SAND2014-18717C

Strategies for Probing Incomplete Graphs

Sucheta Soundarajan[†] Tina Eliassi-Rad[†] Brian Gallagher[‡] Ali Pinar[*]
[†]Rutgers University [‡]Lawrence Livermore Nat'l Laboratory [§]Sandia Nat'l Laboratories
[†]{s.soundarajan, eliassi}@rutgers.edu [‡]bgallagher@llnl.gov [§]apinar@sandia.gov

Abstract

Given a partially observed (a.k.a. incomplete) graph, which potential nodes or edges should we *actively probe* in order to *accurately* estimate the value of some statistic (such as the size of the largest connected component, LCC) or perform better in some task (such as community detection)? For example, consider a cyber-network administrator who has access to an incomplete graph at time t and needs to have an accurate estimate of the size of the LCC on the complete graph for situational-awareness purposes. She has a limited budget for probing the graph. Of all the nodes in her incomplete graph, which ones should she probe to improve her estimate of the LCC's size? We propose a novel and scalable algorithm, called *AGP* (short for *Active Graph Probing*); and run experiments w.r.t. three statistics/tasks: (*i*) the size of the largest connected component (LCC), (*ii*) PageRank (PR), and (*iii*) community detection (Comms). We consider 9 probing strategies in both batch and incremental forms, various budget scenarios, and apply AGP to a variety of network datasets from different domains. We show that certain strategies (such as probing low degree nodes under an edge-budget scenario to improve the LCC estimate) consistently outperform the other strategies. Additionally, we show that incremental probing far outperforms batch probing; and conduct an analysis of which sampling methods are preferable to others for probing incomplete graphs.

1 Introduction

Suppose that one has an incomplete portion G_{samp} of some larger complete network G_{orig}; and to learn more about the structure of G_{orig}, one can probe nodes from G_{samp}. The problem that we address in this paper is: *Which nodes should be probed to reveal the most useful information about the structure of G_{orig}?* Our work is motivated by problems in cybersecurity and other domains, where one only has a partial observation of the complete network; but for situational awareness purposes, one must get the most accurate and informative picture of the complete network. With a limited query budget on probing nodes or edges, how can this be done?

We consider two basic versions of the aforementioned problem in our approach, called *Active Graph Probing* (*AGP*): one in which we have a fixed budget to probe nodes, and another in which we have a fixed budget to probe edges. In both cases, we start with a sample network G_{samp} and must decide which nodes from G_{samp} to probe in order to learn the most about the structure of G_{orig}. Whenever we probe a node, we learn all of its edges in G_{orig}. This is meant to replicate the situation in which, for example, one learns all of an individual's Twitter followers. In the fixed node-budget version of the problem, we use one unit of budget for each node that is probed. In the fixed edge-budget version of the problem, we use one unit of budget for each new edge that is added to the network.

This is a novel problem that is related to previous work on graph sampling. However, unlike much of the work in graph sampling, we are not attempting to generate a sample from scratch. Instead, we are studying how one can enhance an existing sample, regardless of how that sample was generated.

We present a variety of strategies based on important network features. We evaluate these strategies with respect to various tasks such as finding the nodes in the largest connected component of the network (LCC), identifying the highest PageRank nodes (PR), and detecting community structure (Comms). Beginning with a network sample G_{samp}, we conduct probes according to some strategy, and thus obtain an enhanced sample G'_{samp}. We apply each task to both G'_{samp} and G_{orig}. The success of a probing strategy is then defined by how well the results of the task on G'_{samp} match the results of that task on G_{orig}. Our experiments demonstrate that in almost every case, a simple strategy such as probing based on node degree tends to produce the best outcome.

The **contributions** of our paper are as follows:

- We introduce the problem of determining which k nodes in an incomplete network to probe in order to

*Pinar's work is supported by the DARPA GRAPHS program. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

obtain a network structure that is more accurate with respect to a specific task such as the size of the LCC.

- We present *Active Graph Probing* (*AGP*), an general approach for addressing the above problem. We describe 9 probing strategies covering both batch and incremental probing, plus both fixed node- and edge-budget scenarios.
- We perform an extensive set of experiments spanning 7 network datasets, 3 tasks, and 5 sampling methods (which generate the initial incomplete graph).
- We show that for the problem of identifying nodes in the LCC, if one is given a fixed node budget, probing high degree nodes is the best strategy. If one is given a fixed edge budget, probing low degree nodes is the best strategy.
- We show that for the problem of identifying the highest PageRank nodes, probing high degree nodes is the best strategy regardless of whether one has a fixed node or fixed edge budget.
- We show that for the problem of community detection, probing random nodes from throughout the entire network is the best strategy regardless of whether one has a fixed node or fixed edge budget.

We describe *AGP* next. Sections 3 and 4 present our experiments and a discussion of our results, respectively. In Section 5, we discuss related work; and conclude the paper in Section 6.

