On the Cost of Network Inference Mechanisms

Ethan Blanton Sonia Fahmy Greg N. Frederickson Sriharsha Gangam
Department of Computer Science, Purdue University
West Lafayette, Indiana, USA
E-mail: {eblanton, fahmy, gnf, sgangam}@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? We explore three solutions that identify groups of hosts which are likely to benefit from inference. We compare these solutions in terms of effectiveness and algorithmic complexity. Results with synthetic datasets and datasets from a popular peer-to-peer system demonstrate that our techniques accurately identify host subsets that benefit from inference, 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.[1]

Index Terms—Internet measurement, delay inference

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 effort required to collect a large set of measurements, usually 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 [2]. Another example application is the UUSee television streaming service [3].

A network measurement service, which provides measurement results to applications upon request, is uniquely suited to utilize network inference mechanisms. Examples of network measurement services include ScriptRoute [4], the Scalable Sensing Service (S^3) [5], iPlane [6], and the system by Calyam *et al.* [7]. 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 operate a network inference mechanism, and compare this load to that of direct measurement of the requested properties [8].

In this paper, we predict the network traffic injected by inference mechanisms, and use this knowledge to replace direct measurement traffic by inference when the cost of direct measurement exceeds that of inference. After setting up an inference mechanism, continuing measurements typically require $O(n)$ probes to estimate properties of $O(n^2)$ paths [9], [10], [11], [12]. 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. Part of this work (focusing on one of three solutions presented in this paper) appeared in [1].

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.

We present three efficient methods for *identifying opportunities when inference induces less traffic than a given pattern of direct measurements*, allowing for small compromises in measurement accuracy to be traded for a reduction in measurement traffic load. We compare the three methods in terms of effectiveness and algorithmic complexity. Our methods are *not* themselves inference mechanisms, but tools for deploying existing inference mechanisms dynamically on hosts where their use is advantageous.

The remainder of this paper is organized as follows. Section 2 defines the terminology we use throughout the paper. Section 3 defines the problem that we are addressing. Section 4 gives our solutions and their analysis. Section 5 discusses 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 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*, and ultimately returns the results from these tools to the applications. 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 when performing a given measurement. We assume that the cost of contacting a measurement infrastructure is acceptable. For most situations, this cost should be comparable to, or less than, the cost of performing the measurement in question.

We consider inference mechanisms which have a cost linear in the number of hosts participating in the inference, after startup (startup costs may be higher). *Reference-node-based* inference mechanisms utilize measurements to a small set of reference nodes (which may or may not be the same set for each participating host). While inference mechanisms capable of inferring various network properties have been introduced in the literature, the most popular and mature inference mechanisms infer path latency. Our methods are, however, suitable for any inference mechanism whose costs can be approximated as discussed in Section 3.

3 When to Use Inference?

Using an inference mechanism implies a potential sacrifice in the accuracy of measurement results. For example, the Vivaldi and GNP latency inference mechanisms achieve a relative error of 50% or better for 80% and 90% of network paths, respectively [10], [9]. The median error achieved by Vivaldi over all network paths in the same set of experiments was about 11% [10]. Due to the potential error introduced by inference mechanisms, inference may not be an appropriate choice for all measurement scenarios. Determining the suitability of inference for particular applications is outside the scope of this paper. For those applications which can tolerate some error in measurement results, this error can be traded for reduced network load.

Most reference-node-based inference mechanisms 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.*, [10], [11]. Others perform varying numbers of measurements per host, but average a constant number per host, *e.g.*, [9], [12]. 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 in GNP [9]) 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. Initial cost can include all-pairs measurements among a subset of hosts [9], or per-host measurements larger than steady-state operating costs [10].

Assume that we can identify or approximate the constant in an inference mechanism 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 [9], and the authors of the Vivaldi delay inference mechanism find that a selection of 32 neighbors [10]

yields accurate results in their study. For these two systems, under their recommended configurations, k would be 15 and 32, respectively. The cost incurred by each non-landmark GNP host is k measurements, and the cost incurred by each landmark host is kn measurements. Fig. 4 of Ng and Zhang [9] identifies this cost. For Vivaldi, startup costs may be high, but new hosts joining the system need take only a small constant number of measurements to accurately place themselves, as discussed in Section 4.4 of Dabek et. al [10]. 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 number of *requested measurements* exceeds the *number of measurements required to perform inference*. In this case, using 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(s) 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.

