Framework and Algorithms for Network Bucket Testing

Liran Katzir
Yahoo! Labs, Haifa, Israel
lirank@yahoo-inc.com

Edo Liberty
Yahoo! Labs, Haifa, Israel
edo@yahoo-inc.com

Oren Somekh
Yahoo! Labs, Haifa, Israel
orens@yahoo-inc.com

ABSTRACT

Bucket testing, also known as split testing, A/B testing, or 0/1 testing, is a widely used method for evaluating users' satisfaction with new features, products, or services. In order not to expose the whole user base to the new service, the mean user satisfaction rate is estimated by exposing the service only to a few uniformly chosen random users. In a recent work, Backstrom and Kleinberg, defined the notion of *network bucket testing* for social services. In this context, users' interactions are only valid for measurement if some minimal number of their friends are also given the service. The goal is to estimate the mean user satisfaction rate while providing the service to the least number of users. This constraint makes uniform sampling, which is optimal for the traditional case, grossly inefficient.

In this paper we introduce a simple general framework for designing and evaluating sampling techniques for network bucket testing. The framework is constructed in a way that sampling algorithms are only required to generate sets of users to which the service should be provided. Given an algorithm, the framework produces an unbiased user satisfaction rate estimator and a corresponding variance bound for any network and any user satisfaction function. Furthermore, we present several simple sampling algorithms that are evaluated using both synthetic and real social networks. Our experiments corroborate the theoretical results and demonstrate the effectiveness of the proposed framework and algorithms.

Categories and Subject Descriptors

F.2.2 [**Theory of Computing**]: Analysis of Algorithms and Problem Complexity—*Nonnumerical Algorithms and Problems*

General Terms

Algorithms, Experimentation, Performance

Keywords

Social networks, Bucket testing, A/B testing, Network bucket testing, Unbiased estimation

1. INTRODUCTION

To evaluate users' engagement or satisfaction from a new service, feature, or product, providers usually perform bucket testing (also known as split testing, A/B testing, or 0/1 testing). In particular, providers often choose a small subset of users to which a new service is offered. Based on these users' behavior, the overall satisfaction of users can be estimated. If the satisfaction is high enough, the service is released to the entire user population. For example, web page layouts significantly impacts users' engagement. If a new page layout is considered, bucket testing is used to verify that the new layout is better than the existing one. This example also demonstrates why the number of users exposed to the new layout should be considered carefully. On one hand, it should be large enough so that the measurements are statistically valid. On the other hand, it should be as small as possible to limit the potential damage in case the new page layout is 'bad', i.e., to mitigate the number of users who 'suffered' from the experiment in case the new layout is less successful than the existing one. Accordingly, each bucket test is therefore given a budget, $\mathcal{B}$, which represents the maximal number of users it is allowed to effect.

In traditional bucket testing, users' individual interactions depend only on the user and the feature, in which case uniform sampling is optimal. In social networks and services, however, the situation is more complex since users' satisfaction might depend on whether the service is also available to their friends. For example, a messaging service may inherently be very useful but no user can enjoy it if none of his friends use it. Thus, to measure users' engagement the service must be offered to at least some of their friends as well. Other examples include, content tagging, games, certain kinds of ads, event invitations, etc. In [1], the authors suggest to set a parameter $d > 0$ such that users' interactions can be measured only if at least d of their friends have also received the service. We adopt this model as well.

Let the graph $G(V, E)$ represent a social network, where each node corresponds to a user and $\{i, j\} \in E$ iff users i and j are connected (or 'friends'). We denote $|V| = n$ and $|E| = m$. As in [1], we assume, for simplicity, that all the nodes in G have at least d neighbors. Let $f : V \to [0, 1]$ be an arbitrary function over the users measuring the user's engagement or satisfaction. The aim of the test is to estimate the mean value $\mu = \frac{1}{n}\sum_{i\in V} f_i$. For the algorithm to evaluate f on a core set of nodes A, it must pay $|B|$ from its budget $\mathcal{B}$. The set $B \supseteq A$ is the d-closure (or fringe) set of A, which is the minimal set containing A such that all nodes in A have at least d neighbors in B.

Intuitively, the sampling algorithm must balance between two contradicting phenomena. If set A is chosen uniformly at random, the measurement is unbiased but the fringe set B is expected to be roughly d times larger than A. In other words, a large portion of the budget is squandered away on nodes for which the function f cannot be evaluated. This results in a small number of evaluations, which increases the estimator variance. On the other hand, if set A is chosen to contain a heavily connected subset of nodes, the fringe set B is not much larger than A and most of the budget can be used. This, however, requires non uniform sampling, which increases the estimator variance. The challenge is to balance between these effects in order to minimize the variance, while not breaking the budged limitation.

An important difference between our work and [1] is the restrictions put on the satisfaction function. In [1] a random biased coin model generates the function $f : V \to \{0, 1\}$. While this model makes sense and indeed is very useful for analysis, we argue that it is restrictive. The first generalization we propose here is that the function receives real values $f : V \to [0, 1]$. Our rationale is that in many cases, measures of satisfaction take scalar values. These include: the amount of time spent using an application, number of times it was launched, increase in activity or in resulting revenues, etc. Moreover, a random model for f is not always justified. Here, we consider *any* fixed or even adversarially chosen satisfaction function. One can easily verify that the variance of estimating the mean of a random function f is always dominated by the variance obtained (over the random bits of the algorithm) for an adversarial choice of f. Thus, our results hold for any model describing f, including the one used in [1].