2 Proposed Method: Active Graph Probing

Given a graph G_{samp} that is an incomplete (possibly sampled) graph of an unknown larger complete graph G_{orig}, our approach *AGP* selects nodes from G_{samp} for probing. We assume that when a node is probed, we learn all neighbors of that node (e.g., by querying Twitter for all followers of an individual). By probing nodes, we add new nodes and edges to G_{samp}, resulting in graph G'_{samp}. We can then run some task on graph G'_{samp} (such as community detection). Our goal is to select probes such that the results of the task on G'_{samp} are as close as possible to the results of the task on G_{orig}. See Figure 1 for an overview of this process. We refer to this process as *Active Graph Probing*, or *AGP*.

We consider two basic versions of the above problem. In the first version, we are given a **fixed node budget** k_v, which can be used for probing k nodes and learning all of their neighbors. In the second version, we are given a **fixed edge budget** k_e, which is again used for probing nodes and learning their neighbors, but we lose a unit of budget for each new edge that is observed. We consider a variety of strategies: simple strategies, which follow a single rule; and combination strategies, which use multiple rules. See Table 1 for a list.

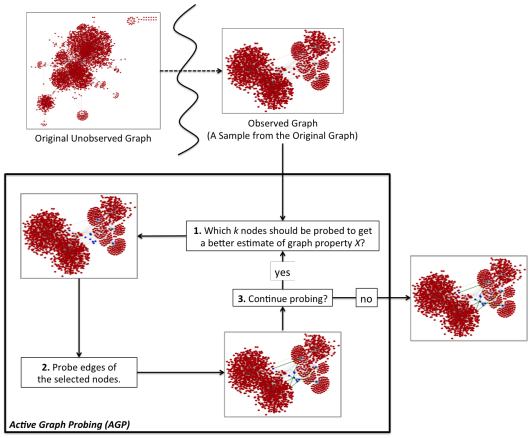

Figure 1: Overview of probing process. One is given a sample of a larger complete graph, then *AGP* suggests which nodes should be probed for more information. After probing, one can terminate or continue.

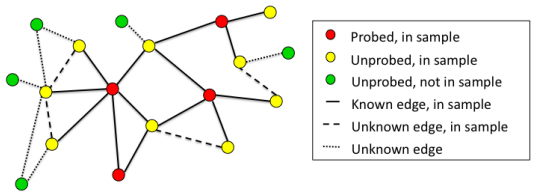

Figure 2: An illustration of a sample graph before probes are conducted. Red nodes have been probed. Yellow nodes are adjacent to red nodes, so they are in the sample but have not been probed (i.e., we do not know their full neighborhood). Green nodes are outside the incomplete graph but are in the complete graph.

Figure 2 depicts what a sample graph might look like. Red nodes have been probed, so we know all of their edges. Yellow nodes are adjacent to red nodes: we know that these nodes exist, but do not have their full neighborhood information. The sample has no knowledge of the green nodes.

2.1 Intuitions Behind Designing Probing Strategies A successful probing strategy selects nodes that, when probed, give structural information about the network that is useful for the task under consideration. Examples of tasks include identifying of LCC in G_{orig}, finding the highest PageRank nodes from G_{orig}, and detecting the community structure of G_{orig}.

Intuitively, for LCC, a probing strategy can be successful if it brings many new nodes into the incomplete network. If a node selected for probing is in the LCC,

	Category	Sub-Category	Strategy Names	Description
Simple Strategies	Exploit	Degree	**HighDeg, LowDeg**	Select the highest or lowest degree nodes.
		Structural Hole	**HighDisp, LowDisp**	Select the highest or lowest dispersion nodes. Dispersion is a measure of how well a node's neighbors are connected to each other. It is an edge-based measure. For each node, we average the dispersion of each of its adjacent edges [2].
			CrossComm	Identify communities using the Louvain modularity algorithm [4], and pick nodes with the highest fraction of neighbors in communities other than their own.
	Explore		**Random**	Randomly select nodes from the sample.
			AllRand	Randomly select nodes from the complete graph. This strategy assumes that a list of all nodes in the graph is available.
Combination Strategies	Explore/ Exploit		**AllRand/HighDeg**	Alternately probe according to both strategies.
			AllRand/LowDeg	

Table 1: AGP's Current Probing Strategies. We group the strategies into 'Explore' and 'Exploit' categories, and further subdivide them into degree-based, structural hole-based, or random. See text for details.

then by probing it we will learn all of its neighbors, which are also in the LCC. In terms of Figure 2, a good probing strategy will select yellow nodes that are adjacent to many green nodes.

For PageRank, if one believes that many of the highest PageRank nodes are already in the sample, getting more information about those nodes is important, regardless of how many new nodes are added to the incomplete network by such probes.

For the Comms tasks, we expect that a balance might be best: while learning about new nodes is important, getting a more accurate picture of the nodes in the incomplete graph is also relevant.

2.2 Simple and Combination Strategies *AGP* considers a variety of *simple strategies* for selecting which nodes to probe. Each of these simple strategies ranks all of the nodes in the incomplete (sampled) graph G_{samp}. At the highest level, these strategies can be grouped into the 'Exploit' and 'Explore' paradigms: 'Exploit' strategies attempt to learn more about important nodes within the sample, while the 'Explore' strategy randomly probes nodes outside of the sample. Within the category of 'Exploit' strategies, we consider sub-categories of strategies based on degree, as well as the general concept of structural holes, or identifying nodes that act as bridges between different parts of the network.