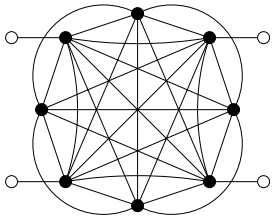

Fig. 1. Graph benefiting from partial-graph inference when $k = 3$.

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 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 $0 \leq |E'| \leq |E|$ (all direct measurements) and $0 \leq |E'| \leq k|V|$ ($k|V|$ is the cost of inference on the entire graph, using one of the inference mechanisms in the literature). The only transformation operation allowed on G is replacement of all edges among subsets V_i of V by $k|Vi|$ edges (*i.e.*, employing an inference mechanism 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 Solutions 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 [13]. 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 approximations. First, in Section 4.1 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. We then explore, in Section 4.2, algorithms which use the degree of individual vertices to select high-degree vertices for inclusion in inference. Finally, we present a hybrid algorithm in Section 4.3 which uses spanning forests to identify likely candidate vertices, and then prunes this set of vertices based on individual vertex degrees.

4.1 The Spanning Forest 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[3] 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 we use. 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.

We first 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:

$$\mathrm{score}(e) = \sum_{i=1}^{f} \begin{cases} 1 & : \quad e \in F_i \\ 0 & : \quad e \notin F_i \end{cases}$$

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). Generation of the weighted graphs and computation of their spanning forests is an easily parallelizable operation, as each graph is treated separately during this stage of the algorithm.

```
def inference_groups_by_forest(G, f, s)
    for i ∈ 1..f do
        let V(G_i) = V(G)
        let E(G_i) = E(G)
        for e ∈ E(G_i) do
            let wt(e) = random()
        let F_i = MSF(G_i)
    for e ∈ E(G) do
        if score(e) > s then
            let E(G) = E(G)\e
    return components(G)
```

Fig. 2. Pseudocode for probabilistic spanning forest selection of nodes for inference.

4.1.1 Bounds on the Expected Score

The motivation for using minimum spanning forests with random edge weights is that we want a quick method for estimating edge-connectivity in a graph.

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

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 [14], $\Pr(S' - \mathrm{E}[S'] \geq t) \leq \exp(-2t^2/f)$. Then

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

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.

4.1.2 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 this 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 [15], or worst-case $O(|V|^{1/2})$ time per operation [16], [17].[4] Efficient handling of on-demand measurements will be a subject of future work.

4.1.3 The Girvan-Newman Algorithm

Our algorithm bears some similarity to the Girvan-Newman algorithm for community identification [18], but there are some important differences. The Girvan-Newman algorithm uses all-pairs shortest paths, rather than minimum spanning

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

forests, and iteratively removes edges using a similar weighting method to our algorithm, in order to partition the graph into communities of vertices within which the edge density is higher than the density between communities. The Girvan-Newman algorithm has a higher algorithmic complexity of O($|E|^2|V|$).

An important distinction between our algorithm and Girvan-Newman lies in the algorithms' respective goals. The Girvan-Newman algorithm places every vertex in a community, based on its connectivity, regardless of the *absolute* density of its connections to that community. This means that even very sparse graphs form communities, and vertices which are sparsely connected to the graph as a whole will be grouped with *some* community. These properties can interact to form legitimately densely-connected communities with a number of sparsely-connected outliers. In contrast, our algorithm aims to select only areas of the graph of high absolute density (that depends on the parameter k), and to exclude vertices of low degree and areas of low overall density entirely. This edge connectivity-based approach stems from the intuition that subgraphs of high absolute edge connectivity are more likely to represent k-regular or denser subgraphs.

```
def inference_groups_by_degree(G, k)
    let V(G') = ∅
    for v ∈ V(G) do
        if degree(v) ≥ k then
            let V(G') = V(G') ∪ v
    for vw ∈ E(G) do
        if v ∈ V(G') and w ∈ V(G') then
            let E(G') = E(G') ∪ vw
    return components(G')
```

Fig. 3. Pseudocode for selection of nodes for inference by node degree.

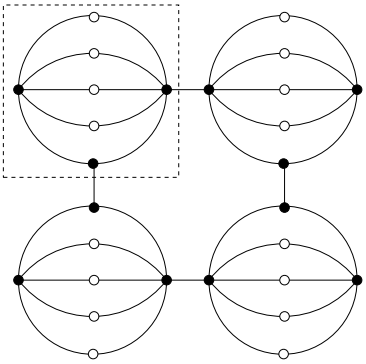

Fig. 4. Highly connected nodes separated by sparsely connected nodes.

