ϵ-WGX: Adaptive Edge Probing for Enhancing Incomplete Networks

Sucheta Soundarajan
Syracuse University
susounda@syr.edu

Tina Eliassi-Rad
Northeastern University
eliassi@ccs.neu.edu

Brian Gallagher
Lawrence Livermore National Laboratory
bgallagher@llnl.gov

Ali Pinar
Sandia National Laboratories
apinar@sandia.gov

ABSTRACT

No matter how meticulously constructed, network datasets are often partially observed and incomplete. For example, most of the publicly available data from online social networking services (such as Facebook and Twitter) are collected via apps, users who make their accounts public, and/or the resources available to the researcher/practitioner. Such incompleteness can lead to inaccurate findings. We introduce the *Adaptive Edge Probing* problem. Suppose that one has observed a networked phenomenon via some form of sampling and has a budget to enhance the incomplete network by asking for additional information about specific nodes, with the ultimate goal of obtaining the most valuable information about the network as a whole. Which nodes should be further explored? We present ϵ-WGX, a network-based explore-exploit algorithm for identifying which nodes in the incomplete network to probe. Aggregated over multiple datasets and a wide range of probing budgets, we find that ϵ-WGX outperforms other explore-exploit strategies and baseline probing strategies. For example, for the task of adding as many nodes as possible, over incomplete networks observed via four popular sampling methods, at the task of adding as many nodes as possible, ϵ-WGX outperforms the best comparison strategy by 12%-23% on average.

ACM Reference format:
Sucheta Soundarajan, Tina Eliassi-Rad, Brian Gallagher, and Ali Pinar. 2017. ϵ-WGX: Adaptive Edge Probing for Enhancing Incomplete Networks. In *Proceedings of ACM Web Science conference, Troy, NY USA, June 2017 (WebSci'17),* 9 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

Keywords: incomplete networks, adaptive probing, graph exploration

1 INTRODUCTION

Collecting network data is expensive, time-consuming, and labor intensive, so most network analyses are conducted on incomplete, partially observed graphs. For example, many researchers obtain graphs from online data repositories for their experiments or collect a limited amount of data through APIs. However, these graphs are often poor representations of the fully observed graph. Having access to more complete data would lead to more accurate analyses, but data acquisition is costly.

A great deal of work has been devoted to the network sampling problem, but here we consider a different problem: If a researcher or analyst is given an incomplete network dataset (e.g., one that was collected by a different individual for some other purpose), how can that person collect a small amount of additional data to best enhance the incomplete network?

Problem Overview: We define the *Active Edge Probing* (AEP) problem as follows: Suppose one obtains an incomplete network $\hat{G}$ that is part of a larger network G, without knowledge of or control over how $\hat{G}$ was generated or observed. Given some amount of budget to obtain more information, which nodes in $\hat{G}$ should one choose to further explore? The specific model for this exploration depends largely on the data collection process (see the discussion below about Probing Models).

Applications: Consider the common case where an organization has access to data within its boundaries, but limited data outside these boundaries. There are many such examples: for instance, a micro-loan company may obtain each applicant's Facebook account, and use his/her social network to make credit decisions about the applicant. In this case, the company wishes to close wedges in the incomplete network, for example by providing some incentive to external users to obtain their data.[1] Alternatively, consider an Internet mapping organization that wishes to identify as many machines as possible in the network. Incomplete maps of the Internet exist, and additional data may be collected by placing monitoring hardware on specific machines/routers. This is an expensive process, so sampling decisions must be made judiciously.

Probing Models: We have observed that the ideal probing algorithm varies substantially depending on the probing model, which describes the information that is returned in response to a query. For example, in response to a query on a node, an API might return all of that node's neighbors, the node's most recent communication, or k random neighbors. In this paper, we consider the case in which a query returns a single random edge adjacent to the queried node. For instance, a probe might represent acquiring an additional random retweet for a Twitter user. This retweet may correspond to an edge already observed. For discussion of other models, refer to our earlier node probing work in [17].

[1] Example based on communications with colleagues at http://www.lenddo.com.

Proposed Method, ϵ-WGX: To solve the AEP problem, we present *ϵ-WGX* (Weighted Graph eXplore), an explore-exploit strategy that takes advantage of the networked representation of the data to determine which node to probe. The strength of ϵ-WGX is that it does not require background information about (1) how the incomplete network was generated, (2) the underlying network structure of the fully observed graph G, or (3) which nodes are good choices for a specific goal/reward.

We compare ϵ-WGX to several baseline strategies, including structural and explore-exploit strategies. We conduct experiments on ten real networks. In this work, we consider three probing goals: (1) Adding as many new nodes as possible to the observed network, (2) adding as many triangles as possible to the observed network, and (3) adding as many new nodes of a certain type to the observed network. Aggregated across sampling methods, networks, and reward functions, ϵ-WGX regularly outperforms the baseline strategies and the other explore-exploit algorithms – e.g., by an average of at least 12% - 23% on the task of adding as many nodes as possible to the incomplete network.

At a high-level, our goals might be considered similar to the goals of network sampling and link prediction. ϵ-WGX differs from network sampling methods primarily in that it is able to adapt to a variety of probing goals and network structures, as opposed to standard network sampling algorithms, which are optimized for one particular goal (e.g., maximizing the number of observed nodes). Depending on the probing goal (e.g., adding triangles to the observed network) ϵ-WGX may appear to be similar to link prediction. However, in such cases, link prediction methods are limited to adding edges *within* the initially observed network, while ϵ-WGX attempts to optimize the *entire* observed network with respect to a feature of interest. ϵ-WGX might thus learn that the initial observation is in a triangle-poor region of the network, and moving to a more triangle-dense region of the network would lead to more triangles than adding edges within the initial observation.