The degree-based methods are based on the intuition that high-degree nodes in G_{samp} are also connected to many nodes outside of G_{samp}. With the structural hole methods, *AGP* targets nodes that "fill" structural holes; and thus, connect different parts of the graph. By probing such nodes, *AGP* learns about many new nodes. In addition to filling structural holes, *AGP* probes nodes by selecting those that are on 'borders' of communities (a.k.a. the CrossComm strategy). The intuition is that with CrossComm we will gain more accurate information about community memberships. We considered a variety of other strategies in the Structural Hole category, including those based on clustering coefficient. We found that their performance was poor, or mimicked that of a listed strategy. We also considered a random-walk based strategy, which performed poorly. For brevity, we have omitted these.

AGP also includes various *combination strategies*, in which it selects two simple strategies and alternately probes nodes according to these strategies. These strategies are based on the 'Explore/Exploit' paradigm.

Note that the 'Explore' AllRand strategy, as well as the combination strategies, require that one has a list of nodes in the complete network G_{orig}.

For both the simple and combination strategies, we consider batch and incremental implementations. In the **batch version**, *AGP* first orders the nodes in G_{samp} based on the selected probing strategy, and queries are made in this order until the budget is exhausted. In this case, the node ordering is not updated as new information is obtained. *AGP* also includes an **incremental version**, in which nodes are probed one at a time, and as new graph information is learned, the ordering is modified. Note that in this case, the same strategy is used for ordering the nodes, but the ordering changes because of the new structural information. Because this method is computationally intensive, *AGP* uses a much faster implementation that performs only one update, halfway through the budget. That is, it selects one probing strategy S, select $b/2$ nodes according to S, and then updates the graph with the new information. It then performs the remaining $b/2$ probes using the same strategy S, but selects these probes using the updated graph.

2.3 Assumptions *AGP* makes two assumptions. (1) it assumes that the complete network is static. (2) it assumes that when one probes a node, one learns all of

its neighbors. This is the only information that can be learned; there is no oracle to answer other questions.

3 Experiments

We present results on a series of experiments in which *AGP* was applied to datasets from a variety of domains. We evaluate the different probing strategies, and conclude with a discussion of sample strategies. We are interested in the following questions: Which probing strategies perform the best? How does the type of budget (node vs. edge) affect the probing strategies? Is incremental probing better than batch probing?

3.1 Experimental Setup We test each of the methods described in the previous section on a variety of datasets. Given a sample graph G_{samp} that represents a portion of some unknown larger graph G_{orig}, we supplement G_{samp} by probing nodes according to strategy S, and learn all of their neighbors. We thus obtain graph G^S_{samp}. We apply a task to G^S_{samp}, with the goal that the results should be as close as possible to the results of the task on G_{orig}.

Our experimental set-up contains three main steps.

1. Given a graph G_{orig}, a sample graph G_{samp} is generated. This is done by performing some number p of sample probes, each of which tells us all of the neighbors of the probed nodes. This sample represents the incomplete graph data that one might have in a real-world task.
2. Next, we apply each probing strategy S to G_{samp}, thus obtaining G^S_{samp}. These strategic probes are at the heart of our method, and should not be confused with the sample probes from the first step.
3. Finally, we run a task on G_{orig} and G^S_{samp}, and compare the results of the task on these graphs. A successful probing strategy produces similar results on both the original graph as well as the probed sample graph. To evaluate S, we compare to the case if we had probed random nodes.

Algorithm 1 contains a more detailed overview of these steps. In the next several sections, we give more details for various stages in this process.

3.1.1 Tasks We consider three tasks. For each task A, we define a quality function $Q_A(G_{orig}, G_{samp})$ that defines how well the task performed on a sample graph relative to how it performed on the original graph.

- PageRank (PR): The goal of the PR task is to identify the highest PageRank nodes in the network [9]. The quality function $Q_{PR}(G_{orig}, G_{samp})$ is calculated by identifying the 1000 highest PageRank nodes in G_{orig} and G_{samp}, and calculating the fraction of nodes that are in both lists.[1]
- Largest Connected Component (LCC): The goal of the LCC task is to identify the nodes in the largest connected component of the graph. The quality function $Q_{LCC}(G_{orig}, G_{samp})$ is defined the Jaccard similarity between the largest connected components of the G_{orig} and G_{samp}.
- Community Detection (Comms): The goal of the Comms task is to identify communities in the graph. We apply the Louvain method for greedy modularity optimization to G^S_{samp} and G_{orig} [4]. This results in two sets of communities, $\mathbf{C}_{samp}$ and $\mathbf{C}_{orig}$. The quality function $Q_{Comms}(G_{orig}, G_{samp})$ is calculated as follows: For each community C in $\mathbf{C}_{samp}$, we find the most similar community D in $\mathbf{C}_{orig}$, as measured by Jaccard similarity. We assign each such C a score corresponding to this Jaccard similarity value. Let S_{samp} be the average of all of these scores over each community C in $\mathbf{C}_{samp}$. We calculate S_{orig} by