4.2 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 some threshold, *e.g.*, k.[5] A pseudocode of this algorithm is given in Fig. 3. We use the same notation as in Section 4.1, with the addition of $\mathrm{degree}(v)$, which returns the degree of a node. The algorithm simply selects high degree nodes and the edges incident upon these nodes as components to use inference on. This algorithm has the apparent benefit that node selection is solely based on information known at the node in question.

While our experiments show that this approach works well on many graphs, there are graphs for which it fails. One graph structure for which this method fails is depicted in Fig. 4. 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 the spanning forest algorithm across two by two and three by three lattices of the structure in the dotted box from Fig. 4 (arranged as in Fig. 4, 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 $s \leq f/k$.

Selection by degree can be used to correctly identify vertices which benefit from inference on the graph in Fig. 4 by simply taking a second pass across the vertices selected as above, and retaining only the selected vertices which have at least k selected neighbors. However, it can be easily shown

5. We use a threshold of k since we are examining the benefit to an individual vertex. We experimented with higher threshold values and found that, as graph density increases, results are not highly sensitive to the specific threshold value within the $[k, 2k]$ range. There is no particular value that was clearly superior for all graphs.

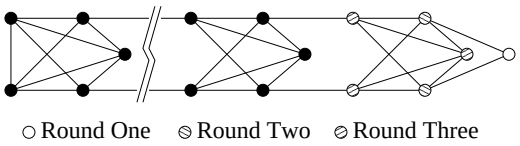

Fig. 5. Graph requiring $2n/k$ iterations of degree-based vertex selection to determine that no inference is warranted for $k = 4$. Vertices eliminated in the first three iterations are marked.

that there exist graphs for which this method is insufficient. These graphs could require as many as $O(n/k)$ iterations to correctly identify those vertices which benefit from inference. In addition, this iteration process complicates distribution of the algorithm, as all rounds of selection after the first require information which is not necessarily known at the node being selected.

An example graph which requires $2n/k$ iterations is shown in Fig. 5. This particular graph is constructed for $k = 4$ and $n = c \times k + 1$ for any $c \geq 1$. Using similar patterns, graphs for any $k \geq 3$ and $n > k + 1$ can be constructed. Each graph is constructed by chaining $\lfloor n/k \rfloor$ components consisting of $k + 1$ vertices and $k(k-1)/2 - 2$ edges each, except the "leftmost" and "rightmost" components. Each of these components forms a fully-connected subgraph less any two edges, with the leftmost component missing only one edge, and the rightmost component consisting of only n mod k fully connected vertices less one edge. These components are then connected together in a chain with two edges connecting each adjacent pair, one edge connecting each of the two endpoints of a "missing" edge from the component on each side.

Fig. 6 presents an algorithm which iterates the process discussed above until no more vertices are removed. Starting with a graph G' that includes all vertices and edges in the original measurement request graph, the outer loop in this algorithm removes those vertices which have degree less than or equal to the threshold (k in this case) in graph G', and all edges incident upon them. This loop terminates when no vertices are removed for one full iteration.

This algorithm results in a set of vertices and edges representing the *k-core* of the input graph [19]. Note that this algorithm does not select vertices having a degree of precisely k for inference;

```
def inference_groups_by_kcoreness(G, k)
    let V(G') = V(G)
    let E(G') = E(G)
    do
        let V_prev = |V(G')|
        for v ∈ V(G') do
            if degree(v) ≤ k then
                let V(G') = V(G')\v
            for w ∈ neighbors(v) do
                let E(G') = E(G')\vw
    while V_prev ≠ |V(G')|

    return components(G')
```

Fig. 6. Pseudocode for selection of nodes by k-coreness.

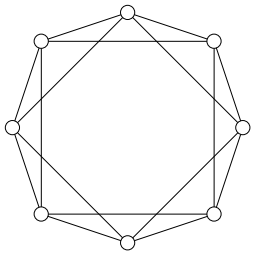

Fig. 7. Graph for which no inference should be performed if $k = 3$, but $\mathrm{inference\_groups\_by\_kcoreness}()$ recommends inference on the entire graph.

it requires that vertices have a degree greater than k.[6] While this algorithm correctly computes the optimal inference groups for a graph such as that described in Fig. 5 (which is to say, it recommends that no inference be performed), it does not correctly compute inference for *all* possible graphs. Fig. 7 depicts a graph where, for $k = 3$, it will recommend inference on the entire graph; however, this yields a total of 24 measurements, whereas the measurement request graph contains only 16 edges.