ϵ-WGX differs from existing network sampling methods in that it is a general method in which the user specifies the goal of the data collection (e.g., observing new nodes or adding additional triangles to the network). Additionally, rather than sampling a network from scratch, it supplements an existing incomplete network observation.
Contributions: Our contributions are as follows:

- *A new research direction:* We formalize the *Adaptive Edge Probing* (AEP) problem, which is important to a variety of applications in complex network analysis and graph mining.
- *An original algorithm for the explore-exploit formulation:* We propose the ϵ-WGX algorithm, which outperforms other explore-exploit strategies at the AEP problem.
- *A thorough empirical study:* Our experiments on various data sets and objective functions show that ϵ-WGX regularly outperforms other approaches.
- *A theoretical analysis of regret:* We prove that ϵ-WGX has linear regret, meaning that it performs a constant factor worse than optimal. Although this regret is large, we argue that it is necessary, because the problem setting leads to rewards that are rapidly changing.

2 PROBLEM STATEMENT

In the AEP problem, we want to analyze a graph G, which is a fully observed, networked representation of some social or physical phenomena. But we have access to $\hat{G}$, which is a partially observed version of G. A node or edge is 'observed' if it appears in $\hat{G}$. To improve the accuracy of our analysis, we have a budget b, which can be used to obtain additional data to enhance $\hat{G}$. Each unit of budget is equal to probing a selected node in $\hat{G}$. The probe involves going to the source of the data and receiving a random edge adjacent to the selected node in the fully observed graph G. Note that this is not the link prediction problem, as we only specify one node (rather than querying for an edge between two specified nodes), and may obtain an edge leading to a previously-unseen node.

Definition 1. *Adaptive Edge Probing (AEP) Problem.*
We assume that we are given:

- *An incomplete graph $\hat{G} = (\hat{V}, \hat{E})$*
- *Probing budget $b \in \mathbb{N}$*
- *A global reward function $g : H \subset G \to \mathbb{R}$, which measures the quality of an observed network with respect to a specific goal (e.g., if the goal is to observe as many nodes as possible, how many nodes are in H?).*
- *A local reward function $f : (H, e) \to \mathbb{R}$, where H is an observed network and e is an edge. f quantifies the value of edge e in graph H. As discussed in Section 4, the local reward function is related to the global reward function, but does not simply measure the improvement in g after an edge is observed.*

We are allowed to perform b iterations of probing. Let $\hat{G_k}$ be the graph observed after k iterations of probing. In iteration k, a probe consists of selecting a node u from $\hat{G}_{k-1}$. The result of probing a node u is an edge, which is selected uniformly at random from among all of u's neighbors in G, including edges seen before. The goal is to select b nodes for probing such that $g(\hat{G_b})$ is maximized.

As an example, consider the graph depicted in Figure 1. Black nodes have been probed at least once, and so each black node has at least one observed edge incident to it. However, it is not necessarily the case that all edges adjacent to a probed node have been observed. The observed graph additionally contains gray nodes, which have been observed but not yet probed. The underlying graph contains white nodes, which the observer is unaware of. The challenge is to determine which of the observed (black or gray) nodes should be probed next so as to improve the value of the global reward function by the greatest amount.

Challenges. There are two major challenges in designing a successful probing strategy. (1) For a reward function (such as maximizing the number of nodes in the network $\hat{G_b}$), it is not immediately clear how one should select nodes based only on their structural profile in $\hat{G}$. As we will see, the success of such a strategy can vary across datasets, how the data were observed, probing budgets, and certainly by the application of interest. (2) Since a probe can return an already known edge, the probing strategy must determine when to stop probing a node.

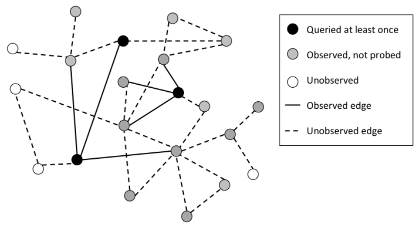


Figure 1: Example of an incomplete graph $\hat{G}$ after some probes have been conducted. Black nodes have been probed at least once, gray nodes have been observed but not yet probed, and white nodes exist in the underlying 'whole' graph, but have not yet been observed.

3 PRELIMINARIES

3.1 Explore-Exploit Algorithms

Explore-exploit algorithms are common in settings where one must choose from among a variety of options, each with a different payoff, without prior knowledge of which option is best. Multi-armed bandit algorithms are a popular class of algorithms within this broad category. The simplest solution to the multi-armed bandit problem is the ϵ-greedy algorithm [20]. In this method, one explores with probability ϵ (i.e., selects an arm uniformly at random), and exploits with probability $1 - \epsilon$ (i.e., selects the arm with the highest average reward so far).

Another popular algorithm is the Upper Confidence Bound (UCB) algorithm [2]. The UCB algorithm selects each arm once, and then calculates an upper confidence bound on the expected reward for each arm, and the arm with the greatest value is chosen.

There are examples of multi-armed bandit strategies on graphs, but these problem settings differ substantially from ours. See Section 7 for details.

3.2 Predicting Population Size with Random Draws

Suppose that we have probed a node k times, and seen w distinct neighbors and r duplicates ($k = w + r$). What is the estimated degree d of that node?

The maximum likelihood estimate (MLE) of d, given k, w, and r, is approximately $\hat{d} = \frac{w+r}{m(s)}$, where $s = \frac{w}{k}$ and $m(s)$ is the solution to $s = (1 - e^{-m})/m$ [16]. We assume that all edges are equally likely to be observed by a probe. For general distributions, one can use Valiants' results [18].

4 PROPOSED METHOD: ϵ-WGX

We view the AEP problem as a version of the general explore-exploit problem: one may probe any node in the incomplete network; and depending on the selection, obtain information of varying value (i.e., reward) for the given goal. For example, if we wish to maximize the number of nodes in the enhanced network $\hat{G_b}$, it would be most valuable to probe nodes in $\hat{G}$ that have a high fraction of neighbors outside of $\hat{G}$. But which nodes are these? One could design heuristic approaches based on graph structure, but we observe that high-reward probes may exhibit significant structural variance depending on the network and the goal for probing.

Alternatively, one could use an explore-exploit approach, which has the following strengths: (1) They can be used *without background knowledge of the network structure or reward function.* (2) They can *adapt* to the network and reward function. Thus, if probing some node produces low reward, the approach is unlikely to probe that node further. (3) They are *stable* and *robust*, regularly providing the best performance for any given network and reward function (while the outcomes of other strategies vary significantly).