Second, the authors of [1] suggest to choose the core set A according to different random walk procedures. While this gives good results, we show that, in fact, there exist a virtually unlimited number of valid algorithms.[1] We argue that *any* distribution over core subsets of nodes can be used. If the probability of any node being in the sample is strictly positive, the obtained estimator Z for μ is also *unbiased*. The only difference between different distributions is the variance this estimator exhibits. Thus, for every network, one should choose the distribution which minimizes the estimator variance. Since one must choose an algorithm before starting the bucket test, we bounds the variance from above for any f.

The rest of this work is organized as follows. In Section 2 we present the meta algorithm which applies the specific algorithms presented in Section 3. Connections of our concept to the random walk approach of [1] is considered in Section 4. A gradient decent optimization procedure aimed to reduce the estimator variance is briefly described in Section 5. Experimental setup and corresponding results are described in Section 6. We discuss the results and conclude the work in Section 7.

2. META ALGORITHM

We start by describing the meta-algorithm, which is identical for all distributions over core sets. The meta algorithm is straightforward and is identical to the one in [1]. Accordingly, a given algorithm produces core sets from some distribution over subsets of graph nodes. Then, for each selected core set it produces an unbiased estimator. It continues to do so until it exhausts the budget $\mathcal{B}$. The output estimator is the mean of all estimators obtained during the process. A formal description of the meta-algorithm is given in Algorithm 1.

Algorithm 1 Network Bucket Testing Meta-Algorithm

```
Input: $\mathcal{B}$, budget
input: $d$, minimal degree threshold
Input: $G(V, E)$, input graph
Input: $Q$, core set distribution over $2^V$
Input: $q$, such that $q(i) \leftarrow \sum_{A \in supp(Q)} Q(A)\mathbb{1}_{\{i \in A\}}$
Output: $Y$, estimation of $\mu = \frac{1}{n}\sum_{i \in V} f(i)$
$b \leftarrow 0$ ; $\ell \leftarrow 1$ ; $L \leftarrow 1$
while True do
  $A_\ell \leftarrow$ drawn according to core set distribution $Q$
  $B_\ell \leftarrow$ $d$-closure (fringe) set of $A$
  $b \leftarrow b + |B|$
  if $b > \mathcal{B}$ then
    break
  end if
  $Z_\ell \leftarrow \sum_{i \in A} \frac{1}{nq(i)} f(i)$ ; $L \leftarrow \ell$ ; $\ell \leftarrow \ell + 1$
end while
return: $Y \leftarrow \frac{1}{L}\sum_{\ell=1}^{L} Z_\ell$
```

The final estimator Y is unbiased since each of the Z estimators are:

$$\begin{aligned}\mathbb{E}(Z) &= \sum_{A \in supp(Q)} Q(A) \sum_{i \in A} \frac{1}{nq(i)} f(i) \\ &= \frac{1}{n}\sum_{i \in V} \frac{f(i)}{q(i)} \left(\sum_{A \in supp(Q)} Q(A)\mathbb{1}_{\{i \in A\}} \right) \\ &= \frac{1}{n}\sum_{i \in V} f_i = \mu \ ,\end{aligned} \tag{1}$$

where Q is the core set distribution over 2^V, and $q(i)$ denotes the probability that node i is in the chosen core set (for more details see Algorithm 1). Note that if $q(i) = 0$, for some i, expression (1) is not valid.

Since the overall estimator Y is unbiased for any distribution, the goal is to reduce its variance. Given the fact that core sets A_ℓ are chosen i.i.d. we have that

$$\mathrm{Var}(Y) = \frac{1}{L^2}\sum_{\ell=1}^{L} \mathrm{Var}(Z_\ell) = \frac{1}{L}\mathrm{Var}(Z) \ .$$

To approximate the size of L, we compute the cost of this experiment which is $\sum_{\ell=1}^{L} |B_\ell|$. Assuming $|B_\ell| \ll \mathcal{B}$ and enforcing that the cost is less than the budget, we get that $L \approx \mathcal{B}/\mathbb{E}(|B_1|)$. Note that since $|B_\ell|$ are also i.i.d. random variables, applying Chernoff's inequality, it is easily shown that the value of $L = (1 \pm o(1))\mathcal{B}/\mathbb{E}(|B_1|)$ with high probability. Finally, we have that

$$\mathrm{Var}(Y) \approx \frac{1}{\mathcal{B}}\mathbb{E}(|B|)\,\mathrm{Var}(Z) \ . \tag{2}$$

where we omit the subscript and use B and Z instead of B_1 and Z_1. Since the budget $\mathcal{B}$ is fixed, the best choice of distribution over core subsets and estimators is the one minimizing $\mathbb{E}(|B|)\,\mathrm{Var}(Z)$. This quantity is highly related

[1]It is worth mentioning that the authors of [1] also consider core sets including single nodes and pairs of nodes.

to the efficiency defined in [1] as $|B|/|A|$. If the engagement function f is itself random, as assumed in [1], we can expect $\mathrm{Var}(Z)$ to be proportional to $1/|A|$.

