.g.*, ping packets to measure end-to-end delay. A *measurement service* is a network service that accepts *requests* for the results of active measurements from applications on-demand, schedules *measurement tools* to be run to service these requests, and ultimately returns the results from the tools to the application. A *measurement host*, or simply *host*, is an infrastructure host in the measurement service that invokes measurement tools and records their output, to be delivered to applications. Measurement *endpoints* are the hosts which source and/or sink traffic in performing a given measurement.

The inference mechanisms we consider can be broadly divided into two categories. The first category includes mechanisms that use knowledge of link-level or autonomous system (AS)-level paths to choose a subset of paths to be probed. We will call these *path-based* inference mechanisms. Inference mechanisms in this category include Chen's Algebraic method [11], IDMaps [12], NetQuest [13], and iPlane [4]. The second category consists of mechanisms which infer pairwise properties using measurements to reference nodes (which may or may not be chosen *a priori*). We will term these mechanisms *reference-based* inference mechanisms. Mechanisms in this category include Vivaldi [8], GNP [7], and Theilmann *et al.*'s Dynamic Distance Maps [9]. Our primary focus in this paper is on mechanisms in this latter category. We identify the pattern of direct measurement workload required to equal or exceed the network load of the inference mechanism, and specify advantageous replacements of direct measurements.

3. When to Use Inference?

Of the reference-based inference mechanisms, many require a number of measurements that scales linearly with the number of hosts participating in the inference. Some mechanisms perform a constant number of measurements per host, *e.g.*, [9], [8]. Others perform varying numbers of measurements per host, but average a constant number per host, *e.g.*, [7], [10]. In this latter group, typically the majority of hosts participate in a constant number of measurements, and a constant number of hosts (such as the so-called landmarks from GNP [7]) participate in a large number of measurements (linear in the number of hosts). Some of the algorithms have an additional cost for initial construction, which we will not consider in this paper.

Assume that we can identify or approximate the constant in a network inference algorithm that requires, on average, a *constant* number of network measurements *per host* to infer all-pairs measurements among participating hosts. We will call this constant k, call the number of hosts participating in the inference n, and call such an inference mechanism a kn-cost inference. The authors of the GNP delay inference mechanism, for example, recommend that hosts take measurements to 15 landmarks [7], and the authors of the Vivaldi delay inference mechanism recommend a selection of 32 neighbors [8]. For these two systems under their recommended configurations, k would be 15 and 32, respectively. Observe that the constant k is dictated by the workings of the inference mechanism under consideration, and is *not* a tunable parameter in our work. Given a set of measurements requested from a measurement service and the constant k, we can determine the *tipping point* at which the total number of measurements requested becomes greater than or equal to the number of measurements required to perform inference. In this case, inference can reduce the total load on the network, at the cost of reduced accuracy.

If we only use inference when the *total* number of requested measurements exceeds kn, we will miss key opportunities when inference is beneficial. This is because some hosts may be participating in measurements to a large number of hosts, while others may be involved in very few measurements. Consider a situation where n hosts are interested in performing delay measurement to at least one endpoint. Assume that m out of the n hosts are performing a complete all-pairs measurement mesh, where each of the m hosts measures delay between it and the other $m-1$ hosts. Additionally, $n-m$ hosts are measuring delay to only one endpoint each. We therefore have $O(m^2+n-m)$ total measurements. If $m < \sqrt{n}$ and $k \geq 3$, comparing total numbers indicates that direct measurement requires fewer total measurements than inference. However, if $m > 2k+1$, performing inference on the subset of m hosts, and direct measurements for the remaining hosts, would require fewer total measurements than only using direct measurements.

Fig. 1 illustrates this scenario with $n = 12$ and $m = 8$. We represent each host requesting measurement as a node in a graph. A requested measurement between two hosts is represented as an undirected edge in that graph. When $k = 3$, this graph (which has 32 edges) superficially appears to see no benefit from inference, as $32 < 3n = 36$. However, by performing inference among the eight nodes marked in black, we reduce the number of measurements taken to $3 \times 8 + 4 = 28$, realizing savings of four measurements.