While the complexity of the algorithm in Fig. 6 is $O(|V| \times |E|)$, the literature gives algorithms which compute k-cores in $O(|E|)$ time [20].

6. Omitting vertices with degree k will not increase the number of measurements taken; their inclusion can cause graph areas of marginal density to be included in inference which should in fact participate in direct measurements.

```
def inference_groups_by_hybrid(G, k)
    let C = inference_groups_by_forest(G, k)
    let V(G') = ∅
    let E(G') = ∅
    let V(G') = ⋃_{C'∈C} V(C')
    let E(G') = ∅
    for vw ∈ E(G)
        if v ∈ V(G') and w ∈ V(G')
            let E(G') = E(G') ∪ vw
    return inference_groups_by_degree(G', k)
```

Fig. 8. Pseudocode for selection of nodes by the hybrid algorithm.

4.3 The Hybrid Algorithm

The final approach we investigate is a hybrid combination of the two algorithms previously discussed. Fig. 8 presents pseudocode for this algorithm. Recall that $\mathrm{inference\_groups\_by\_forest}$ returns the *connected components* of its input graph which are recommended for inference. The hybrid algorithm takes these components and, based on the edges internal to these components in the original input graph, prunes those vertices which are of degree less than k. Defining this algorithm in terms of $\mathrm{inference\_groups\_by\_forest}$ and $\mathrm{inference\_groups\_by\_degree}$, we take the output of $\mathrm{inference\_groups\_by\_forest}$ and build a graph from the components that it outputs. This graph consists of all vertices in those components and all edges from the *input graph* internal to this vertex set. Any vertices having a degree less than the threshold k in this constructed graph are then eliminated.

This single pass on the results provided by $\mathrm{inference\_groups\_by\_forest}$ eliminates vertices which clearly do not benefit from joining an inference group. This pass is taken *after* computing candidate inference groups, rather than before, as the connectivity of individual vertices to other vertices participating in inference is in question, rather than the connectivity to other vertices in the original complete graph. Pruning *before* the computation of candidate inference groups would prune based on the latter property. Pruning both before and after incurs extra cost, and Section 5 will show that the spanning forest algorithm performs well in eliminating vertices with poor global connectivity in the course of its operation, making an early pruning pass unnecessary.

The calculation of vertex degrees and vertex pruning in this algorithm require time $O(|V|+|E|)$, which is bounded above by the computation time for the spanning forests in $\mathrm{inference\_groups\_by\_forest}$ at $O(f \times |E| \log |E|)$, as discussed in Section 4.1.2, so the asymptotic running time remains unchanged. We are currently investigating whether the hybrid algorithm can be computed as efficiently as the spanning forest algorithm in Section 4.1 in the face of dynamic updates.

5 Experimental Evaluation

We experimentally evaluate the algorithms given in Section 4.1, Section 4.2, and Section 4.3 on graphs of various structure. As a baseline, we evaluate our methods 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 [3]. These graphs represent a typical scenario where users select peers or servers with which to communicate based on measured network path properties.

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 algorithms output 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 the spanning forest 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. 9 depicts a synthetic topology having two complete subgraphs of 8 vertices each, connected by a "bridge" consisting of two edges and a separating vertex. 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. 9 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. 9). It is important to note that *inference comes at the cost of reduced accuracy*, so using it when unnecessary is highly undesirable.

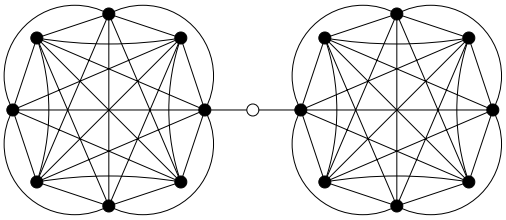

Fig. 9. A two-cluster measurement request topology with optimal kn-cost inference groups for $k = 3$ marked in black.

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 according 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 [3], [21]. 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 spanning forest algorithm identifies the correct subgraphs as candidates for inference. Fig. 10 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 figure illustrates that, as expected, s scales sublinearly with f (at about $0.24f$ for the plotted values of f). For each value of f, we plot the minimum and maximum values of s where correct subgraphs were always identified in our experiments. It can be seen that the values of s are stable, and fall within a narrow range of about $0.02f$.