It remains to compute the variance $\mathrm{Var}(Z) = \mathbb{E}(Z^2) - (\mathbb{E}(Z))^2$. We start by calculating the second moment of the estimate Z,

$$\begin{aligned}\mathbb{E}(Z^2) &= \sum_{A\in 2^V} Q(A) \sum_{i\in A}\sum_{j\in A} \frac{1}{n^2 q(i)q(j)} f(i)f(j)\\ &= \sum_{i\in V}\sum_{j\in V} \frac{1}{n^2}\frac{1}{q(i)q(j)} f(i)f(j) \sum_{A\in 2^V} Q(A)\mathbb{1}_{\{\{i,j\}\subset A_\ell\}}\\ &= \sum_{i\in V}\sum_{j\in V} \frac{1}{n^2}\frac{q(i,j)}{q(i)q(j)} f(i)f(j)\end{aligned} \tag{3}$$

where $q(i,j) \equiv \sum_{A\in 2^V} Q(A)\mathbb{1}_{\{\{i,j\}\subset A_\ell\}}$ is the probability that both nodes i and j are simultaneously included in a core set A. Maximizing this expression over functions f such that $\sum f(i) = \mu n$ gives the worst variance possible. It is easily verified that the maximal obtainable value is $\mathbb{E}(Z^2) = \frac{\mu^2}{q(1)}$, assuming w.l.o.g. that $\max_i \frac{1}{q(i)} = \frac{1}{q(1)}$. This bound, however, is overly pessimistic since it is obtained in the unrealistic case where $f(1) = \mu n$ and all other values are $f(i) = 0$. We therefore need to enforce that the non zero values of f are distributed over many nodes. A natural way to achieve this is to limit ourselves to functions f such that $f(i) \in [0,1]$.

Proposition 1 *Let*

$$W(i,j) = \frac{q(i,j)}{q(i)q(j)} \ ,$$

and

$$U_\mu = \arg\max_{|U|\le \mu n} = \sum_{\{i,j\}\subset U} W(i,j) \ .$$

Then

$$\mathrm{Var}(Z) \le \sum_{\{i,j\}\subset U_\mu} W(i,j)/n^2 - \mu^2 \ . \tag{4}$$

PROOF. First, note that W is a positive semidefinite matrix. This is because $f^T W f = n^2\mathbb{E}(Z^2) \ge 0$ for all f. Therefore, $f^T W f$ is a convex function of f defined over the convex set $f(i) \in [0,1]$ and $\sum f(i) = \mu n$. The maximal value of such functions is obtained in an extreme point of the body. Let u_μ be this extreme point $u_\mu(i) \in \{0,1\}$ and $\sum_i u_\mu(i) = \mu n$. That is, for μn nodes we have $u_\mu(i) = 1$ and for $(1-\mu)n$ nodes $u_\mu(i) = 0$. Setting $U_\mu = \{i | u_\mu(i) = 1\}$ completes the claim. □

Computing the RHS of (4) amounts to finding the heaviest submetrix of size μn of W. The heaviest subgraph problem is notoriously hard [4][2]. It does, however admit scalable approximation algorithms that work well in practice [3][7]. Regardless, in our scenario, it is natural to assume that μ is at least a small constant. Therefore, a random choice of U_μ is expected to yields a μ^2 approximation factor to the optimal. Moreover, if any algorithms improves on the random choice by a factor of t then the solution is guaranteed to by a $t\mu^2$ approximation to the optimal.[2]

[2]This discussion, goes beyond the scope of this paper.

For the sake of simplicity, we use a more relaxed bound which uses the spectral norm of W. Applying the Cauchy-Schwarz inequality yields $\mathrm{Var}(Z) \le \frac{1}{n^2}\lambda_1(W)\|u_\mu\|^2 - \mu^2$. Here $\lambda_1(W)$ denotes the spectral norm of W (its largest eigenvalue). Substituting $\|u_\mu\|^2 = \mu n$ we get that

$$\mathrm{Var}(Z) \le \left(\frac{1}{n}\lambda_1(W) - \mu\right)\mu \ . \tag{5}$$

The largest eigenvalue $\lambda_1(W)$ can be easily calculated using various methods (e.g., the *power iteration* method). In addition for this bound (expression (5)) being significantly easier to compute, we will see in Section 6.7 that it is also tight enough to provide valuable information.

As seen above, the variance of the estimator Y is proportional to $\mathbb{E}(|B|)\mathrm{Var}(Z)$, where both $\mathbb{E}(|B|)$ and $\mathrm{Var}(Z)$ are complex functions of the core set distribution Q. In the next section we describe specific algorithms for efficiently drawing core subsets from a distribution Q and producing the probability vector q for every graph. Our goal here is to reduce $\mathbb{E}(|B|)\mathrm{Var}(Z)$ as much as possible.

3. SPECIFIC ALGORITHMS

In order to describe the algorithm we require some additional notations. Let $N_i = \{j \in V : (i,j) \in E\}$ indicate the set of neighbors of node i and $N_i^+ = \{\{i\} \cup N_i\}$. We denote by $N_{i,j} = N_i \cap N_j$ (similarly $N_{i,j}^+ = N_i^+ \cap N_j^+$) and $M(i,j) = \min\{|N_{i,j}|, d-1\}$. We also denote Q the distribution over subsets of nodes and $Q(A)$ the probability of core set A being chosen. Finally, let $supp(Q)$ be the support of Q, i.e., $A \in supp(Q)$ iff $Q(A) > 0$.