If, given a set of requested measurements, we wish to determine whether or not inference can save effort over *any subset* of the participating hosts, we have to answer a slightly different question. For any subset of hosts of size

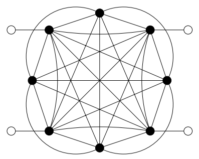
Figure 1. Graph benefiting from partial-graph inference when $k = 3$.

n performing measurements among themselves, if the total number of measurements being performed is greater than kn, then a kn-cost inference mechanism requires fewer total measurements than direct measurement. Using our measurement request graph above, determining whether inference can reduce the total number of measurements is a matter of finding subgraphs for which the number of measurements within each subgraph is greater than k multiplied by the number of nodes in the subgraph. Replacing the direct measurements in these subgraphs with inference will yield a smaller total number of measurements performed.

In other words, given a graph $G = (V, E)$, our goal is to transform it into another graph $G' = (V, E')$ such that we minimize $|E'|$, where $0 \leq |E'| \leq |E|$ and $0 \leq |E'| \leq k|V|$. The only transformation operations allowed on G are replacement of all edges among subsets V_i of V by $k|Vi|$ edges (i.e., employing one of the inference mechanisms in the literature on subsets V_i of vertices, while using direct measurements for remaining edges). Note that a vertex in one of the V_i subsets can still be an endpoint in a direct measurement, as long as the other endpoint does not belong to a subset V_i.

4. Solution and Analysis

A *k-regular* graph is a graph in which each node has a degree of exactly k. An undirected graph where each node has a degree of exactly $2k$ involves kn total measurements, and thus represents the tipping point for a set of hosts using a kn-cost inference. Unfortunately, showing that a general graph contains a k-regular subgraph has been shown to be NP-complete [14]. Therefore, we must find a solution that trades off optimality for tractable computational complexity.

To identify the subgraphs that have sufficient "density" so that replacement of their direct measurements with a kn-cost inference will result in a net reduction, we investigate the following approximation.[3] We use a *set of minimum spanning forests* to identify edges in the request graph (as described in Section 3) that are likely to be a part of low-edge-count cuts of the graph, and prune them. The nodes of the connected components which remain are assumed to be hosts which would benefit from participating in inference.[4]

4.1. Algorithm

The pseudocode for an algorithm to identify groups of hosts which may benefit from inference is presented in Fig. 2. The algorithm takes three parameters as input. The first, G, is an unweighted, undirected[5] graph representing measurement hosts and requested measurements, as described in Section 3. The second and third parameters, f and s, are integer arguments representing parameters for the heuristic itself. Let f be the number of spanning forests used by the algorithm, which balances the tradeoff between computation time and accuracy of the algorithm. Let s be a threshold score used to identify edges which are assumed to be part of a low-edge-count cut of the graph G. As we will show later, the value of s is a function of f and of the constant k of the inference algorithm.

Let $V(G)$ represent the vertices of G, and $E(G)$ represent the edges of G. Let $\overline{vw}$ represent an edge from vertex v to vertex w. Let $wt(\cdot)$ represent the weight of the edge given as its argument.

First, we construct a set of f graphs $\{G_1, \ldots, G_f\}$, identical to the unweighted graph G, except that each edge in these graphs is assigned a weight from a uniform random distribution. We then find a minimum spanning forest F_i for each such graph G_i via the function $\mathrm{MSF}(\cdot)$ (using, *e.g.*, Kruskal's algorithm). Next, we define a score for each edge in G as follows:

$$\text{score}(\overline{vw}) = \sum_{i=1}^{f} \left\{ \begin{array}{lcl} 1 & : & \overline{vw} \in F_i \\ 0 & : & \overline{vw} \notin F_i \end{array} \right.$$

Any edge in G having a score greater than the threshold score s is then removed from G. The connected components of G are calculated by the function $\mathrm{components}(\cdot)$, and each connected component is assumed to represent a set of hosts which would benefit from having measurements internal to the set replaced with inference. Additionally, any two components that were connected via an edge in the original graph are merged into a single component. This is because using inference on the merged graph incurs no additional cost (as the merged graph includes no additional vertices).

4.2. Bounds on the Expected Score

The motivation for using minimum spanning trees with random edge weights is that we want a quick method

3. We will compare this approximation to a simple heuristic that uses inference on all hosts with degree exceeding k in Section 5.2.