Algorithm 1 Overview of our experimental set-up, divided into sample generation, probing, and evaluation. Sample size and budget values are the values that we used, but other values are also possible.

function EXPERIMENTALOVERVIEW(G_{orig}, *App*)
  Create a Sample
  Select a sampling method ∈ {RandNode, RWJ, SBJ-25, SBJ-50, SBJ-75}
  Select a sample size ∈ {1%, 2%, 5%, 10% of nodes in G_{orig}}
  Generate sample G_{samp}

  Select and Conduct Probes
  Select $B \in$ { fixed node budget or fixed edge budget}
  Select *budget* ∈ {1%, 2%, 3%, 4%, 5%, 10%}
  Select a probing strategy S
  Select $A \in$ {batch or incremental probing}
  G^S_{samp} = CONDUCTPROBES(G_{samp}, S, B, *budget*)
  G^R_{samp} = CONDUCTPROBES(G_{samp}, *Random*, B, *budget*)

  Evaluation
  $Qual_{samp} = Q_{App}(G_{orig}, G_{samp})$
  $Qual_S = Q_{App}(G_{orig}, G^S_{samp})$
  $Qual_R = Q_{App}(G_{orig}, G^R_{samp}$
  $Improve_S = Qual_S - Qual_{samp}$
  $Improve_R = Qual_R - Qual_{samp}$
  return $\frac{Improve_S}{Improve_R}$

function CONDUCTPROBES(G_{samp}, S, B, *budget*)
  Probe G_{samp} using strategy S, subject to B and *budget*

[1] For the small network LBL, described later, we use 100 nodes.

performing the same procedure for communities in $\mathbf{C}_{orig}$. The value of $Q_{Comms}(G_{orig}, G_{samp})$ is the harmonic mean of S_{samp} and S_{orig}.

These tasks are roughly ordered from easiest to hardest. Finding the top PageRank nodes is related to finding the highest degree nodes, which does not require detailed structure. Similarly, the largest connected component is typically easy to identify. Community detection, on the other hand, often requires nuance, and so can be much more challenging.

3.1.2 Sampling Methods and Sizes We consider five sampling methods to create graph G_{samp}. In each case, we are given a budget of p sample probes (not to be confused with the strategic probes that will later be performed on G_{samp}). For each sampled node, we learn all of that node's neighbors. Thus, the total number of nodes that exist in the sample graph is much larger than p. We consider values of p equal to 1%, 2%, 5%, and 10% of the total number of nodes in the network. For each sampling method and size, we generate 3 independent samples. The sampling strategies are as follows:

- Random Node Selection: (RandNode) p nodes are randomly selected from G_{orig} for probing.
- Random Walk with Jump (RWJ): We performing a random walk beginning at a randomly selected node. In each step, the random walk transitions to a neighbor (selected with equal probability), or, with 15% chance, jumps to a random node. This process is continued until p different nodes have been visited.
- Snowball Sampling with Jump (SBJ- 25, SBJ-50, SBG-75): These models are similar to breadth-first-search sampling, in which all of a node's neighbors are added. However, instead of adding all neighbors of the current node, we add k fraction of its neighbors, where k is in $[0.25, 0.50, 0.75]$. As with the random walk sampling method, at each step there is a 15% chance of jumping to a random node in the graph.

We are not attempting to downsample a network given full access to the full network, as this would require probing every node to learn its neighbors. This is why we do not consider sampling methods that are known to preserve aspects of network structure.

3.1.3 Budgets Given a sample graph G_{samp}, we consider budget values b that are defined as a fraction f of the total number of nodes (for the case of a fixed node budget) or edges (for the case of a fixed edge budget) in G_{orig}, where f takes values in $[0.01, 0.02, 0.03, 0.04, 0.05, 0.1]$. Given a probing strategy S and a budget b, we probe b nodes (or edges) according to that strategy, and thus obtain G^S_{samp}.

Type	Network	# Nodes	# Edges	Clust. Coef.	# Comps
Communications	Enron	84,429	325,564	0.15	950
	Yahoo	100,000	594,988	0.20	360
	Replies	260,830	308,490	0.004	11,315
	Retweets	39,546	45,796	0.14	3,896
	LBL	3,055	8,769	0.09	9
Likes	Amazon	270,347	741,124	0.40	3840
	Youtube	166,763	1,037,988	0.09	1

Table 2: Statistics for the network datasets that we consider.

3.1.4 Evaluation Methodology We are interested in calculating how well a strategy P performs on network samples generated by a specific sampling method M, with respect to a specific task. To do this, we consider many different samples generated by sampling method M across different network datasets and sample sizes. We also consider several probing budgets.

We first choose a network G_{orig}, and fix a sample graph G_{samp} and a probing budget b. To quantify the success of strategy S, which we call $Eval_A(S, b, G_{samp})$, we compare how well S performs after b probes to the case in which we randomly probed b nodes: how much better is S than random?