3.1 Naïve Algorithm

Here the core sets are simply the nodes of the graph, $supp(Q) = \{\{i\} | i \in V\}$. In addition, the core set distribution Q is $Pr(A) = p(i)$, where p is some distribution defined over the nodes of G. Hence, $q(i) = Pr(i \in A) = p(i)$. Since each core set contains only one node and we randomly pick d of its neighbors to form the closure set, we clearly have $|B| \le d+1$.[3] To compute the variance we note that $W(i,j) = \frac{1}{q(i)}$ for $i = j$ and zero otherwise. Since W is a diagonal matrix in this case, its top eigenvalue equals its maximal diagonal entry, we have that the spectral bound (5) reduces to

$$\mathrm{Var}(Z) \le \left(\frac{1}{n}\frac{1}{\min_i p(i)} - \mu\right)\mu.$$

This is minimized using the uniform distribution $p(i) = 1/n$ and gives $\mathrm{Var}(Z) \le (1-\mu)\mu$. In this case, it turns out, that the naïve spectral bound of (5) is tight.

The overall variance achieved by the naïve algorithm for any f and uniform node distribution is therefore

$$\mathrm{Var}(Y_{naive}) = \frac{1}{\mathcal{B}}(d+1)(1-\mu)\mu \ . \tag{6}$$

3.2 Edge Algorithm

The first non trivial core set distribution can include any two nodes that are connected in the graph, namely $\{i,j\} \in supp(Q)$ if $\{i,j\} \in E$. Setting the probability of choosing

[3]Recall our assumption that all nodes have at least d neighbors.

edge $\{i,j\}$ to be $p(i,j)$, we have

$$\mathbb{E}\left(|B|\right) \leq 2d - \sum_{\{i,j\}\in E} p(i,j)M(i,j)$$

$$W(i,j) = \frac{p(i,j)}{\left(\sum_{k\in N_i} p(i,k)\right)\left(\sum_{k\in N_j} p(j,k)\right)} .$$

A possible good assignment for p could be achieved by producing a maximal weighted *matching* on the graph G, where the weight of edge $\{i,j\}$ is set to $M(i,j)$, assigning probability $2/n$ for all edges in the matching and probability zero to all other edges. Admittedly, not all graphs yield good maximal weighted matching, or even any matching that includes all nodes. We therefore experimented with a simpler edge selection algorithm that applies to any graph.

3.3 Neighborhood Algorithm

Here the core sets are the graph nodes and their neighbors $supp(Q) = \{N_i^+ | i \in V\}$. In addition, we assign different probabilities to each core set according $Q(N_i^+) = p(i)$, where p is an arbitrary distribution defined over the nodes of G. The node i will be referred to as the center of N_i^+. Hence, a node i belongs to the core set A if one of its neighbors (or itself) is the center node of A. Therefore we have

$$\mathbb{E}\left(|B|\right) \leq 1 + \sum_{\{i,j\}\subset E} (p(i)+p(j))\,(d - M_{i,j})$$

$$W(i,j) = \frac{\sum_{k\in N_{i,j}^+} p(k)}{\left(\sum_{k\in N_i^+} p(k)\right)\left(\sum_{k\in N_j^+} p(k)\right)} .$$

4. CONNECTION TO RANDOM WALKS

In [1], the authors suggested generating the core sets according to a random walk, namely, start at any node and at each step move to one of the neighboring nodes uniformly at random. One can theoretically consider a core set distribution Q that includes all length t paths in the graph. The probability of a core set A is the probability of it being the set of nodes produced by the random walk. Although it is computationally impossible to compute Q it is quite easy to sample from it simply by simulating the random walk. In order to execute the meta algorithm one must also be able to compute $q(i)$. It is well known as per [6] that after a certain number of such steps, one reaches the stationary distribution. According to the stationary distribution the probability of being at node i is proportional to its degree $deg(i)$. Therefore, the expected number of times a node is included in a length t random walk is $t \cdot deg(i)/\sum_{j\in V} deg(i)$. Setting $q(i) = t \cdot deg(i)/\sum_{j\in V} deg(i)$ completes the description of the algorithm. It is worth noting that there is a slight difference in notation between the ones in [1] and those used here. There, a node can appear multiple times in the core set, so in a sense, it behaves more like a list than a set. This is the reason the authors of [1] introduced the multiplicity of nodes in core sets into their estimator.

A similar view is also possible for the other variants of random walks proposed in [1]. The authors use random walks that try to balance the probabilities $q(i)$. This is possible using a Metropolis-Hastings random walk as used, for example, in [5]. Another option is to assign weights to edges and transition with probabilities proportional to edge weights. It turns out that it is possible, in most cases, to assign such weights that the probability of visiting each node is roughly the same. Again, computing or storing Q is computationally impossible but sampling from it is easy. Setting $q(i) = t/n$ and applying the meta algorithm is identical to the algorithms in [1].

One problematic aspect of using random walks is that it is impossible to compute the matrix W and hence impossible to analytically bound the variance for arbitrary unknown functions f. Surprisingly, there is a way to overcome this problem. In particular, one can simulate a very large number of random walks and produce an empirical matrix W'. It is not hard to see that after a sufficient number of simulations, we would have $W' \sim W$ at least in the spectral sense. Moreover, it can be shown using sampling argumentation, that $O(n^2 \log(n))$ simulations would suffice for W' to be close enough to W to give similar bounds.