4. Note that our approach bears some similarity to (but has important differences from) the Girvan-Newman algorithm for community identification. The Girvan-Newman algorithm has a higher complexity of O($|E|^2|V|$).

5. We consider only undirected graphs in this work. The algorithm presented generalizes readily to directed graphs, at the expense of some clarity.

```
def inference_groups$(G, f, s)$
    for $i \in 1..f$ do
        let $V(G_i) = V(G)$
        let $E(G_i) = E(G)$
        for $e \in E(G_i)$ do
            let $wt(e) = \text{random}()$
        let $F_i = \text{MSF}(G_i)$
    for $e \in E(G)$ do
        if $\text{score}(e) > s$ then
            let $E(G) = E(G) \backslash e$
    return $\text{components}(G)$
```

Figure 2. Pseudocode for probabilistic spanning tree selection of nodes for inference.

for estimating edge-connectivity in a graph. For any edge in the graph, we are able to lower-bound the probability that a given edge will be in a most constricting cut that contains that edge, which leads to a lower bound on the expected score for that edge. In addition, we can upper-bound the probability that the actual score for the edge falls significantly below the lower bound on its expected score.

Let $e = \overline{v_1 v_2}$ be an edge in G, and let (V', E') be the connected component of G that contains edge e. Define vertex sets V_1 and $V_2 = V' - V_1$ such that $v_1 \in V_1$ and $v_2 \in V_2$ and $c(V_1, V_2)$ is minimized, where $c(V_1, V_2)$ is the number of edges $\overline{uw}$ in G such that $u \in V_1$ and $w \in V_2$.

Lemma 1: The probability that edge e is an edge in F_i is at least $1/c(V_1, V_2)$, and the expected score for e is thus at least $f/c(V_1, V_2)$. Furthermore, $\Pr(\text{score}(e) \leq f/c(V_1, V_2) - \alpha\sqrt{f}) \leq \exp(-2\alpha^2)$ for any constant α.

Proof. Consider any of the graphs G_i. Whenever all edge weights in G_i are distinct, an edge in G_i that is of minimum cost among all edges between V_1 and V_2 will be in the minimum spanning forest F_i.

Since the weights of all edges in G_i are chosen randomly, the probability that edge $e = \overline{v_1 v_2}$ is of minimum cost among all edges between V_1 and V_2 is $1/c(V_1, V_2)$. Thus e appears in F_i with probability at least $1/c(V_1, V_2)$.

For $i = 1, 2, \ldots, f$, let X_i be a random variable with $X_i = 1$ if e has minimum weight among all edges $\overline{uw}$ between V_1 and V_2, and $X_i = 0$ otherwise. Clearly, the X_i are independent random variables. Let $S = X_1 + X_2 + \ldots + X_f$. Then E[$S$] = $f/c(V_1, V_2)$. Let X_i' be variables, with $X_i' = -X_i$. Let $S' = -S$. Similarly the X_i' are (amongst themselves) independent random variables.

By Hoeffding's inequality [15], $\Pr(S' - \text{E}[S'] \geq t) \leq \exp(-2t^2/f)$. Then

$$\begin{aligned} \Pr(S' - \text{E}[S'] \geq t) &= \Pr(-S + \text{E}[S] \geq t) \\ &= \Pr(S - \text{E}[S] \leq -t) \\ &= \Pr(S \leq f/c(V_1, V_2) - t) \\ &\leq \exp(-2t^2/f). \end{aligned}$$

Choosing $t = \alpha\sqrt{f}$ gives the claimed result. Note that score(e) $\geq S$, since edge e might not have the smallest weight of edges between V_1 and V_2 and yet still be in F_i. □

As clear from this lemma, the expected score is a function of both $c(V_1, V_2)$, which depends on the structure of the graph, and of the number of forests f, which balances the tradeoff between complexity and accuracy. Based on the bounds on the expected score, the value of the threshold score s must be essentially proportional to f. Additionally, the threshold s must be inversely related to the constant k of the inference algorithm. This is because the higher the value of k, the higher the overhead of the inference mechanism, and therefore the more aggressively we want to prune edges, so that direct measurements rather than inference are used in relatively sparse areas of the graph. Reducing the value of s increases the number of pruned edges, thus increasing the number of direct measurements that will be performed. The relationship between s, f, k, and the structure of the graph is further explored in our experiments in Section 5.