4.1 Overview

We propose ϵ-WGX (Weighted Graph eXplore), an explore-exploit algorithm that takes advantage of the graph structure inherent in the AEP problem. We make the following key observation: *When we probe a node and obtain an edge, we have gained information not just about that node, but also about the observed neighbor.* In a typical explore-exploit problem setting, such as advertising, when one selects an arm, one obtains reward data about only that arm. Although some methods have been developed for cases in which arms have linked rewards (e.g., if a user clicks on an advertisement about cars, we might infer a higher reward for other advertisements about cars), we are not aware of algorithms that operate directly on graphs, in which the graph itself is the structure of interest.

ϵ-WGX requires that the user supply reward functions corresponding to the desired probing goal: for example, if the goal is to add as many new nodes as possible to $\hat{G}$, then the reward function could give a score of 1 for a probe that produces a new node, and a score of 0 to other probes. See Section 4.2 for a discussion of reward functions. As a node is probed, ϵ-WGX keeps track of the rewards observed from those probed, as described in Section 4.3. In Section 4.4, we describe how ϵ-WGX uses these observed rewards to update not only the probed node, but also the observed neighbor. After a node is probed repeatedly, ϵ-WGX must consider not just past rewards obtained from that node, but the probability of obtaining new information if it were to be probed again. This process is discussed in Section 4.5.

ϵ-WGX uses a nested exploration structure. In the outer level, it chooses between the two broad options of graph expansion (i.e., observing a broader area of the underlying network) and graph densification (i.e., obtaining more information about the nodes already observed). Once that choice is made, it must choose which node to probe. See Figure 2 for a pictoral overview of ϵ-WGX.

ϵ-WGX has two important characteristics that differentiate it from existing explore-exploit algorithms:

(1) ϵ-WGX uses a two-level nested structure that first chooses between expansion vs. densification, and then selects which node to probe.
(2) With each probe, ϵ-WGX is able to update expected rewards for both the probed node as well as the observed neighbor.

ϵ-WGX could be considered to be a multi-armed bandit algorithm; however, we simply refer to it as an explore-exploit algorithm. This is because, unlike in most multi-armed bandit settings, we cannot make any theoretical statements about the underlying distribution of rewards.

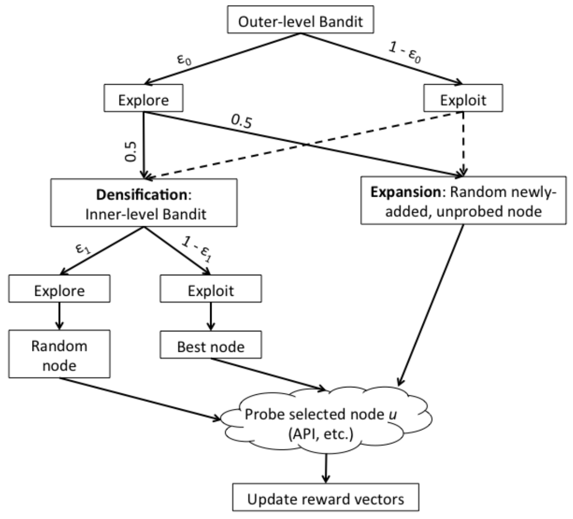

Outer-level Bandit
ε0
1 - ε0
Explore
Exploit
0.5
0.5
Densification: Inner-level Bandit
Expansion: Random newly-added, unprobed node
ε1
1 - ε1
Explore
Exploit
Random node
Best node
Probe selected node u (API, etc.)
Update reward vectors

Figure 2: Flowchart depicting the ϵ-WGX node selection process. The outer level chooses between expansion and densification, and then a specific node is selected. Dashed lines indicate that the choice that has been the best in the past is chosen.

4.2 Global and Local Reward Functions

The user must supply two reward functions to ϵ-WGX: global and local. The global reward function is tied to an application goal, such as maximizing the number of nodes observed, and measures the 'quality' of the observed network as a whole. The local reward function is related to the global reward function, and allows ϵ-WGX to evaluate the value of an individual observed edge. In some cases the local reward function may simply be the change in global reward function. However, there are some reasons that this may not necessarily be the case:

1: Graph Complementarities: The local reward function f must take into account complementarities due to graph structure, or interactions between nodes. Consider the case when two observed nodes have the same unobserved neighbor. If we simply define the reward of an edge as the change in the global reward function g after adding that edge, then the mean rewards of probed nodes would be highly dependent on the random order in which the nodes were probed. For example, suppose observed nodes u and v are both adjacent to node w, and ϵ-WGX probes u first and observes (u, w), and then probes v and observes (v, w). If we simply measured the change in total number of observed nodes, the first probe would receive a reward of 1, and the second would receive a reward of 0. But these rewards are dependent entirely on the random order of probes, which is undesirable. Thus, the local reward function should not just measure the change in the global reward function g.

2: Repeated Observations: If a neighbor is observed multiple times, this should not affect the overall mean reward of that node. Instead, as discussed in Section 4.5, as neighbors are observed multiple times, the observed reward is discounted by the probability of observing a new edge if that node is probed again to obtain the expected reward.

We consider three global reward functions and corresponding local reward functions, described below. In each description, let (u, v) be the edge just observed by a probe (i.e., node u was probed and neighbor v observed).

The **Number of Nodes** global reward function g calculates the number of nodes in $\hat{G_b}$. The corresponding local reward function f has a value of 1 if node v was added to $\hat{G}$ after node u was added to the graph, and a value of 0 otherwise. For example, if both u and v were in the original observed network before any probes were made, then ϵ-WGX calculates a reward of 0. If v was added to the graph after u, or the edge (u, v) is observed from multiple probes on u, then each of these probes has a reward of 1.