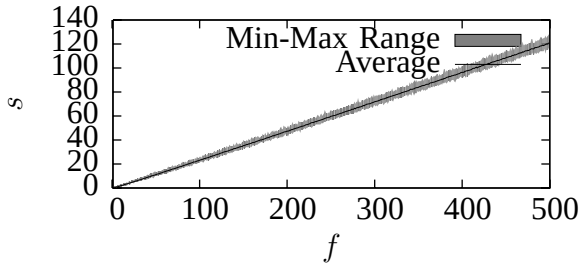

Fig. 10. Number of forests (f) versus the threshold score (s) in the spanning forest algorithm required to identify correct inference components, averaged over 10 runs. Min. and max. s also plotted.

The degree algorithm and k-coreness algorithm identify the optimal solution for this graph (yielding the nodes marked in black), so results for these algorithms are not shown. The curves for the hybrid algorithm corresponding to those in Fig. 10 are identical to the curves for the spanning forest algorithm.

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, the 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 the spanning forest algorithm is still a few milliseconds.

The graph in Fig. 9 is an interesting case study, but presents a trivial case for the spanning forest 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. 9, but with a variable number of bridges between the two fully connected subgraphs. Each bridge is configured analogous to the white node in Fig. 9, having two adjacent edges, one rooted in each of the fully connected subgraphs. No pair of nodes is directly connected by more than

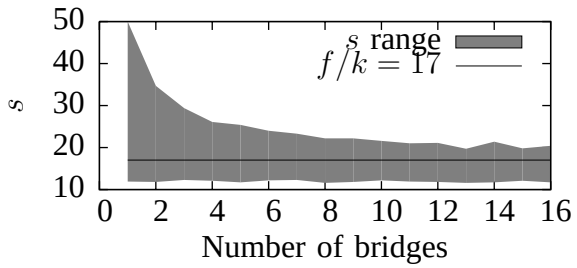

Fig. 11. Number of bridges between fully-connected subgraphs vs. range of s values yielding an optimal recommendation.

one bridge. Fig. 11 illustrates the average values of s bracketing the correct inference recommendation for $f = 50$ and $k = 3$. For reference, $s = 17 = f/k$ is 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 converges 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 (80 versus 96 measurements).

As in the previous example with only one bridge, both the degree algorithm and k-coreness algorithm discard the bridge vertices and arrive at an optimal solution for these graphs. However, we see a divergence in the behavior of the spanning forest and hybrid algorithms. The hybrid algorithm produces a lower curve identical to that in Fig. 11, but due to its elimination of vertices of degree less than k, the upper curve is pushed up to $s = f$ for all values of f. In other words, for small values of s, the hybrid algorithm may fail to select some vertices which would optimally be included in the inference, but the bridge vertex will be discarded even for values of s approaching f as its degree is less than k.

5.2 Single Channel UUSee Topologies

The Magellan project [3] 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 [21].

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 [3].

We generate topologies using the small-world topology generator described by Jin and Bestavros in [22]. We use the node degree distribution information of the UUSee service reported in [3], and set the local preference parameter p to 0.5. Our topologies correspond to viewers of a typical UUSee channel. In [3], 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 smaller topologies, and with graphs representing multiple channels as discussed in Section 5.3.

Fig. 12 depicts the number of measurements required to fulfill the mixture of inference and direct measurement recommended by the spanning forest and k-coreness algorithms. 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