4.3. Complexity

To be useful for on-demand measurement requests in an Internet-scale system, the decision to use inference or take direct measurements must be made rapidly. Traditional methods of computing minimum spanning trees run in $O(|E| \log |E|)$ time (*e.g.*, Kruskal's algorithm or Prim's algorithm), which, for large systems with thousands or tens of thousands of measurement hosts and measurements being requested many times per second, are too expensive to compute for every measurement request entering the system.

In order to make our solution feasible for on-demand measurements, we plan to turn to algorithms that maintain the minimum spanning forest of a graph in the face of dynamic updates in amortized time $O(\log^4 |V|)$ per insertion or deletion [16], or worst-case $O(|V|^{1/2})$ time per operation [17], [18].[6] Efficient handling of on-demand measurements will be a subject of future work.

5. Experimental Evaluation

We experimentally evaluate the algorithm given in Section 4.1 on graphs of various structure. As a baseline, we evaluate our method on a set of simple synthetic topologies, including graphs having uniform random edge placement. Additionally, we utilize graphs based on real datasets from a large-scale peer-to-peer system: the popular UUSee streaming television service [19]. These graphs represent a typical scenario where users select peers or servers with which to communicate based on measured network path properties.

6. We note that $|V|^{1/2} \leq \log^4 |V|$ whenever $|V| \leq 10^{12}$.

The key measure of comparison for this study is the amount of reduction in measurements required between hosts in the graph if we employ inference on the subgraphs ($\mathrm{components}(\cdot)$) that our algorithm outputs and direct measurements on the remaining edges, compared to performing all the requested direct measurements. We also report the cost of performing a single inference on the entire graph. We give the running time of our algorithm, and investigate the values of its two key parameters (number of forests f and threshold score s). We also study the relationship between s and the constant k of the inference mechanism.

5.1. Synthetic Topologies

Fig. 3 depicts a synthetic topology having two complete subgraphs of 8 vertices each, connected by a "bridge" consisting of two edges and a separating vertex. Investigating subgraphs for a kn-cost inference with $k = 3$, an *optimal solution* identifies the nodes marked in black as candidates for inference, and the node marked in white would participate only in direct measurements. Nodes participating in inference make up two connected components, with a single unconnected vertex left over. The direct measurement request graph in Fig. 3 has 58 edges. Performing inference separately on each of the two clusters costs $8k = 24$ edges each. Adding the two direct measurements yields 50 edges. Observe that performing inference on the entire graph using an inference algorithm with $k = 3$ costs $kn = 51$. The savings over this full-graph inference increase with increasing k, and increase when graphs contain more sparse areas (as opposed to the single "bridge" in Fig. 3). It is important to note that *inference comes at the cost of reduced accuracy*, so using it when unnecessary is highly undesirable.

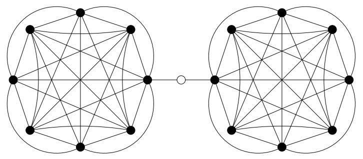

Figure 3. A two-cluster measurement request topology with optimal kn-cost inference groups for $k = 3$ marked in black.

A graph of this general form (densely connected areas separated by sparse regions) is interesting because it is a case when inference is beneficial, but full-graph inference is not the right choice. More importantly, it is a typical measurement request graph due to the characteristics of today's distributed systems. For example, the two clusters in the graph can represent viewers of two peer-to-peer video streaming channels, or downloaders of two files, with only a few users simultaneously participating in more than one streaming channel or download session. This type of topology where channels or downloads are largely, but not completely, disjoint has often been observed in real application scenarios [19], [20]. Many video streaming systems have an option, picture-in-picture, that allows viewing one channel at a high resolution, while viewing small window(s) showing (an)other channel(s).