The **Number of Triangles** global reward function g calculates the number of triangles in $\hat{G_b}$. The corresponding local reward function f has a value of 0 if v had not been observed prior to the probe, or if edge (u, v) was in the original observed network $\hat{G}$. Otherwise, it has a value of c, where c is the number of neighbors that u and v share c in $\hat{G_k}$, the current observed network (i.e., the number of triangles in which the edge (u, v) participates). Note that using this reward function does not make the problem equivalent to link prediction, as (1) we are only interested in links to nodes that are two steps away, (2) we must consider the probability of obtaining a new edge upon probing, and (3) we are permitted to add new nodes to the network, as opposed to only adding edges between observed nodes.

The **Number of Targeted Nodes** global reward function is like the Number of Nodes reward, except that only nodes with certain attributes (e.g., nodes representing men over 40) produce a reward of 1.

We refer to the value returned by the local reward function as the reward of a probe.

4.3 Reward Vectors and Node Selection

Recall that ϵ-WGX makes two levels of choices: First, the outer level chooses expansion vs. densification, and if densification is chosen, the inner level makes a decision on which node to probe.

Outer Level: Expansion vs. Densification: In each iteration, with probability ϵ_0, ϵ-WGX explores, and chooses between the choices of expansion and densificiation with equal probability. With probability $1 - \epsilon_0$, it chooses the best of these two options as described below.

To make this choice, ϵ-WGX maintains two values, r_d and r_e. r_d measures the average reward observed when it chooses densification. r_e is intended to measure the average reward observed from expansion probes. This includes probes conducted both when ϵ-WGX chooses expansion, or when, during a densification probe, ϵ-WGX selects a node that was not in the original $\hat{G}$ and has not yet been probed.

If expansion is chosen, then ϵ-WGX selects a node from the currently observed network that was not in the original $\hat{G}$ uniformly at random from the set of all such nodes (if no such nodes are available, then ϵ-WGX defaults to densification).

Inner Level: Node Selection: If the outer level chooses densification, then with probability ϵ_1, ϵ-WGX chooses to explore, and chooses a node from the current observed network uniformly at random. With probability $1 - \epsilon_1$, ϵ-WGX chooses the node with

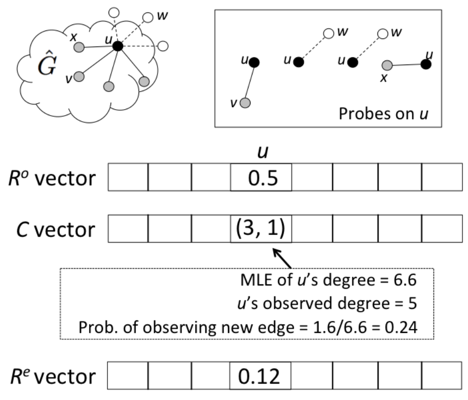


Figure 3: This example depicts rewards for the goal of adding new nodes. Node u is probed four times. The first probe reveals an edge to the known node v, the next two probes produce edges to the new node w, and the fourth probe produces an edge to the known node x. The first and fourth edges produce rewards of 0, and the others produce a reward of 1. R_u^o is thus 0.5. ϵ-WGX estimates u's true degree as 6.6, and the probability of observing a previously-unseen edge upon a probe as $\frac{6.6-5}{6.6} = 0.24$. R_u^o is thus set to $0.5 \times 0.24 = 0.12$.

the greatest R_u^e, the mean reward expected for that node (described below).

To track rewards, ϵ-WGX uses three vectors:

(1) R^o, a vector of mean observed rewards. R_u^o contains the mean reward observed for each node u that has been observed through probing (that is, either u was directly probed or observed in a probe). Section 4.4 explains how R^o is calculated.
(2) C, a vector of counts. C_u contains the pair (A_u, B_u), where A_u is the number of distinct neighbors of u observed when probing u, and B_u is the number of duplicate neighbors of u observed when probing u. Section 4.5 explains how the probability of observing a new edge when u is probed is calculated from these values.
(3) R^e, a vector of expected rewards. R_u^e contains the reward that we expect to observe if u is probed. R_u^e is equal to $R_u^o \times p_u$, where p_u is calculated as described in Section 4.5.

All reward vectors are updated appropriately after each probe is conducted. Figure 3 contains an example of this process.

4.4 Neighbor Updates

The vector R^o contains the mean of a set of observed rewards for each node. R_u^o captures both rewards observed when directly probing node u, and certain rewards obtained when a neighbor v of u is probed and edge (u, v) is observed. Specifically, R_u^o is the mean of the following values: (1) rewards observed when node u is probed and (2) when a different node v is probed and u is observed as a neighbor of v, if u was already in the network at the time of the probe of v, the reward that *would have been* obtained if u had been probed and (u, v) were observed.

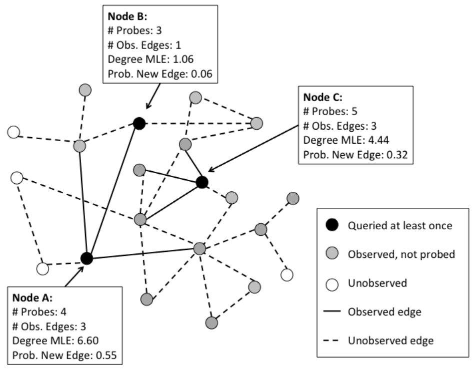


Figure 4: In this example, there are 17 observed nodes (in black and gray). Three have been probed at least once. For each of the black probed nodes, ϵ-WGX calculates the probability of seeing a new edge upon another probing. For the gray unprobed nodes, the probability of seeing a new edge is set to 1. ϵ-WGX expands (select a gray node) or densifies (select from the set of black and gray nodes). If densification is chosen, it may explore, by selecting a random node, or exploit, by selecting the best black node.

4.5 New Edge Probability

For each node u, ϵ-WGX calculates p_u, the probability of seeing a new edge when u is probed. If u has not been probed, or if u has been probed but no duplicate neighbors have been observed, then $p_u = 1$. Otherwise, p_u is calculated as follows: Suppose that for a node u we have conducted C_u probes in total, and have observed A_u distinct neighbors and B_u duplicates. Then ϵ-WGX sets p_u using the MLE method described in Section 3.2.