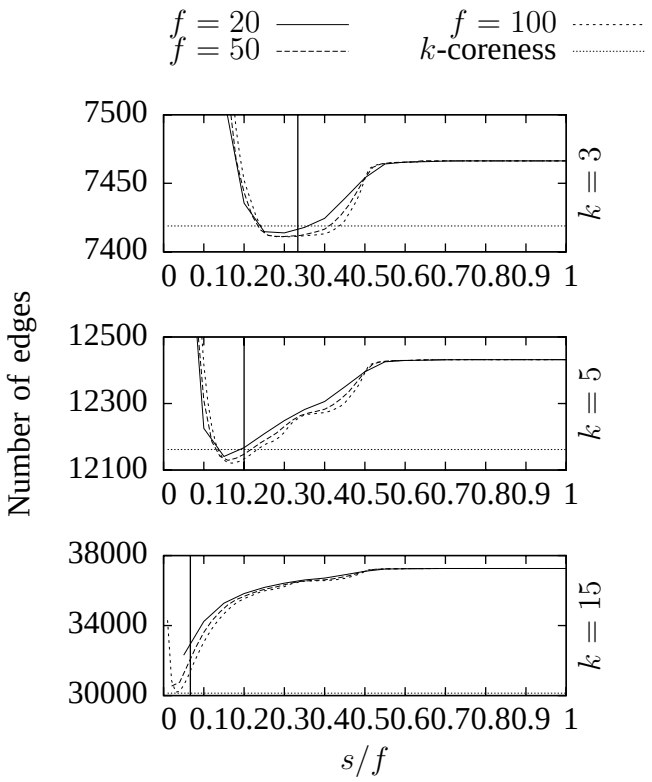

Fig. 12. UUSee topologies: Total measurements taken (edges) using the spanning forest algorithm for three values of k. To compare f and s together, the x axis gives s/f. A vertical line is marked on each plot at $x = 1/k$. The measurements recommended by the k-coreness algorithm for each graph are plotted for comparison.

approaches the 53,000 edges of the original topology. The figure illustrates substantial savings over the original 53,000-edge topology. Savings over performing inference on the entire graph increase as k increases. The number of edges is slightly lower (and the range of s values yielding a lower number of edges broader) with larger values of f, validating the hypothesis that increasing f increases spanning forest algorithm accuracy, at the cost of increased computation time. The k-coreness algorithm does not take parameters f and s, so its recommendation is plotted as a horizontal line.

As seen in the figure, for each value of k, the total number of measurements required when using the recommendation of the spanning forest algorithm 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 4.1.1, suggesting that s must increase with increasing values of f, and decrease with increasing values of k.

Fig. 13 compares the spanning forest, k-coreness, and hybrid algorithms on the same input graphs as the bottom plot of Fig. 12. The minimum values for the spanning forest and k-coreness results are comparable. The hybrid algorithm performs somewhat better than both. The minimum for the hybrid algorithm occurs at a value of s somewhat larger than f/k, whereas the minimum for the spanning forest algorithm is at a value somewhat smaller than f/k. The relationships between these three curves are similar for the other values of k plotted in Fig. 12 (we omit the graphs for brevity), with the exception that the spanning forest algorithm performs somewhat better than k-coreness for small values of k, as previously seen. The relationship between s, f, and k remains similar between the spanning forest and hybrid algorithms for other values of f and k.

The absolute difference in number of measurements required for the three algorithms in Fig. 13 is small for reasonably-tuned values of s. It amounts to about 1.8% of the best figure for all three algorithms, which occurs at $s = 11, f = 100$ for the hybrid algorithm. The primary advantage of the hybrid algorithm over the spanning forest algorithm, therefore, appears to be its relative robustness to the chosen value of s.

In order to investigate the impact of our selection of $p = 0.5$ for topology generation, we examined a number of graphs having values of p varying (in increments of 0.05) from 0.05 to 1.0. We plot the number of edges in the mixed inference and direct measurement schedule recommended by the spanning forest algorithm over an average of ten graphs per value of p. As Fig. 14 illustrates, the value of p (for values other than $p = 1.0$) makes a difference of only 31 edges in the computed topology from least to greatest. The savings increase for a graph with $p = 1.0$, which creates a graph with

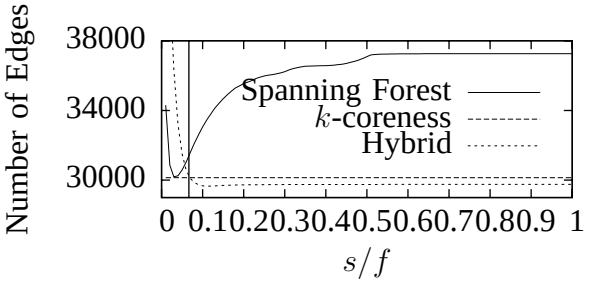

Fig. 13. Average measurements recommended by the spanning forest, k-coreness, and hybrid algorithms on ten 2500-vertex UUSee topologies for $k = 15$. The spanning forest and hybrid algorithms are plotted for $f = 100$. The x axis represents s/f and the vertical line $1/k$, for easy comparison to Fig. 12.

maximum locality and a minimum of "long" edges between neighborhoods.