Evaluation on this sample graph illustrates that, even for small values of f (stable results appear at $f = 5$), the correct subgraphs are identified as candidates for inference. Fig. 4 shows the average value of s above which the correct subgraphs benefiting from inference are identified in every case, for increasing values of f. The plot illustrates that, as expected, s scales sub-linearly with f (at about $0.24f$ for the plotted values of f). For each value of f, the minimum and maximum s in our experiments where correct subgraphs are identified are shown. It can be seen that the values of s are stable, and fall within a narrow range of about $0.02f$.

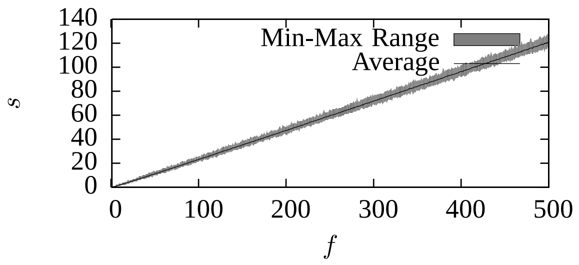

Figure 4. Number of forests (f) versus the threshold score (s) required to identify correct inference components, averaged over 10 runs. Min. and max. s also plotted.

Computing the optimal inference groups for any given graph is NP-hard, as discussed in Section 4. Computation of optimal inference groups on this 17-node topology takes about 450 ms on a 1.8 GHz processor; by comparison, our spanning forest algorithm takes about 5 ms. As the graph grows, this difference in computation time becomes larger. For a graph of only 32 nodes, the optimal algorithm requires 14 hours and 52 minutes of processing, while the running time of our algorithm is still a few milliseconds.

The graph in Fig. 3 is an interesting case study, but presents a trivial case for our algorithm, since both edges of the bridge between the two fully connected subgraphs will always have a score equal to f. To further explore the relationship between f, s, and k, we build a number of graphs on the pattern in Fig. 3, but with a variable number of bridges between the two fully connected subgraphs. Each bridge is configured analogous to the white node in Fig. 3, having two adjacent edges, one rooted in each of the fully connected subgraphs. No pair of nodes is directly connected by more than one bridge. Fig. 5 illustrates the average values of s bracketing the correct inference recommendation for $f = 50$ and $k = 3$. For reference, $s = 17 = f/k$ is

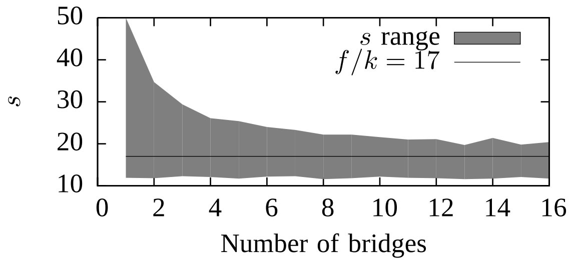

Figure 5. Number of bridges between fully-connected subgraphs vs. range of s values yielding an optimal recommendation.

depicted as well. This graph shows that, as each bridge becomes relatively lighter in weight for a given value of f (due to the effect of additional bridges between the fully connected subgraphs), the range of values of s yielding a correct recommendation converge toward a value near f/k. As mentioned above, the total savings in measurements compared to full-graph inference grows linearly with the number of bridges in the graph, at $kn-(2km+2\times bridges)$, or one edge per bridge for $k = 3$. The value $2km$ in this computation represents inference on the two fully connected subgraphs of m nodes each. In the graph having 16 such bridges, this represents a savings of 16 measurements over full-graph inference, or a 17% savings in measurements performed (80 versus 96 measurements).

5.2. Selection by Degree

A straightforward approach to identifying hosts which could benefit from inclusion in a kn-cost inference is to select all hosts which have a degree greater than or equal to k. While preliminary experiments show that this approach works well on many graphs, there exist graphs for which it fails.

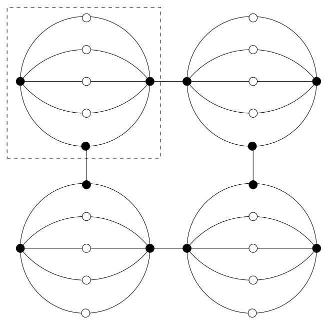

Figure 6. Highly connected nodes separated by sparsely connected nodes.