As an example, consider Figure 4.

4.6 Parameter Settings

The following parameters represent the tradeoff between exploration and exploitation. In general, for the ϵ-Greedy algorithm (which ϵ-WGX is based on), ϵ is set experimentally.

ϵ_0 should be set to a low number, because it governs the choice between only two options (expansion and densification). However, ϵ_0 should not decay to 0 over time, because as the observed graph expands to new regions, the optimal choice may change.

ϵ_1 should be set to a higher number, as it governs the choice between many different nodes. To observe the rewards of such a large number of nodes, frequent exploration is necessary. As before, ϵ_1 should not decay to 0 over time, because new nodes are being added.

Our parameter settings are discussed in Section 5.

4.7 Performance with Respect to Optimal Strategy

Note first that if the underlying network is finite, then ϵ-WGX is trivially a zero-regret strategy. It will eventually fully observe the entire graph, achieving the maximum possible reward.

However, if the total number of probes conducted is small relative to the size of the underlying graph, and if we assume that the reward function is binary- i.e., the reward for a probe can only be 0 or 1, then ϵ-WGX's regret is linear. Here, the regret is measured with respect to the optimal probing strategy, which knows the true expected reward obtained by probing any node. Intuitively, this means that we expect to do some constant factor worse than optimal.

We argue that linear regret is necessary, because as argued above, rapidly changing graph structure leads to rapidly changing rewards, meaning that neither ϵ_0 nor ϵ_1, which control the frequency of exploration, should decay to 0.

Theorem 4.1. *If the probing budget is small relative to the size of the underlying graph, then for binary reward functions, the expected regret of ϵ-WGX is linear.*

Proof: To show that ϵ-WGX has linear regret, we will show that it has regret no worse than linear and no better than linear.

First note that after b probes, the optimal strategy achieves a total reward of at most b, because the maximum reward obtained by a single probe is 1.

Consider the worst possible probing strategy, which selects a node whose edges have all already been observed and probes it repeatedly. The per-probe reward achieved by this strategy is always 0, and so its regret with respect to the optimal strategy is clearly linear. ϵ-WGX cannot do worse than this strategy.

Next, observe that ϵ-WGX cannot achieve regret that is better than linear. ϵ-WGX chooses to explore at least e_0 fraction of the time. Let $\bar{r}$ represent the mean reward obtained by probing a random node, and r^* be the expected reward obtained by following the optimal strategy. Then the total regret of ϵ-WGX after T probes is at least $T(r^* - \bar{r})$, which is a linear function of T.

Thus, ϵ-WGX has linear regret.

5 EXPERIMENTAL SETUP

Our experiments aim to support the following claims:

(1) As motivation for a multi-armed bandit strategy, we show that the best baseline strategy varies across tasks, networks, and even probing budgets. Multi-armed bandit algorithms, including ϵ-WGX, are valuable in cases when a user does not know which types of nodes (e.g., high degree) should be probed.
(2) Because ϵ-WGX takes advantage of the graph nature in the AEP problem, it regularly outperforms both the structural baseline strategies as well as other multi-armed bandit strategies.

We assess the performance of our ϵ-WGX algorithm on ten network datasets using incomplete networks generated/observed by the four popular sampling methods, and compare against relevant baseline probing strategies using three reward functions across a variety of budgets. For each incomplete network, we perform 10 trials of each strategy and report averages and standard deviations. For ϵ-WGX, we set $\epsilon_0 = 0.05$ and $\epsilon_1 = 0.3$.

5.1 Datasets

In our experiments, we probe samples of various types from larger network datasets. However, because obtaining complete network data is very challenging, our larger network datasets (from which we sample) may themselves be samples. To address this issue, we made a specific attempt to identify complete network datasets (KDD, ICDM, and SIGMOD, described below). To increase the number of datasets, we also consider incomplete networks.

We use ten network datasets. **FB-Grad** and **FB-UGrad** represent portions of the Facebook network corresponding to, respectively, graduate and undergraduate students at a university. FB-Grad contains 523 nodes and 3,256 edges, and FB-UGrad contains 1,220 nodes and 43,208 edges [13]. **FB-SocCir** represents a portion of the Facebook network collected by the Social Circles application. It contains 4,039 nodes and 88,234 edges.[2] **Amazon** represents book co-purchases from Amazon.com. It contains 270,347 nodes and 741,124 edges.[2] **Pokec** represents the Pokec social network, and contains 1.6 million nodes and 30 million edges. Nodes are annotated with information such as gender, percentage of profile completed, etc.[2] **Retweet** represents retweets from the Twitter network, and contains 39,546 nodes and 45,796 edges.[2] **Enron** contains e-mail networks from the Enron corporation, and contains 84,429 nodes and 325,564 edges.[3] **KDD, ICDM**, and **SIGMOD** represent coauthorship relationships from the named conferences from 2005-2009. They contain 1366, 1654, and 1365 nodes and 2769, 2808, and 3541 edges, respectively. These networks are complete within the specified dates and conferences.[4]

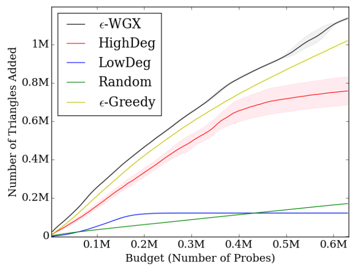
5.2 Incomplete Graphs Obtained by Graph Sampling

We evaluate ϵ-WGX and the baseline algorithms on incomplete networks generated by the following four sampling methods: (1) **Breadth-first search (BFS)** sampling, which is a popular way of exploring portions of the Web [4]. (2) **Random Edge** sampling, which selects a specified number of edges uniformly at random. For example, the Twitter firehose provides a 10% uniform sample over the set of tweets. If retweets constitute edges, then this data produces a sample similar to the Random Edge method. (3) **Random Walk** sampling, which is also a common method for sampling large graphs (see [9]). (4) **Random Walk with Jump** sampling, which is similar to the Random Walk sampling method, except that at each step, there is a specified chance of transitioning to a random node in the graph. In this paper, we used 0.15 as the probability of jumping to a random node.