5. GRADIENT DECENT OPTIMIZATION

In cases where the support of the core set distribution is small, we can directly minimize the overall variance of the estimator. That is, find values of Q that minimize $\mathbb{E}\left(|B|\right)\mathrm{Var}\left(Z\right)$. One obstacle in doing so is that an exact expression for the variance $\mathrm{Var}\left(Z\right)$ is not available. From Proposition 1 we have that $\mathrm{Var}\left(Z\right) \leq \sum_{\{i,j\}\subset U_\mu} W(i,j)/n^2$. Alas, computing this value requires solving an NP-hard problem. We therefore replace it with a simple bound which sums over all elements of W, namely, $\mathrm{Var}\left(Z\right) \leq \sum_{\{i,j\}\subset V} W(i,j)/n^2$. Although this bound is extremely naïve, it serves well as a surrogate to the actual value. The second obstacle is that computing the closure set for a core set is also computationally hard. For this problem we also use a simple bound which is the size of closure set achieved by a greedy algorithm, B_g. Finally we are faced with minimizing $\psi(Q) = \mathbb{E}\left(|B_g|\right)\sum_{\{i,j\}\subset V} W(i,j)/n^2$. Since $\psi(Q)$ is a complex function of the core set distribution we cannot hope to minimize it exactly. We resort to minimizing $\psi(Q)$ heuristically using *Gradient decent* method. The results of using Gradient decent on Neighborhood algorithm core sets are presented in the Section 6.6.

6. EXPERIMENTAL SETUP AND RESULTS

Evaluating algorithms' efficiency or preferring one algorithm to another is impossible in general. The correct choice of algorithm heavily depends on a wide range of parameters. While the authors of [1] report good results of their algorithms when applied to portions of the Facebook network, we observe that it does less well for others. Likewise, our non trivial algorithms mostly outperform the naïve implementation but for some values of f and some graphs the naïve algorithm still prevails. The number of variations possible in the graphs, satisfaction functions, algorithms, and measures of success, is practically endless. Nevertheless, we tried to be as thorough as possible. Our choices are described below.

6.1 Graphs

The first crucial factor in the success of an algorithm is the Network it operates on. In this work we examine 3 different graphs, one real and two synthetic.

DBLP: we used the Digital Bibliography and Library Projects (DBLP) entire database. It contains data about authors of manuscripts. We associated each node in the graph with an author, where an edge corresponds to co-authorship of at

least one paper. The graph we obtained contains 845, 211 nodes, each with at least one edge (authors with no co-authors were discarded).
BA: a synthetic graph constructed according to the model of Albert and Barabási [8]. We start with a ring graph of size 10. Then we add nodes one at a time. Each new node is connected by edges to 10 other nodes already in the graph with probabilities proportional to their degrees.
WS: a synthetic graph constructed according to the model of Watts and Strogatz [9]. To construct a network of n nodes, we start with an n node ring graph. Then, we connect each node to 10 nodes to its right along the ring. Finally, we reroute each edge to a random node with probability 1/2.

One immediate problem we encounter is that, due to our model, nodes with degree lower than d cannot be measured. That is simply because, even if all their neighbors are chosen, they would not have d chosen neighbors. One can think of several solutions for this issue. For example, change the model by deciding that such nodes are still measurable if all their neighbors are chosen. Alternatively, define the mean to not include those nodes and never measure their satisfaction. However, since this is not the main point of the paper, we chose (as in [1]) to simply remove those nodes. Note that after removing some small degree nodes, other nodes might become removable too. Here, we simply continued removing those until all degrees in the graph were at least d. This process will be referred to as *trimming*.

6.2 Satisfaction functions

As explained throughout the paper, one of the most crucial factors governing the variance of our estimation is the satisfaction function f. While our proofs bound the variance for all functions f simultaneously, we experimented with only four kinds of such functions. These were taken to represent extreme cases of network biases.
Uniform: samples uniformly, without replacement, exactly μn nodes from the graphs. Then, set $f(i) = 1$ for chosen nodes and $f(i) = 0$ otherwise. This serves mostly as a sanity check and as a baseline. One cannot expect any meaningful social feature to truly have such a satisfaction function.
BFS: starts a *Breadth First Search* algorithm in an arbitrary node of the graph and assigns a value of $f(i) = 1$ to all nodes it encounters until it reaches a count of μn nodes. Then, the rest of the nodes are set to $f(i) = 0$. This function gives an extreme graph topological bias.
Degree Percentile: assigns the value of $f(i) = 1$ to all nodes in the top μ percentile in terms of degree. In other words, $f(i) = 1$ for the μn nodes whose degree in the graph is the highest. Then, the rest of the nodes are set to $f(i) = 0$. Here we simulate another extreme case of degree bias. This is an important case for two reasons. First, our algorithms are heavily influenced by node degrees and so this choice of f might present a 'difficult' scenario. Second, in reality, social features are not independent of degree biases. This is because the node degree usually relates to the user's activity or 'socialness' in some sense. This function is the extreme case of all satisfaction distributions that are positively correlated with the degree.
Degree Bias: proposed in [1] and gives a less extreme degree bias. Nodes are chosen randomly with probability proportional to $\log(deg(i))$. The process terminates when we have picked μn nodes. Then, we set $f(i) = 1$ for chosen nodes and $f(i) = 0$ otherwise.