One graph structure for which this method fails is depicted in Fig. 6. This graph consists of a number of highly connected nodes separated by nodes of low degree, such that there exist no large clusters of highly connected nodes. A simple selection on node degree for $k = 3$ will select the nodes marked in black for inference, despite the fact that this yields a larger total number of measurements than direct measurement. In contrast, employing our algorithm across two by two and three by three lattices of the structure in the dotted box from Fig. 6 (arranged as in Fig. 6, as well as short linear "chains" of the same structure attached between the black nodes of degree six) yields the optimal result of no vertices recommended for inference for values of f as low as 35, with $s = f/k = 11$. As expected, decreasing f decreases the accuracy of the algorithm, and for smaller values of f, some vertices are recommended for inference for values of $s \leq f/k$.

5.3. UUSee Topologies

The Magellan project [19] characterized the connections between peers of the UUSee live streaming video service, which is highly popular in China. They found that the UUSee graphs exhibit *small-world* properties. A small-world graph is characterized by two important properties: (1) a small average path length between any pair of nodes, and (2) a relatively large clustering coefficient, indicating that there is high connectivity between neighboring nodes. Studies of the Gnutella peer-to-peer network have also shown that it exhibits small-world characteristics in client peering [20].

We generate topologies using the small-world topology generator described by Jin and Bestavros in [21]. We use the node degree distribution information of the UUSee service reported in [19], and set the local preference parameter p to 0.5.[7] Magellan found that the topology representing all UUSee channel viewers included around 100,000 viewers, with approximately one third of these being stable. The average path length in the UUSee topologies of stable viewers of all channels was close to 5 hops, while the clustering coefficient was close to 0.3, which is more than an order of magnitude higher than typical clustering coefficients of random graphs. The graphs of the different channels (up to 800 channels) were reported to be largely disjoint [19].

We generated UUSee-inspired topologies to correspond to viewers of a typical UUSee channel. In [19], one channel was reported to have about 2500 viewers. We validated the clustering coefficient of our graphs, which was found to be approximately 0.25, and the average path length which was close to 2.3. Each UUSee-inspired topology used in our experiments in this section had 2500 vertices and in the neighborhood of 53,000 edges. We also experimented with graphs representing multiple channels, and with smaller topologies (results not reported here for brevity).

Fig. 7 depicts the number of measurements required to fulfill the mixture of inference and direct measurement rec-

7. We examined a number of graphs having values of p between 0.05 and 1.0 (in increments of 0.05), and the results were consistent.

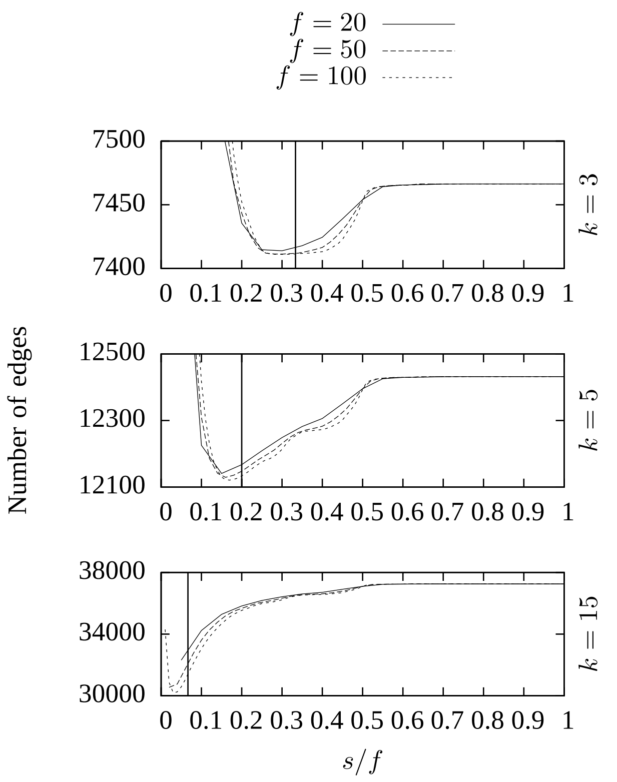

Figure 7. UUSee topologies: Total measurements taken (edges). x axis gives s/f. A vertical line is marked on each plot at $x = 1/k$.

ommended by our algorithm on UUSee-inspired topologies. Each plot is an average over ten random UUSee topologies. The y axis of these plots is truncated; for very small values of s, no inference is recommended or inference is recommended on very small clusters of vertices, and thus the number of edges in the resulting graph approaches the 53,000 edges of the original topology. The figure illustrates substantial savings over the original 53,000-edge topology, and increasing savings over performing inference on the entire graph as k increases. The number of edges is slightly lower with larger values of f, validating the hypothesis that increasing f increases algorithm accuracy, at the cost of increased computation time.