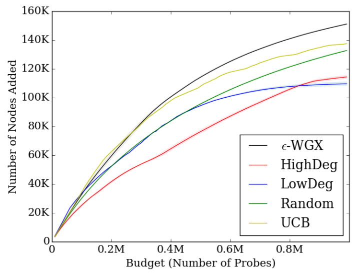
5.3 Competing Probing Strategies

5.3.1 Structural Baseline Probing Strategies. We considered a variety of simple baseline probing strategies based on structural information, and consistently saw that two baselines performed well: probing high degree nodes and probing low degree nodes. In the `High Degree` and `Low Degree` strategies, we assign each node a value proportional (for `High Degree`) or inversely proportional (for `Low Degree`) to its degree. Additionally, we calculate the probability of seeing a new edge when a node is probed multiple times using the same MLE process as ϵ-WGX uses, and multiply the degree-based values by this probability. In each step, the node with the highest value is selected.

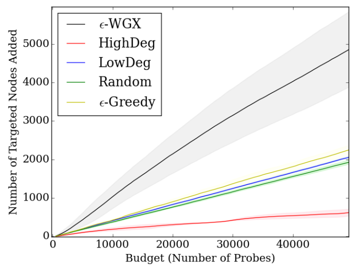
[2]Obtained from http://snap.stanford.edu.
[3]Obtained from https://www.cs.cmu.edu/~./enron/.
[4]Obtained from http://dblp.uni-trier.de/.



(a) Probing an Enron BFS sample, with the goal of adding triangles.



(b) Probing an Amazon RWJ sample, with the goal of adding nodes.



(c) Probing a Pokec RW sample, with the goal of adding nodes representing incomplete male profiles.

Figure 5: Examples of probing results. Shading indicates one standard deviation (not visible when standard deviation is low). Only the better of the two multi-armed bandit algortihms is shown. In all cases shown, ϵ-WGX is the best.

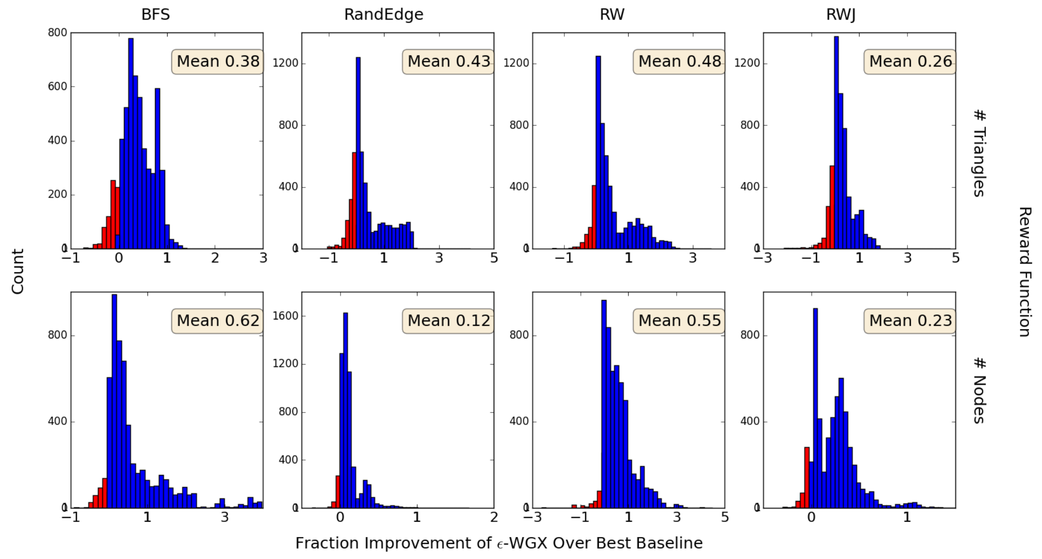


Figure 6: Fraction by which ϵ-WGX outperformed the best degree-based baseline strategy. The top row compares ϵ-WGX to baseline probing strategies with the goal of adding triangles to the observed network, and the bottom row is for the goal of adding new nodes to the observed networks. Results are aggregated over all considered networks. Values are calculated as described in Algorithm 1. The text boxes indicate the means of the distributions. Values in blue are above 0, indicating the fraction by which ϵ-WGX outperformed the baseline method, while values in red are below 0, indicating the fraction by which the baseline method outperformed ϵ-WGX. ϵ-WGX outperforms the best baseline strategy in almost all cases.

In Figure 4, the `High Degree` strategy selects Node A, which has an observed degree of 3 and a probability of seeing a new edge upon probing of 0.55, and thus a total score of 1.65. The `Low Degree` strategy selects one of the gray nodes, each of which has an observed degree of 1 and a probability of seeing a new edge upon probing of 1.0.

We also define a random strategy (`Random`), which randomly selects a node in each step. `Random` removes from consideration those nodes for which it has observed duplicate edges in more than half of their probes.

5.3.2 Other Explore-Exploit Algorithms. We compare ϵ-WGX to the ϵ-greedy and UCB multi-armed bandit algorithms. As with ϵ-WGX, we multiply the values that these methods assign to each node u by p_u (as described in Section 4.5), to ensure that the nodes are not probed even after their edges are all observed.

5.4 Probing Budgets

For each incomplete network $\hat{G}$, we calculate b_{max}, the number of edges adjacent to at least one node in $\hat{G}$ which are not already present in $\hat{G}$). Because one may obtain duplicate edges with multiple probes, it is theoretically possible to probe forever. Thus, we set an upper limit budget at $b^R_{max} = 3 \times b_{max}$, and consider budgets at the 100 quantiles of the interval $(0, b^R_{max})$. Recall that each unit of budget corresponds to probing one node and observing one edge.