DBLP	Uniform	BFS	Degree Percentile	Degree Bias
Naïve	0.41	0.41	0.42	0.41
Edge	0.38	**0.37**	**0.20**	0.36
Edge-Matching	0.32	0.38	0.34	0.32
Neighborhood	0.34	0.85	0.63	0.34
Neighborhood-20	0.32	0.53	0.39	0.32
Neighborhood-40	0.28	0.57	0.46	0.28
Neighborhood-GD	**0.24**	0.48	0.43	**0.24**
Simple-RW	0.30	0.47	0.32	0.29
Metropolis-H-RW	0.29	0.47	0.42	0.28
Matrix-Scaling-RW	0.27	0.49	0.43	0.27
Triangle-Closing-RW	0.26	0.49	0.44	0.26

Table 1: The table gives the Normalized RMSE scores for the various algorithms and satisfaction functions. The graph here is the DBLP graph and the satisfaction functions mean is $\mu = 0.1$.

Barabási	Uniform	BFS	Degree Percentile	Degree Bias
Naïve	0.31	0.31	0.31	0.31
Edge	0.34	**0.30**	**0.17**	0.33
Edge-Matching	0.31	**0.30**	0.29	**0.30**
Neighborhood	0.54	0.60	0.41	0.54
Neighborhood-20	0.40	0.38	0.23	0.38
Neighborhood-40	0.40	0.40	0.23	0.39
Neighborhood-GD	0.36	0.34	0.19	0.35
Simple-RW	0.33	0.31	**0.17**	0.32
Metropolis-H-RW	0.36	0.35	0.28	0.35
Matrix-Scaling-RW	**0.30**	0.31	0.28	**0.30**
Triangle-Closing-RW	0.31	0.31	0.28	**0.30**

Table 2: The table gives the Normalized RMSE scores for the various algorithms and satisfaction functions. The graph contains $n = 10^5$ nodes and is generated according to the model of Albert and Barabási [8]. As before, the satisfaction functions mean is $\mu = 0.1$.

6.3 Algorithms

All the algorithms we have examined produce their estimates according to the meta algorithm (see Algorithm 1). Here we describe algorithms by the manner in which they choose core sets. All Random Walk (RW) based algorithms are described briefly below (see [1] for more details). We have included those to serve mostly as a baseline but also to validate their results on graphs other than Facebook. These algorithms perform random walks on the network and collect nodes they encounter into the core set (see Section 4). Below, we shortly recap each of the tested algorithms.
Naïve: each core set includes one single node chosen uniformly at random (see Section 3.1).
Edge: refers to core sets of size two. An edge is chosen uniformly at random and the core set contains its supporting nodes (see Section 3.2).
Edge-Matching: a variant of the former Edge algorithm. Here, the core sets are also pairs of nodes supported by edges. The idea is to create a set of edges that behaves like a matching but is simpler to obtain. The process proceeds as follows. Start with an empty edge set E_m. For every node,

Watts-Strogatz	Uniform	BFS	Degree Percentile	Degree Bias
Naïve	0.31	**0.31**	0.31	0.31
Edge	0.29	**0.31**	0.25	0.29
Edge-Matching	0.28	**0.31**	0.28	**0.28**
Neighborhood	**0.27**	0.44	**0.24**	0.27
Neighborhood-20	0.30	0.44	0.25	0.30
Neighborhood-40	**0.27**	0.44	0.25	0.27
Neighborhood-GD	0.28	0.41	0.26	**0.28**
Simple-RW	0.28	0.34	**0.24**	**0.28**
Metropolis-H-RW	0.29	0.35	0.27	0.29
Matrix-Scaling-RW	0.28	0.34	0.27	**0.28**
Triangle-Closing-RW	**0.27**	0.34	0.27	0.27

Table 3: The table gives the Normalized RMSE scores for the various algorithms and satisfaction functions. The graph contains $n = 10^5$ nodes and is generated according to the model of Watts and Strogatz [9]. As before, the satisfaction functions mean is $\mu = 0.1$.

pick the edge connecting it to its neighbor (in G) with the least degree with respect to E_m. In case of ties, pick the one maximizing $M_{i,j}$ (for definition see Section 3). Add the picked edge to E_m and continue. The core sets are pairs of nodes supports of edges in E_m chosen uniform at random.
Neighborhood: corresponds to a uniform distribution over the sets N_i^+. This is achieved by selecting a node and its neighbors uniformly at random (see Section 3.3).
Neighborhood-k: a variant of the former Neighborhood algorithm. Here, the algorithm also chooses nodes and their neighborhoods but avoids doing so for nodes of very high degree. More accurately, Neighborhood-k chooses node i uniformly at random. Tehn, if $|N_i^+| \leq k$ it returns N_i^+ otherwise it returns a singleton core set $\{i\}$.
Neighborhood-GD: here the core sets are still N_i^+. However, the distribution Q over them is optimized using Gradient decent method (see Section 5) to reduce the overall estimation variance.
Simple-RW: a random walk algorithm which uses the standard transition probability, i.e., move from node i to j with probability $1/deg(i)$.
Metropolis-Hastings-RW: moves from node i to j with probability $\min(1/deg(i), 1/deg(j))$, and stays in node i with the remainder probability. This produces a uniform stationary distribution but tends to visit the same nodes many time.
Matrix-Scaling-RW: moves from node i to j with probability $w(i, j)$. The latter probabilities are computed by an iterative process that strives to make the stationary distribution as uniform as possible.
Triangle-Closing-RW: the transition probability between nodes i and j depends on the node h visited before i. If $\{h, j\} \notin E$ the transition probability is $w'(i, j)$. If $\{h, j\} \in E$ this probability is increased by a factor $\alpha \geq 1$ to be $\alpha w'(i, j)$. The weights w' are chosen to produce a node distribution which is as close to a uniform distribution as possible.