As seen in the figure, for each value of k, the total number of measurements required falls rapidly as s approaches somewhat less than f/k. The number of measurements climbs toward kn as s approaches f and the algorithm recommends that nearly all vertices participate in inference. The minimum number of resulting measurements occurs for all three values of k (as well as other values of k not depicted here) at a value of s slightly smaller than f/k. This coincides with the finding of Section 5.1, where s of at least $0.24f$ yielded correct results on the synthetic topology with one bridge for $k = 3$. The result also agrees with the discussion in Section 4.2, suggesting that s must increase with increasing values of f, and decrease with increasing values of k.

Computation time for our algorithm is manageable for these topologies. For $f = 20$, computation on the same 1.8 GHz processor as referenced in Section 5.1 takes about six seconds. For $f = 50$, computation takes about 15 seconds, and $f = 100$ takes about 30 seconds. As discussed in Section 4.3, this is tractable computation in comparison to the optimal algorithm for the NP-hard problem. However, it underscores the need for incremental computation with on-demand measurement requests.

5.4. Uniform Random Edge Placement

In this section, we consider graphs which have uniform random edge placement. For each pair of vertices in the graph, an edge is present with probability $0 < p \leq 1$. Such graphs exhibit roughly uniform edge density across all vertices and all subsets of vertices, and, as such, tend to either not benefit from inference at all, or benefit from a full-graph inference including all vertices.

A graph created in this fashion with the same number of vertices and a similar number of edges to the UUSee topologies in Section 5.3 has 2,500 vertices and $p = 0.017$. This graph is uniformly sufficiently dense that for all values of s greater than about $0.05f$ across a broad range of values for k, our algorithm (correctly) recommends full-graph inference.

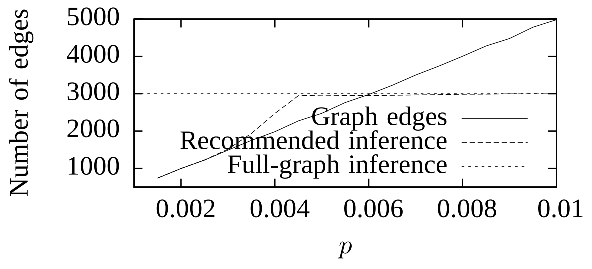

Figure 8. Uniform random graphs: Measurements taken as edge density increases for a graph of 1,000 nodes.

Fig. 8 shows the results of our algorithm on 1,000 node graphs with uniform random edge placement across a range of values of p. The x-axis gives the probability p that any of the possible $n(n-1)/2$ edges appears in the graph, and the y-axis gives the measurement edges (with $k = 3$) for no inference, full-graph inference, and recommended inference. The plot illustrates the transition from edge density less than full-graph inference to density greater than full-graph inference. The value of s is selected, in this example, to be f/k with $f = 50$ and hence $s = 17$. As depicted in the figure, our algorithm correctly recommends mostly direct measurements for graphs with low density. The algorithm rapidly converges toward full-graph inference for graphs with higher density, saving only a few measurements here and there. In between, there is a brief region of over-estimation, where inference is recommended for borderline regions of the graph which do not quite have edge densities meriting inference.[8]

6. Conclusions and Future Work

In this paper, we have studied the network load induced by inference mechanisms, and presented an efficient algorithm to identify subgraphs where replacing direct measurements with inference is most advantageous. Our results show that we achieve significant measurement savings with small-world graphs, which represent popular peer-to-peer and distributed system measurement request patterns. We demonstrate the ability to identify regions of measurement graphs which see no cost benefit from inference, and accordingly use more accurate direct measurements in those regions. We analyze the performance and configuration of our algorithm, and make recommendations for its discretionary parameter s based both on theory and empirical results.