Let G^S_{samp} be the graph obtained by performing b probes using strategy S on G_{samp}, and let G^R_{samp} be the graph obtained by performing b random probes on G_{samp} . Let $Qual_{samp} = Q_A(G_{orig}, G_{samp})$, let $Qual_S = Q_A(G_{orig}, G^S_{samp})$, and let $Qual_R = Q_A(G_{orig}, G^R_{samp})$, where Q_A is the quality function from Section 3.1.1. q_{samp}, q_S, and q_R measure how well the unprobed sample, strategically probed sample, and randomly probed sample perform with task A.

The success of strategy S is then defined as $\frac{q_S - q_{samp}}{q_R - q_{samp}}$; that is, we measure how much better strategic probing worked over random probing.

For example, suppose the largest connected component of G_{samp} has a Jaccard similarity of 0.1 with the LCC of G_{orig}. By strategically probing b nodes according to strategy S, we might raise this score to 0.3, but by randomly probing b nods, we might only raise it to 0.2. The strategic probes improved the sample by twice as much as the random scores, so S gets a score of 2.

To aggregate over datasets, sample sizes, and probing budgets, we do the following: Let $AllEval_A(S, M)$ be the set of all $Eval_A(S, b, G_{samp})$ scores on samples that were produced by sampling method M (regardless of sampling size, probing budget, or dataset). We eliminate the top and bottom 20% of scores from this set to reduce the effect of outliers, and then take the average.

3.2 Datasets We consider 7 datasets from two categories: Communications networks and Co-Likes networks, which link two nodes that are commonly liked

Task:PR	Probing Scenario					
Sample Type	Fixed Node Budget			Fixed Edge Budget		
	Batch Probing	Inc. Probing	Regr. Probing	Batch Probing	Inc. Probing	Regr. Probing
RandNode	HighDegree (2.6x)	HighDegree (3.4x)	2.1x	HighDegree (1.6x)	HighDegree (3.4x)	2.7x
RWJ	HighDegree (4.2x)	HighDegree (6.3x)	2.8x	HighDegree (2.4x)	HighDegree (4.4x)	1.8x
SBJ-25	HighDegree (4.7x)	HighDegree (6.6x)	3.2x	HighDegree (2.4x)	HighDegree (4.2x)	1.7x
SBJ-50	HighDegree (4.2x)	HighDegree (5.9x)	2.9x	HighDegree (2.4x)	HighDegree (4.3x)	1.7x
SBJ-75	HighDegree (4.5x)	HighDegree (6.2x)	3.3x	HighDegree (2.4x)	HighDegree (4.3x)	1.7x

Table 3: Best performing probing strategy over different sample types for the PR task, aggregated over all datasets. Values in parentheses indicate how much better the strategy performed over probing nodes at random.

Task:LCC	Probing Scenario					
Sample Type	Fixed Node Budget			Fixed Edge Budget		
	Batch Probing	Inc. Probing	Regr. Probing	Batch Probing	Inc. Probing	Regr. Probing
RandNode	HighDegree (1.6x)	HighDegree (2x)	1.2x	HighDegree (1.3x)	HighDegree (2.9x)	1x
RWJ	HighDegree (2.1x)	HighDegree (3.4x)	1.3x	LowDegree (1.3x)	LowDegree (2.5x)	1x
SBJ-25	HighDegree (2.4x)	HighDegree (3.5x)	1.5x	LowDegree (1.2x)	LowDegree (2.4x)	1x
SBJ-50	HighDegree (2.3x)	HighDegree (3.3x)	1.4x	LowDegree (1.2x)	LowDegree (2.3x)	1x
SBJ-75	HighDegree (2.2x)	HighDegree (3.3x)	1.3x	LowDegree (1.2x)	LowDegree (2.4x)	1x

Table 4: Best performing probing strategy over different sample types for the LCC task, aggregated over all datasets. Values in parentheses indicate how much better the strategy performed over probing nodes at random

by the same people. See Table 2 for statistics.

The communications networks are: the **Enron** email dataset,[2] a **Yahoo** IM messaging network spanning four weeks,[3] a network of Twitter **Replies** spanning one month and a network of Twitter **Retweets** spanning one month,[4] and a network collected by Lawrence Berkeley National Laboratory (**LBL**) of IP-to-IP communications during the span of one hour.[5]

The Co-Likes networks are **Amazon**, in which nodes are books sold on Amazon.com,[6] and two nodes are linked if the corresponding books are frequently purchased together, and **Youtube**, in which the nodes are Youtube videos, which are linked together if they are frequently watched by the same people.[7]

3.3 Analysis of Probing Strategies We perform our evaluation as described in Section 3.1.4, by comparing each strategy S to the Random strategy. Tables 3, 4, and 5 contain the results for the PageRank, LCC, and Comms tasks, respectively. Scores indicate the amount by which S out-performs the Random strategy (scores over 1 mean that S performs better than Random).

3.3.1 PR Results and Analysis For the PageRank task, the High Degree probing strategy is the best strategy for every sampling type and budget scenario that we considered. It performs substantially better than random probing; in some cases, we see almost a five-fold improvement over random probes.