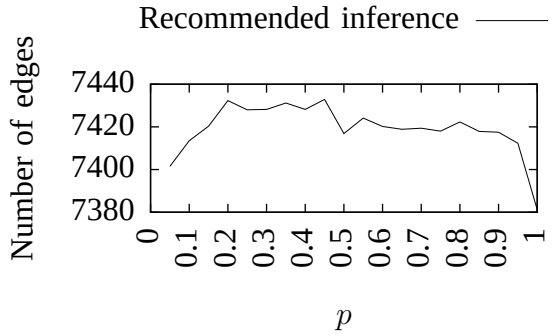

Fig. 14. Spanning forest algorithm performance for various values of the local preference parameter p. Here, k = 3.

Computation time for all algorithms is quite manageable for these topologies. For $f = 20$, executing the spanning forest algorithm on the same 1.8 GHz processor 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 4.1.2, 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.

5.3 Multiple Channel UUSee Topologies

We examine topologies created by joining multiple UUSee channel graphs as described in the previous section. We create ten UUSee channels of between 500 and 3000 hosts each, uniformly distributed, and then join them with twenty edges, as follows: a vertex is selected from a uniform random distribution of all vertices in the ten channels, and then a second vertex is selected from a uniform random distribution of the nine channels of which this vertex is not a member. The two selected vertices are then merged into one vertex having a neighbor set which is the union of the two selected vertices' neighbor sets. The resulting graph is used for our experiments. Note that this resulting graph need not be connected, and in practice often consists of 2 or more connected components.

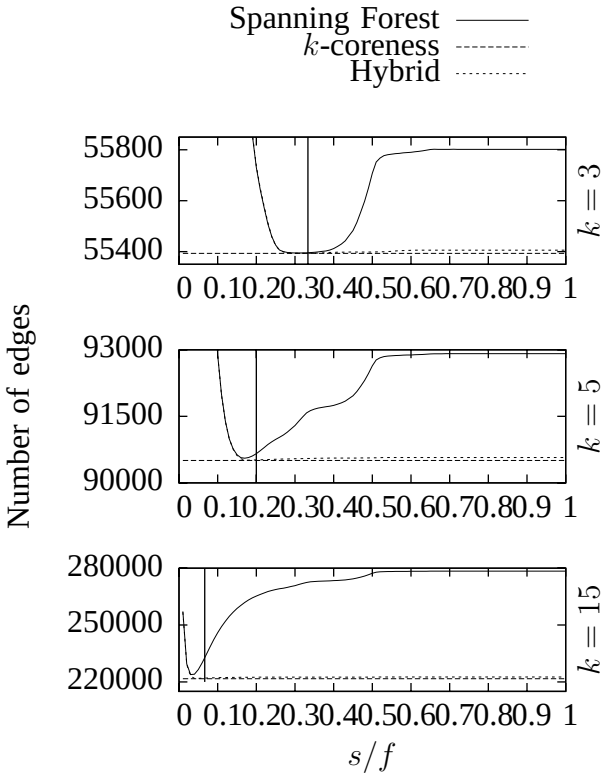

Fig. 15. Multi-channel UUSee topologies: Total measurements taken (edges) for three values of k for the spanning forest, hybrid, and k-coreness algorithms. For consistency with other plots, the x axis gives s/f. A vertical line is marked on each plot at $x = 1/k$.

We execute the spanning forest, k-coreness, and hybrid algorithms on ten such graphs, and present results based on the average of the results from these ten graphs. Due to the increased number of random variables in the creation of these graphs, their metrics vary somewhat more widely than the graphs in Section 5.2. The number of total vertices

in each graph varies from $16,333$ to $20,479$, with an average of $20,479$. The number of edges varies from $323,112$ to $408,319$, with an average of $375,087$.

Fig. 15 shows the results on these multi-channel graphs when s and f are varied. The behavior of the algorithms is consistent with that witnessed in Section 5.2, with the minimum total number of measurements required for the spanning forest algorithm occurring at a value of s somewhat smaller than f/k, and the hybrid algorithm at a value of s somewhat larger than f/k. As in the single channel case, the absolute difference in number of measurements required for the three algorithms is small for reasonable values of s. Once again, for graphs of this density, mixing inference and direct measurements allows us to achieve a total measurement load which is lighter than the total number of direct measurements requested for all three values of k and lighter than performing inference on the entire graph.

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 5.2 has 2,500 vertices and $p = 0.017$. This graph is 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.