6.4 Measure of Success

Our main measure of success for an algorithm is the Root Mean Square Error (RMSE) of its outputs. Assume we execute an algorithm t times and produce outcomes $Y_1, \ldots, Y_t$.

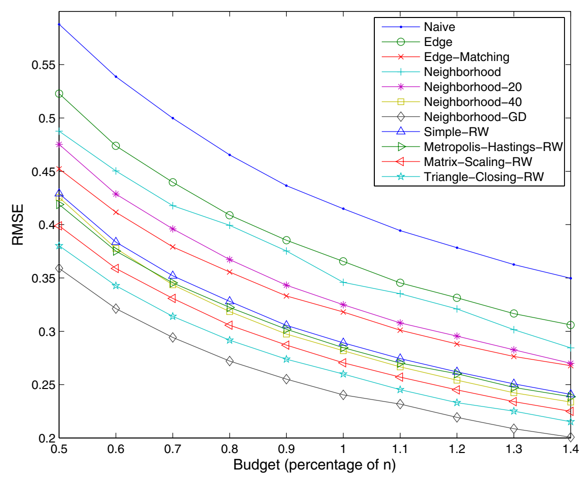

Figure 1: Estimate normalized RMSE for the DBLP graph and the Degree Bias satisfaction function plotted vs. the budget $\mathcal{B}$.

Since the satisfaction function f has mean μ the normalized RMSE is given by $\frac{1}{\mu}(\frac{1}{t}\sum_{i=1}^{t}(Y_i - \mu)^2)^{1/2}$. We chose RMSE as our measure of success since it embodies both accuracy and reliability. Note that since RMSE (squared) estimates $\mathbb{E}[(Y_i - \mu)^2]/\mu^2$, by Markov's inequality we also get confidence intervals.

6.5 Experimental Results

Tables 1–3 give the RMSE values achieved by the different algorithms for different choices of f. Each combination of algorithm and satisfaction function was run 1000 times and the RMSE value is calculated according to Section 6.4. For all tables we set the budget $\mathcal{B}$ to 1% of the network size. This is a reasonable budget for actual bucket tests.

In Figure 1 the estimate normalized RMSE for the DBLP graph and the Degree Bias satisfaction function is plotted vs. the budget $\mathcal{B}$. It is clearly visible that the RMSE decreases with the increase in budget.

6.6 Gradient Decent Experiments

To demonstrate the benefits of the Gradient Decent optimization (see Section 5) we applied the Neighborhood algorithm to the DBLP graph. As before, we iteratively trimmed the minimal degree in the graph to be 10. This resulted in a graph containing $n = 57285$ nodes. In Table 4 we provide several statistics for three different distributions over Neighborhood algorithm core sets, N_i^+ for all $i \in V$. The first is uniform (Uniform), the second is relative to $1/|N_i^+|$ (Degree) and the third is the probability $Q(N_i^+)$ assigned by the Gradient Decent procedure (GD).

By examining the table we observe that the Gradient Decent optimization reduces $\psi(Q)$, mainly by reducing the average closure set size. In parallel, it also increases the efficiency of the algorithm. On the other hand it increases the largest eigenvalue when compared to that calculated for the Uniform distribution.

While it is hard to foresee the exact strategy that Gradient Decent optimization follows to reduce $\psi(Q)$, Figure 2 may

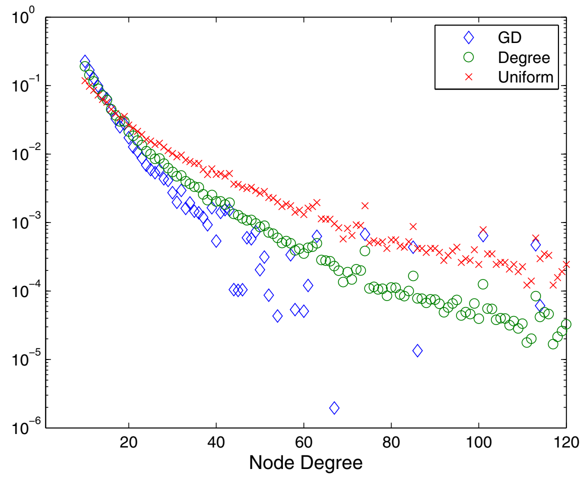

Figure 2: Probability distributions of the size of core sets for the DBLP graph. The three distributions represent different distributions over the core set N_{i+}. Uniform, selects each with constant probability, $1/n$. Degree, selects N_{i+} w.p. proportional to $1/|N_{i+}|$, and GD selects N_{i+} w.p. according to the output of the Gradient Decent optimization.