This is expected, because nodes with high PageRank are often those of high degree [5]. Unlike the LCC and Comms tasks, for which even low degree nodes are important, for the PR task it is most important to learn the area around the most central nodes. Because these nodes often have many neighbors, the HighDegree strategy is the most successful.

3.3.2 LCC Results and Analysis For the LCC task, when one has a fixed node budget, the clear winning strategy is to probe high degree nodes. However, for the fixed edge budget case, one is generally better off probing low degree nodes.

Why is the High Degree strategy successful with a fixed node budget? Do new edges connect to new nodes, or join two existing components? We do the following: For each sample, we probe according to the High Degree strategy. For each new edge that is added to the sample, if that edge has at least one endpoint in the existing LCC, we calculate whether the edge (1) is entirely within the existing LCC, (2) connects to a new node outside the sample, or (3) connects two components.

Overwhelmingly, most edges are either within the existing LCC or connect to a single new out-of-sample node. Very few edges connect existing components, and most such components are a single edge. On a Yahoo random node sample, using a probing budget of 10%, we see that an average of approximately 300,000 edges are added within the LCC, 50,000 edges are added from the LCC to new nodes, and fewer than 500 edges connect existing components. Of these latter edges, almost one-third are joining a component of size 2 to the LCC.

[2] http://www.cs.cmu.edu/~./enron/
[3] http://webscope.sandbox.yahoo.com
[4] http://www.twitter.com
[5] http://www.icir.org/enterprise-tracing/download.html
[6] http://snap.stanford.edu/data/amazon-meta.html
[7] http://netsg.cs.sfu.ca/youtubedata/

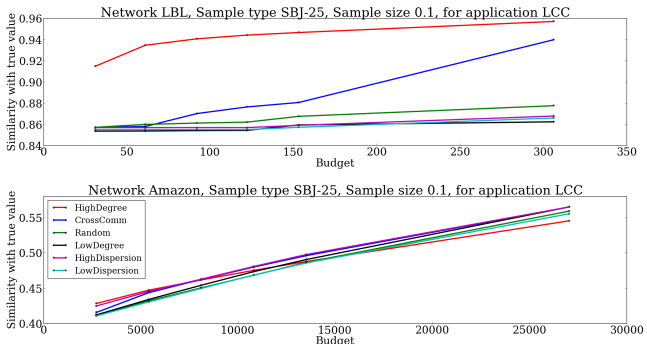

Figure 3: Results of strategies on LBL and Amazon SBJ-25 samples for the LCC task. For LBL, the High Degree strategy is successful. For Amazon, all strategies are similar.

Do high degree nodes give us important structural information, or are they simply adjacent to many nodes? To answer this, we first probe the b highest degree nodes and calculate the number of new nodes that are brought into the network through these probes. Call this value N_{new}. We then revert to the sample before probing, and probe as many *low degree* nodes as are required to bring N_{new} new nodes into the network.

These two strategies perform almost identically, so probing high degree nodes is successful simply it brings in many new nodes to the network.

Why is the Low Degree probing strategy successful for the fixed edge budget case? With a fixed edge budget, we still want to add new nodes into the network. Because we use one unit of budget for every new edge, we do not want to add many edges in the existing LCC, because such edges add no new information.

We conduct probes according to both the High Degree and Low Degree strategies on a fixed sample. For each case, we calculate the number of edges that are contained entirely within the LCC, as well as the number of edges outgoing from the LCC, and take the ratio of these two values. For High Degree probes, this ratio is much higher than for Low Degree probes, indicating that when we probe high degree nodes, more resulting edges are contained within the LCC, thus adding no new information for this task.[8]

For example, consider an Enron sample generated using the SBJ-25 method, with a sample size of 10%, and an edge budget of 15,000. When we probe high degree nodes, barely 700 of the 15,000 edges are outgoing from the LCC to new nodes. If we probe low degree nodes, nearly 6,000 edges are outgoing from the LCC.

[8]These results hold for RWJ and SBJ samples; to understand the random node sampling results, see Section 4.

3.3.3 Comms Results and Analysis For almost all cases, AllRand is by far the best probing strategy for the Comms task. However, this strategy requires that the user have a list of all nodes that exist in the complete network. Thus, it is also important to identify the best probing strategy that does not require this sort of global access. For the fixed node budget case, the best probing strategy besides AllRand is HighDegree, whereas for the fixed edge budget case, the best probing strategy besides AllRand is LowDegree.

Of the tasks that we considered, community detection is the one for which the structure of the entire network is the most important, because even small components should be placed into the correct community. The AllRand strategy allows one to gain information about unexplored parts of the network. Getting information about these parts is much more valuable than fleshing out partially-explored sections of the network, which is what the other strategies do.

If AllRand cannot be used because only nodes in the sample can be accessed, then the most successful strategies are the same as for LCC, and for the same reasons they were successful for the LCC task: these strategies result in the greatest number of new nodes added to the network.

3.3.4 Performance of Other Methods We considered probing strategies based on the intuition that probing nodes that fill structural holes could add previously-unseen parts of the network. For PageRank and LCC, with a fixed node budget, these strategies (High Dispersion and CrossComm) perform substantially better than random (though not as well as High Degree probing). With the SBJ-25 sampling method and the LCC task, the High Dispersion and CrossComm strategies produce a 40% improvement over random, while the High Degree strategy gives over a 100% improvement.