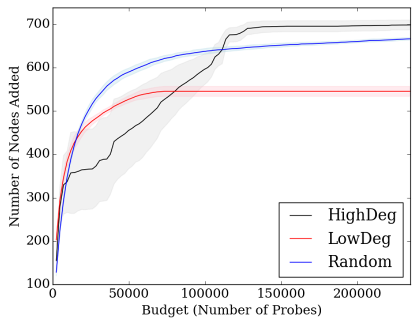


Figure 7: Results of baseline probing strategies on a random edge sample of the FB-SocCir network. Initially, `Random` **is the best, but is soon overtaken by** `High Degree` **probing.**

6 RESULTS

We present results in 3 sections: First, we observe that performance of the three baseline strategies (`High Degree`, `Low Degree`, and `Random`) is highly variable, and so picking a single strategy is difficult (Section 6.1). Second, we compare ϵ-WGX to the baseline strategies, as well as to the ϵ-Greedy and UCB multi-armed bandit strategies (Section 6.2). Third, we consider the Number of Targeted Nodes global reward function on the Pokec network, and again show that ϵ-WGX outperforms the comparison strategies (Section 6.3).

6.1 Analysis of Baseline Strategies

When evaluating the three baseline strategies described in Section 5.3.1, we observe that the best baseline can vary in surprising ways. For example, we consider incomplete networks generated by random walk and random edge sampling on the ICDM network, for the Number of Nodes reward function. For the random edge sample, probing low degree or random nodes is much better than probing high degree nodes. In contrast, on the network generated by a random walk with jump, `High Degree` probing is best.

As shown in Figure 7, on an incomplete network generated by random edge sampling on the Facebook-SocCir network, the best probing strategy varies: on low budgets, `Random` probing is best, but `High Degree` probing is better for higher budgets. This example illustrates the difficulty in selecting a structural baseline strategy, because the best strategy depends on how the incomplete network was generated (which may not always be known), probing budgets and application. These results motivate the use of explore-exploit strategies, which adapt to the network and reward function.

6.2 Analysis of ϵ-WGX

We show that ϵ-WGX outperforms both the best simple structural baseline strategy and the best multi-armed bandit strategy.

Figure 5a shows results on a BFS sample of the Enron email network, with the Number of Triangles reward function, and Figure 5b illustrates various probing strategies on a random walk sample of the Amazon network, with the Number of Nodes reward function. The x-axis represents the number of probes conducted; and the y-axis represents the number of triangles or nodes, respectively, in the enhanced network $\hat{G}'$. Note that in both cases, ϵ-WGX is clearly the best across all considered probing budgets.

Algorithm 1 Aggregate comparisons of ϵ-WGX to baseline strategies for the # of Nodes reward function (can be trivially modified for other reward functions).

```
Given sampling method $M$, baseline strategy $C$
$Dist \leftarrow []$
for Networks $G$, budgets $b$ do
    $\hat{G} \leftarrow$ sample of $G$ produced by method $M$
    $N \leftarrow$ number of nodes in $\hat{G}$
    for Trials 1-10 do
        $\hat{G}_{WGX} \leftarrow \hat{G}$ after $b$ probes by $\epsilon$-WGX
        $\hat{G}_{comp} \leftarrow \hat{G}$ after $b$ probes by strategy $C$
        $N_{WGX} \leftarrow$ number of nodes in $\hat{G}_{WGX}$
        $N_{comp} \leftarrow$ number of nodes in $\hat{G}_{comp}$
        if $N_{WGX} > N_{comp}$ : $v \leftarrow \frac{N_{WGX}}{N_{comp}} - 1$
        else: $v \leftarrow 1 - \frac{N_{comp}}{N_{WGX}}$
        $Dist$.append($v$)
Plot and calculate mean of $Dist$
```

We aggregate results as described in Algorithm 1. We calculate the distribution of the fraction improvement of ϵ-WGX over each comparison strategy over networks, budgets, and trials. Fractions above 0 indicate that ϵ-WGX is performing better than the comparison strategies. For symmetry, we use a different calculation depending on which strategy did better. For instance, if ϵ-WGX adds 120 nodes and the comparison strategy adds 100 nodes, this score will be reported as 0.2, and if the values were reversed, it would be reported as -0.2.

For brevity, we only plot ϵ-WGX in comparison to the best structural baseline strategy (chosen from high degree and low degree probing depending on which does better for a given network, sample type, and reward) in Figure 6. The values in boxes indicate the means of the distributions (values above 0 indicate that ϵ-WGX outperformed the comparison strategy).

On average, in every case, ϵ-WGX outperforms all comparison methods. Table 1 contains the means of these distributions for ϵ-WGX vs. ϵ-Greedy, UCB, and random probing. Again, all means are above 0.[5]

6.3 Identifying Certain Types of Nodes

We now consider the Number of Targeted Nodes global reward function on the Pokec network. Here, we wish to find nodes that represent male users who have filled out less than half of their profile.[6] Interestingly, there is little homophily expressed with respect to this characteristic. We conduct 50,000 probes on the Pokec samples, and show results on a random walk sample in Figure 5c. Similar results were observed for other sampling methods. ϵ-WGX is the clear winner.

6.4 Running Time

The running times of the various multi-armed bandit and baseline strategies are similar. For example, on a BFS sample of the FB-SocCir network, `High Degree` probing takes 4.3e-4 seconds per probe, and ϵ-WGX requires approximately 6.5e-5 seconds per probe.

[5]For additional results, please refer to the version of this paper posted at www.soundarajan.org/EpsilonWGX.pdf, which contains an Appendix with more detailed results.
[6]We saw similar results for other targets.

	Goal: Increase # of Nodes				Goal: Increase # of Triangles			
	BFS	RandEdge	RandWalk	RandWalk/Jump	BFS	RandEdge	RandWalk	RandWalk/Jump
ϵ-Greedy	0.22	0.14	0.54	0.17	0.46	0.68	0.60	0.63
UCB	0.19	0.16	0.23	0.15	2.29	3.47	3.05	3.75
Random	0.63	0.28	0.56	0.32	1.65	1.80	1.42	2.20

Table 1: Fraction by which ϵ-WGX outperformed the comparison strategy, averaged over networks, probing budgets, and trials as described in Pseudocode 1. On average, ϵ-WGX always outperforms the comparison strategy. For example, for the goal of increasing the number of nodes observed, on initial samples produced by a BFS crawl, ϵ-WGX outperforms ϵ-Greedy by 22%.