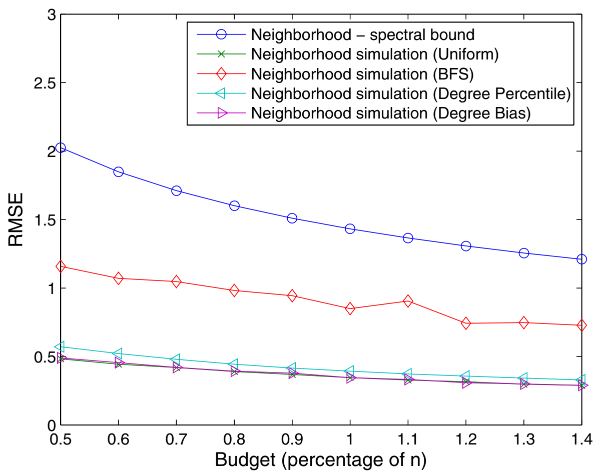

Figure 3: Estimate normalized RMSE upper bound plotted vs. the budget $\mathcal{B}$ for the DBLP graph using the Neighborhood Algorithm. Along with the bounds we give simulation results for different satisfaction functions, namely, Uniform, BFS, Degree Percentile, and Degree Bias. As expected, all simulation results reside below the theoretical bound.

provide some insights. In Figure 2, the DBLP graph degree PDF is plotted for the three core set distributions (Uniform, Degree, and GD). It is apparent that the GD optimization causes the graph degree PDF to drop much faster than those of the Uniform and Degree distributions. It turns out that the GD optimization reduces the probabilities of higher degree nodes. In fact, for the DBLP graph, it assigned a zero probably to any Neighborhood core set of size greater than 114. Hence, more than half of the core sets were discarded!

	Uniform	Degree	GD
Average core set size $\mathbb{E}(\lvert A\rvert)$	22.92	17.096	15.31
Average closure set size $\mathbb{E}(\lvert B_g\rvert)$	72.56	54.77	26.29
Efficiency bound $\mathbb{E}(\lvert A\rvert)/\mathbb{E}(\lvert B_g\rvert)$	0.316	0.312	0.58
$\frac{1}{n^2}\Sigma_{i,j}W(i,j)$	1.360	1.315	1.20
$\psi(Q)$	98.71	72.00	31.66
$\frac{1}{n}\lambda_1(W)$	1.718	4.621	2.13

Table 4: Statistics for different distributions over Neighborhood algorithm core sets applied to the DBLP graph.

6.7 Spectral Bounds

Being able to analytically bound the accuracy (RMSE) of a network bucket test is crucial for two main reasons. Before the test, the administrator must choose the best algorithm to use. After the test, he/she must supply error bounds on the resulting estimate. Given the discussion following Proposition 1, this is a hard computational task. However, using the spectral bound of Equation (5), designers can get a rough bound for this quantity. To demonstrate the benefits of this bound we use values from Table 4, derived for the aforementioned Neighborhood algorithm core set distributions, to calculate bounds for the DBLP graph. The calculated bounds along with their corresponding simulation results are plotted in Figure 3 for $\mu = 0.1$.

7. DISCUSSION AND CONCLUDING REMARKS

In this paper we proposed and analyzed several algorithms for network bucket testing. The achieved results are comparable or better than previous algorithms depending on the setup. However, we argue that the contribution goes beyond that. First, our algorithms are simple to program, provide unbiased estimates, efficient to execute, and analyzable. Moreover, we can efficiently produce good error bounds for their performance. This gives us the ability to choose the best algorithm for a network well before running the test.

In addition, the framework lets designers analyze a very large variety of algorithms. For example, one can consider core sets of triangles in the graph. Or, cover the graph with small tightly connected subgraphs and consider those as core sets. Indeed, the possibilities are endless. We hope the derivations also provide walk-through examples on how to analyze those.

An additional benefit which is not mentioned in the paper but is an immediate outcome of the Gradient Decent approach: That is, one can combine any number of different algorithms and consider the superset of their core sets. Applying the Gradient Decent process to the core superset can automatically mix the different algorithms. The resulting mixed algorithm is guaranteed to perform at least as good as the best single algorithm.

8. REFERENCES

[1] Lars Backstrom and Jon M. Kleinberg. Network bucket testing. In *WWW*, pages 615–624, 2011.

[2] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities: an $o(n^{1/4})$ approximation for densest k-subgraph. In *STOC*, pages 201–210, 2010.

[3] Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, pages 84–95, 2000.

[4] Uriel Feige, Guy Kortsarz, and David Peleg. The dense k-subgraph problem. *Algorithmica*, 29:2001, 1999.

[5] Minas Gjoka, Maciej Kurant, Carter T. Butts, and Athina Markopoulou. Walking in Facebook: A Case Study of Unbiased Sampling of OSNs. In *Proceedings of IEEE INFOCOM '10*, San Diego, CA, March 2010.

[6] L. Lovasz. Random walks on graphs. a survey. *Combinatorics*, 1993.

[7] David Gibson Ravi, Ravi Kumar, and Andrew Tomkins. Discovering large dense subgraphs in massive graphs. In *In VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 721–732, 2005.

[8] Albert Reka and Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, June 2002.

[9] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.