These methods do not perform as well as High Degree probing because they select many low degree nodes. Consider the CrossComm strategy, for instance, which looks for nodes with many neighbors in communities other than the node's own. A node with degree 2 that has its two neighbors in different communities would have half of its edges outgoing from its own community, and so might be selected by this strategy. In future work, we are looking at improving these methods to remove such low-degree nodes.

3.3.5 Batch vs. Incremental Probing We see large improvements by performing probes incrementally. For example, for RandNode samples and the LCC task, with a fixed edge budget, batch probing using the High-Degree strategy results in a 1.3x improvement over random, but incremental probing using the HighDegree

Task:Comms	Probing Scenario					
Sample Type	Fixed Node Budget			Fixed Edge Budget		
	Batch Probing	Inc. Probing	Regr. Probing	Batch Probing	Inc. Probing	Regr. Probing
RandNode	AllRand/HighDeg (1.5x) HighDegree (1.1x)	HighDegree (1.4x)	0.7x	AllRand (3.6x) LowDegree (1.4x)	AllRand (3.6x) LowDegree (2.6x)	0.5x
RWJ	AllRand (20x) HighDegree (1.2x)	AllRand (20x) HighDegree (2.1x)	0.7x	AllRand (30x) LowDegree (1.6x)	AllRand (30x) LowDegree (3.6x)	1.1x
SBJ-25	AllRand (20x) HighDegree (2.2x)	AllRand (20x) HighDegree (2.9x)	1.2x	AllRand (29x) LowDegree (1.7x)	AllRand (29x) LowDegree (4.1x)	1.2x
SBJ-50	AllRand (21x) HighDegree (1.5x)	AllRand (21x) HighDegree (2.5x)	0.9x	AllRand (32x) LowDegree (1.7x)	AllRand (32x) LowDegree (1.7x)	1.1x
SBJ-75	AllRand (24x) HighDegree (1.5x)	AllRand (24x) HighDegree (2.3x)	0.9x	AllRand (35x) LowDegree (1.7x)	AllRand (35x) LowDegree (3.7x)	1.1x

Table 5: Best performing probing strategy over different sample types for the Comms task, aggregated over all datasets. Values in parentheses indicate how much better the listed strategy performed over probing nodes at random. The AllRand strategy is only applicable if we assume that the user is able to probe any node in the original network, rather than just nodes that are already in the sample. Because this is unrealistic under certain task scenarios, we also list the best probing strategy that does not make this assumption.

strategy results in a 2.9x improvement.

3.3.6 Variation Across Networks Tables 4 and 5 list results aggregated over all datasets. Are they true for every network? Across the different networks, we typically see one of two common patterns, which are illustrated in Figure 3. This figure shows the results of various probing strategies at different budget levels for the LCC task on the LBL and Amazon networks, using the SBJ-25 sampling method with a sampling budget of 0.1. On the LBL network, there are clear differences between the different probing strategies. On the Amazon network, there are no meaningful differences between the strategies.

We see these two patterns repeated across different sampling methods and datasets. In general across networks, either the strategies listed in Tables 4-5 are clearly dominant, or all strategies perform similarly.

3.4 Guidance on Selecting a Sampling Methods Suppose when obtaining the incomplete graph, one has control on which sampling method to choose. We found that random walk and snowball sampling often produce high quality samples, where quality is measured both in terms of the number of nodes in the sample (recall that when *AGP* probes a node, it learns all of its neighbors) as well as the performance on the different tasks. An extensive discussion of this matter is available in our supplemental file.

3.5 Running Times The running time of a probing strategy is dependent entirely on the amount of time required to calculate the relevant statistic for each node. We conducted our experiments on a PC with a 3.4 GHz processor and 32GB of memory. On Enron RWJ samples that contain 10% of the nodes from the original network, probing nodes based on degree (either high or low) takes $\approx$ 1 second, which is similar to randomly probing. The CrossComm strategy takes an average of 17 seconds, and the dispersion-based strategies take over an hour. Similar patterns hold in general.

4 Discussion

Our experiments demonstrated the following results: **(1)** For the LCC application, with a fixed node budget, probing high degree nodes is the best, but with a fixed edge budget, probing low degree nodes is better. This is because high-degree nodes have more neighbors outside the sample, but a higher fraction of neighbors inside the sample. **(2)** For the PR application, probing high degree nodes is the best strategy. This is because high degree nodes often have high PageRank, so learning more about these nodes is the most useful. **(3)** For the Comms application, randomly sampling from the complete network is the best strategy, because knowledge of the entire network is important. **(4)** Incremental probing−in which one performs some probes, updates the sample, and then performs the remaining probes−far outperforms batch probing.

Why doe *AGP* work? Theoretical ties Why does picking highest degree nodes to estimate LCC after random node sampling work best? Our supplemental material provides a detailed discussion on this question; here we summarize. Since the sample was generated by randomly selecting nodes, the true degree of a node can be estimated based its degree in the sample. We rely on empirical results [12], which show that clustering coefficients of vertices decrease with increasing degree and they are near zero, for the highest degree vertices.