7 RELATED WORK

Our work is related to the literature on graph sampling, network analysis, and multi-armed bandits.

Graph Crawling and Sampling: Leskovec and Faloutsos provide an excellent overview of several popular sampling methods [9]. Maiya and Berger-Wolf [12] and Wu et al. [21], sample graphs for community detection. Maiya and Berger-Wolf [11] estimate centrality measures; and Cho et al. [5] propose a method for determining which URLs to examine in a web-crawl. Unlike these works, we assume that we improve a given incomplete network, as opposed to having control or knowledge of sample creation.

Network Analysis with Limited Information: Another question is how to infer characteristics of a graph from a sample. Hanneke and Xing [7] attempt to predict topology given access to only a few nodes. Kim and Leskovec [8] attempt to infer missing pieces of a network. Avrachenkov et al. [3] propose a method for locating high-degree nodes in a network using a limited number of queries, and Cohen et al. [6] show how one can efficiently immunize a networked population in which the network structure is unobserved. Macskassy and Provost [10] show how one can identify malicious actors in a network by gathering limited information. Soundarajan, et al. consider the problem of adding as many nodes as possible to an incomplete network [17]. Unlike these works, ϵ-WGX considers arbitrary rewards.

Multi-Armed Bandits: The multi-armed bandit problem was introduced by Robbins [15]. The simple ϵ-Greedy approach [20] works well in practice [19]. In graph settings, Reverdy, et al. [14] and Alon, et al. [1] consider a model in which neighbors have similar rewards. In contrast, ϵ-WGX does not assume that the reward for node u is linked to that of its neighbors.

8 CONCLUSIONS

We presented the *Adaptive Edge Probing* (AEP) problem, in which one selects nodes in an incomplete graph for further exploration, without knowledge of or control over how that incomplete network was observed. We presented the ϵ-WGX algorithm: an explore-exploit method, which identifies nodes that, when probed, produce useful information for a specified goal (e.g., making the incomplete graph more whole by increasing the number of observed nodes). Our results illustrated three main points. (1) Explore-exploit strategies are a useful tool for selecting nodes for probing within the context of the AEP problem, as it is difficult to select a simple baseline strategy that will be successful across reward functions, datasets, and sampling/observation methods. (2) ϵ-WGX consistently performs well across various settings. (3) By using neighbor information, ϵ-WGX outperforms both of the competing multi-armed bandit algorithms, as well as the structural baselines.

REFERENCES

[1] Noga Alon, Nicolo Cesa-Bianchi, Claudio Gentile, Shie Mannor, Yishay Mansour, and Ohad Shamir. 2014. Nonstochastic Multi-armed bandits with graph structured feedback. *CoRR* abs/1409.8428 (2014).

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47, 2 (2002), 235–256.

[3] Konstantin Avrachenkov, Nelly Litvak, Liudmila O. Prokhorenkova, and Eugenia Sayargulova. 2014. Quick detection of high-degree entities in large directed networks. In *ICDM*. 54–65.

[4] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. 2000. Graph structure in the web. In *WWW*. 309–320.

[5] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. 1998. Efficient crawling through URL ordering. In *WWW*. 161–172.

[6] Reuven Cohen, Shlomo Havlin, and Daniel Ben-Avraham. 2003. Efficient immunization strategies for computer networks and populations. *Phys. Rev. Lett.* 91, 24 (2003), 247901.

[7] Steve Hanneke and Eric P. Xing. 2009. Network completing and survey sampling. In *AISTATS*. 209–215.

[8] Myunghwan Kim and Jure Leskovec. 2011. The network completion problem: Inferring missing nodes and edges in networks. In *SDM*. 47–58.

[9] Jure Leskovec and Christos Faloutsos. 2006. Sampling from large graphs. In *KDD*. 631–636.

[10] Sofus A. Macskassy and Foster Provost. 2005. Suspicion scoring based on guilt-by-association, collective inference, and focused data access. In *International Conference on Intelligence Analysis*. 6.

[11] Arun S. Maiya and Tanya Berger-Wolf. 2010. Online sampling of high centrality individuals in social networks. In *PAKDD*. 91–98.

[12] Arun S. Maiya and Tanya Berger-Wolf. 2010. Sampling community structure. In *WWW*. 701–710.

[13] Alan Mislove, Bimal Viswanath, Krishna P. Gummadi, and Peter Druschel. 2010. You are who you know: Inferring user profiles in online social networks. In *WSDM*. 251–300.

[14] Paul Reverdy and Naomi Leonard. 2014. Modeling human decision making in generalized Gaussian multiarmed bandits. *Proc. of the IEEE* 102, 4 (2014), 544–571.

[15] Herbert Robbins. 1952. Some aspects of the sequential design of experiments. *Bull. of the AMS* 58 (1952), 527–535.

[16] Ester Samuel. 1968. Sequential maximum likelihood estimation of the size of a population. *Annals of Mathematical Statistics* 39, 3 (1968), 1057–1068.

[17] Sucheta Soundarajan, Tina Eliassi-Rad, Brian Gallagher, and Ali Pinar. 2016. MaxReach: Reducing network incompleteness through node probes. In *ASONAM*. 152–157.

[18] Gregory Valiant and Paul Valiant. 2011. Estimating the unseen: An n/log(n)-sample estimator for entropy and support size, shown optimal via new CLTs. In *STOC*. 685–694.

[19] Joannès Vermorel and Mehryar Mohri. 2005. Multi-armed bandit algorithms and empirical evaluation. In *ECML*. 437–448.

[20] Christopher J. C. H. Watkins. 1989. *Learning from Delayed Rewards*. Ph.D. Dissertation. King's College.

[21] Jianshe Wu, Xiaoxiao Li, Licheng Jiao, Xiaohua Wang, and Bo Sun. 2013. Minimum spanning trees for community detection. *Physica A* 392, 9 (2013), 2265–2277.