Fig. 16 shows the results of the spanning forest, k-coreness, and hybrid algorithms 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 and the recommended inference for each algorithm. The plot illustrates the transition from edge density less than full-graph inference to density greater than full-graph inference. The value of s for the spanning forest and hybrid algorithms is selected, in this example, to be f/k with $f = 50$ and hence $s = 17$. As depicted in the figure, all algorithms correctly recommend mostly direct measurements for graphs with low density. All algorithms also rapidly converge toward full-graph inference (which results in 3,000 edges) 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.[7] This over-estimation is minor for the k-coreness algorithm, and greater for the spanning forest and hybrid algorithms, although the hybrid algorithm over-estimates by a somewhat lower amount than the pure spanning forest algorithm.

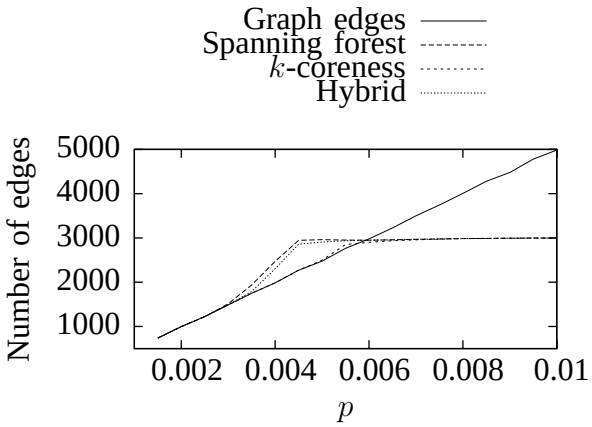

Fig. 16. Uniform random graphs: Measurements taken as edge density increases for a graph of 1,000 nodes.

6 Conclusions and Future Work

We have investigated the network load induced by inference mechanisms, and presented efficient algorithms 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 subgraphs of measurement graphs which see no cost benefit from inference, and accordingly use more accurate direct measurements

7. 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.

in those subgraphs. We analyze the performance of our algorithms, and make recommendations for the discretionary parameter s provided to the spanning forest and hybrid algorithms based on both theory and empirical results.

Our three algorithms, the spanning forest algorithm in Section 4.1, the k-coreness algorithm in Section 4.2, and the hybrid algorithm in Section 4.3, achieve the same goal but have different properties. The k-coreness algorithm can be executed in $O(|E|)$ time, making it asymptotically faster than both the spanning forest and hybrid algorithms, each requiring $O(|E| \log |E|)$ computation. However, for small values of k, the latter two algorithms outperform k-coreness, and for large values of k, the hybrid algorithm outperforms both k-coreness and the spanning forest algorithm. Choosing between the algorithms therefore represents a tradeoff between time and accuracy. In the spanning forest and hybrid algorithms, the discretionary parameter f represents a similar tradeoff, with small values of f requiring far less computational effort, and larger values of f providing not only better results, but increased robustness to the choice of the value of s.

Our future work plans include conducting additional experiments on graphs of other sizes, structures, and dynamics. We will also investigate incremental computations, *e.g.*, as discussed in [15]. Finally, we plan to investigate inference mechanisms with different linear constants, *e.g.*, GNP [9], Vivaldi [10], and DMAPS [11], and include their startup costs in our analysis.

ACKNOWLEDGMENTS

This research is sponsored in part by a gift from Hewlett-Packard, GENI project 1723, and NSF CAREER grant 0238294. We would like to thank Sujata Banerjee, Puneet Sharma, and Praveen Yalagandula (HP Labs) for several helpful discussions on this work. We would also like to thank Chuan Wu and Baochun Li (U of Toronto) for sharing with us their UUSee node degree data.

REFERENCES

[1] E. Blanton, S. Fahmy, and G. N. Frederickson, "On the utility of inference mechanisms," in *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2009.

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

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

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

[5] 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.

[6] 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.

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

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

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

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

[12] 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.

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

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

[15] 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.

[16] 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.

[17] 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.

[18] M. Girvan and M. E. Newman, "Community structure in social and biological networks," in *Proc. of Natl. Acad. Sci.*, vol. 99, 2002, pp. 7821–7826.

[19] S. B. Seidman, "Network structure and minimum degree," *Social Networks*, vol. 5, no. 3, 1983.

[20] V. Batagelj and M. Zaveršnik, "An $o(m)$ algorithm for cores decomposition of networks," in *Proceedings of Recent Trends in Graph Theory, Algebraic Combinatorics, and Graph Algorithms*, Sep. 2001.

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

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