Based these two observations, higher degree in the sample leads expected higher degree in the entire graph, and due to the likely low clustering coefficients, most of these edges will reach out to new vertices as opposed to closing wedges in the sample. Thus, probing higher degree nodes is expected to add more vertices to the

graph in the fixed node-budget scenario.

For the fixed edge-budget case, what makes a node better is the *ratio* of edges reaching out to new nodes, as opposed to their *count*. Recall that higher degree vertices tend to have lower clustering coefficients, thus in expectation they will have proportionally more edges outside the sample, which makes high degree nodes again the better choice in the fixed edge budget case.

5 Related Work

Our work is most related to graph sampling and active learning.

Sampling Graphs for Specific Tasks Much attention has been paid to the problem of graph sampling. For instance, Maiya and Berger-Wolf [6] study the problem of online sampling for centrality measures including PageRank. Previous literature has examined the study of community detection from graph samples−e.g., by using minimum spanning trees (MSTs) [14, 15], or by expanding a graph sample [7]. Additionally, the MST can be used to exactly calculate the size of the LCC. The component sizes in the MST are the same as the component sizes in the complete network. In [8], the authors describe a single-pass algorithm for creating a MST from streaming edge data for a weighted network. Avrachenkov *et al.* [1] show how to locate high-degree nodes with a very small number of queries. Our work differs from the main body of sampling literature, because we are concerned with the problem of selecting which nodes from an existing sample one should probe, rather than constructing a sample from scratch. Additionally, unlike most sampling methods in the literature, we are not attempting to down-sample a network to which we have complete access, but use a limited number of probes to improve performance on a specific task.

Active learning. Our problem is also related to active learning. For example, Sheng *et al.* [13] consider the problem of when to get another label for elements in a class. Bilgic *et al.* [3] and Pfeiffer *et al.* [10, 11] studied active learning on networks for classification of nodes.

6 Conclusions

We introduced the problem of determining which nodes in an incomplete network to probe in order to learn the most information about the original complete network. We presented our approach *AGP* with a large variety of probing strategies and scenarios−including the cases of having a fixed node or fixed edge budget, and batch vs. incremental probing.

We showed that for LCC, probing high degree nodes is the best strategy when one is given a fixed node budget and probing low degree nodes is the best strategy when there is a fixed edge budget. For PR, probing high degree nodes is the best strategy regardless of the budget type. For Comms, probing randomly nodes from the entire network is the best strategy. We provided theoretical bases for why these probing strategies work for the selected tasks.

Future work. We are working on the following problems: How many sets of probes should one conduct in an incremental probing scenario? When do the gains from additional probes begin to diminish? Is it better to have a small sample size and a large probing budget, or a larger initial sample and a smaller probing budget? In addition, we are working on a regression-based approach in order to learn to select appropriate probes. Lastly, we are in the process of developing a theoretical basis for other graph-mining tasks and graph-sampling methods.

References

[1] K. Avrachenkov, N. Litvak, L. O. Prokhorenkova, and E. Sayargulova. Quick detection of high-degree entities in large directed networks. In *ICDM*, 2014.

[2] L. Backstrom and J. M. Kleinberg. Romantic partnerships and the dispersion of social ties: a network analysis of relationship status on facebook. In *CSCW*, pages 831–841, 2014.

[3] M. Bilgic, L. Mihalkova, and L. Getoor. Active learning for networked data. In *ICML*, pages 79–86, 2010.

[4] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech. Theor. Exp.*, page 10008, 2008.

[5] S. Fortunato, M. Boguna, A. Flammini, and F. Menczer. *Approximating PageRank from In-Degree*, pages 59–71. Springer-Verlag Berlin, 2007.

[6] A. S. Maiya and T. Berger-Wolf. Online sampling of high centrality individuals in social networks. *Advances in Knowledge Discovery and Data Mining*, pages 91–98, 2010.

[7] A. S. Maiya and T. Berger-Wolf. Sampling community structure. In *WWW*, pages 701–710, 2010.

[8] T. C. O'Connell. A survey of graph algorithms under extended streaming models of computation. *Fundamental Problems in Computing*, pages 455–476, 2009.

[9] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[10] J. J. Pfeiffer III, J. Neville, and P. N. Bennett. Active sampling of networks. In *MLG*, 2012.

[11] J. J. Pfeiffer III, J. Neville, and P. N. Bennett. Active exploration in networks: Using probabilistic relationships for learning and inference. In *CIKM*, 2014.

[12] C. Seshadri, A. Pinar, and T. G. Kolda. Triadic measures on graphs: The power of wedge sampling. In *SDM*, pages 10–18, 2013.

[13] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *KDD*, pages 614–622, 2008.

[14] J. Wu, X. Li, L. Jiao, X. Wang, and B. Sun. Minimum spanning trees for community detection. *Physica A*, 392(9):2265–2277, 2013.

[15] S.-Y. Yun and A. Proutiere. Community detection via random and adaptive sampling. In *COLT*, pages 138–175, 2014.