Our future work plans include conducting additional experiments on graphs of other sizes, structures, and clustering properties. We will also investigate the implementation and evaluation of incremental computations, *e.g.*, as discussed in [16]. Finally, we plan to investigate specific inference mechanisms with different linear constants, *e.g.*, the reference-based mechanisms [9], [8], [7], and include their startup costs in our analysis.

References

[1] J. Ledlie, P. Pietzuch, M. Mitzenmacher, and M. Seltzer, "Network coordinates in the wild," in *Proc. of NSDI*, Apr. 2007.

[2] N. Spring, D. Wetherall, and T. Anderson, "ScriptRoute: A public internet measurement facility," in *Proc. of USITS*, 2002.

[3] P. Yalagandula, P. Sharma, S. Banerjee, S. Basu, and S.-J. Lee, "S^3: A scalable sensing service for monitoring large networked systems," in *Proc. of INM*, Sep. 2006.

[4] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An information plane for distributed services," in *Proc. of OSDI*, Nov. 2006, pp. 367–380.

[5] P. Calyam, C. Lee, E. Ekici, M. Haffner, and N. Howes, "Orchestration of network-wide active measurements for supporting distributed computing applications," *IEEE Transactions on Computers*, vol. 56, no. 12, Dec. 2007.

8. Note that the entire plot represents $0.0015 \leq p \leq 0.01$, and hence many of the graphs toward the lower end of the plot are disconnected.

[6] E. Blanton, S. Fahmy, and S. Banerjee, "Resource management in an active measurement service," in *Proc. of the IEEE Global Internet Symposium*, Apr. 2008.

[7] T. S. E. Ng and H. Zhang, "Towards global network positioning," in *Proc. of ACM Workshop on Internet Measurement*, 2001, pp. 25–29.

[8] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *Proc. of SIGCOMM*, 2004, pp. 15–26.

[9] W. Theilmann and K. Rothermel, "Dynamic distance maps of the internet," in *Proc. of INFOCOM*, Mar. 2001.

[10] M. Pias, J. Crowcroft, S. Wilbur, T. Harris, and S. Bhatti, "Lighthouses for scalable distributed location," in *Proc. of Workshop on Peer-to-Peer Systems*, 2003.

[11] Y. Chen, D. Bindel, H. Song, and R. Katz, "An algebraic approach to practical and scalable overlay network monitoring," in *Proc. of SIGCOMM*, Aug. 2004.

[12] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "IDMaps: A global internet host distance service," *IEEE/ACM Transactions on Networking (ToN)*, vol. 9, Oct. 2001.

[13] H. H. Song, L. Qiu, and Y. Zhang, "NetQuest: a flexible framework for large-scale network measurement," in *Proc. of SIGMETRICS*, 2006, pp. 121–132.

[14] I. A. Stewart, "Finding regular subgraphs in both arbitrary and planar graphs," *Discrete Applied Mathematics*, vol. 68, no. 3, pp. 223–235, Jul. 1996.

[15] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, pp. 13–30, Mar. 1963.

[16] J. Holm, K. de Lichtenberg, and M. Thorup, "Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity," *Journal of the ACM*, vol. 48, no. 4, pp. 723–760, Jul. 2001.

[17] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig, "Sparsifi cation–a technique for speeding up dynamic graph algorithms," *Journal of the ACM*, vol. 44, pp. 669–696, Sep. 1997.

[18] G. N. Frederickson, "Ambivalent data structures for dynamic 2-edge-connectivity and k smallest spanning trees," *SIAM Journal on Computing*, vol. 26, pp. 484–538, Apr. 1997.

[19] C. Wu, B. Li, and S. Zhao, "Magellan: Charting large-scale peer-to-peer live streaming topologies," in *Proc. of ICDCS*, 2007.

[20] D. Stutzbach, R. Rejaie, and S. Sen, "Characterizing unstructured overlay topologies in modern P2P fi le-sharing systems," in *Proc. of IMC*, Oct. 2005.

[21] S. Jin and A. Bestavros, "Small-world characteristics of internet topologies and multicast scaling," in *Proc. of IEEE/ACM MASCOTS*, 2003.