Effective and Efficient Sampling Methods for Deep Web Aggregation Queries

Fan Wang
The Ohio State University
wangfa@cse.ohio-state.edu

Gagan Agrawal
The Ohio State University
agrawal@cse.ohio-state.edu

ABSTRACT

A large part of the data on the World Wide Web resides in the *deep web*. Executing structured, high-level queries on deep web data sources involves a number of challenges, several of which arise because query execution engines have a very limited access to data. In this paper, we consider the problem of executing aggregation queries involving data enumeration on these data sources, which requires sampling. The existing work in this area (*HDSampler* and its variants) is based on simple random sampling. We observe that this approach cannot obtain good estimates when the data is skewed. While there has been a lot of work on sampling skewed data, the existing methods are based on prior knowledge of data, and are therefore not applicable to hidden databases.

In this paper, we present two *prior-knowledge-free* sampling algorithms, *Adaptive Neighborhood Sampling* (ANS) and *Two Phase adaptive Sampling* (TPS), which allow an aggregation query to be answered with a high accuracy (even when there is a skew), and a low sampling cost. For this purpose, we have developed robust estimators for aggregation functions including AVG, MAX, and MIN.

Our experiments show that for data with a moderate or a large skew, ANS and TPS yield more accurate estimates, outperforming HDSampler by a factor of 4 on the average. Even for the cases where data has a small skew, our TPS method has an important advantage, which is that it has only one-third of the sampling costs of HDSampler.

Categories and Subject Descriptors

H.2.4 [**Database Management**]: Query Processing

1. INTRODUCTION

A large portion of the data on the World Wide Web resides in the *deep web*. To access such *hidden data*, a user must issue queries through the input interfaces of deep web data sources. Then, answers are returned to the user as dynamically generated HTML pages. An study on the deep web estimated that the public information in the deep web is 500 times larger than the surface web, with 7,500 Terabytes of data, across 200,000 deep web sites [3].

With the growth of the size and popularity of the deep web, it is becoming increasingly desirable to support structured, high-level queries over deep web sources. Such support, however, involves many distinct challenges, depending upon the queries that one may support [12, 22]. In this paper, we focus on aggregation queries that require *data enumeration*. These are distinct from the aggregation queries that can be answered directly using a data source's input interface. For example, the query *"find the hotel with the lowest price in Boston on 2/15/2010"* can be directly answered using a travel web-site, and specifying the city and the date. In contrast, the following aggregation query requires data enumeration:

Example 1: *Suppose a student wants to study US aviation market, and he wants to obtain the average airfare from the US to Europe, across all flights in the next week using Expedia.com.*

To obtain the exact answer for this aggregation query, one needs to enumerate every pair of US and European cities, and issue corresponding deep web queries. This can be extremely time-consuming, if not impossible. In general, finding the exact answer to an aggregation query that requires data enumeration is not practical due to the following reasons. First, it is extremely hard to obtain all possible input values to enumerate the data. Second, deep web data is returned over a network, and thus, executing a large number of queries can be extremely time consuming. Some data sources even charge access costs. Third, many deep web sources limit the number of queries a particular IP address can issue, or the number of data records can be returned, to either protect their data from being completely downloaded, or to disallow a denial-of-service attack.

Therefore, deep web aggregation queries can only be answered approximately, using sampling. This, however, requires effective and efficient sampling. Even generating random samples from hidden databases can be challenging. Recently, Dasgupta *et al* have addressed this problem by developing HDSampler and its variants [10, 8, 11], which are able to select a *simple random sample* (SRS) from hidden databases.

Our work is driven by the same motivation, but further considers the problem that a random sampling approach like HDSampler may not work effectively, because of the following reasons:

Low Estimation Accuracy on Skewed Data: HDSampler and related methods obtain a SRS from a hidden database. As is well known, SRS cannot provide good estimates on skewed data [23, 5]. For hidden databases, we do not know the data distribution in advance. The problem arising when the hidden data is skewed is shown in the example below.

Example 2: *We have a data source* $D = (1, 1, 1, 1, 1, 1, 1, 1, 10, 1000)$. *We want to compute the average value on D. The data distribution in D is highly skewed. The true average is* 101.8. *Suppose HDSampler takes a SRS of size 2. There are four possible samples we could obtain:* $s_1 = (1, 1)$, $s_2 = (1, 10)$, $s_3 = (1, 1000)$, *and*

Table 1: Different Estimates on Skewed Data

HDSampler Sample	HDSampler Estimate	ACS Networks	ACS Estimate
s_1=(1,1)	1	({1},{1})	1
s_2=(1,10)	5.5	({1},{10*,1000})	273.47
s_3=(1,1000)	500.5	({1},{10,1000*})	273.47
s_4=(10,1000)	505	({10*,1000*})	272.98

$s_4 = (10, 1000)$. *The estimated averages obtained using these samples are shown in the second column of Table 1. None of them yields a good estimate for the true average. This is because we sample data with equal* selection probability. *However, the* skew causing data*, which usually dominates the true average, only accounts for a small portion of the entire data. Therefore, these data records are often under-sampled.*

High Sampling Cost: In the deep web, *sampling cost* refers to the network transmission time for the queries issued to obtain the sample. As studied by Dasgupta *et al* [10, 11], to obtain one random sample using HDSampler and its variants, multiple queries are often needed. Thus, obtaining a modest sized SRS using HDSampler would involve a high sampling cost.

The limitations of SRS for skewed data are well known, and many sampling methods have been developed to address these. For example, many *unequal probability sampling* techniques have been developed and demonstrated to be effective with skewed data. Examples include density biased sampling [23], stratified sampling [7], sampling with outlier-indexing [5], histogram and wavelet based sampling [4, 24] and random sampling with supplemental statistics [14]. However, all these methods are built based on two assumptions. First, they require knowledge of the data distribution or certain *key statistics*. Second, the sampler has access to the full dataset, i.e., it can off-line scan the full data at least once, and obtain certain data statistics. These two assumptions are reasonable in a relational database. However, they do not hold true when there is a very limited access to hidden data, as in the deep web case. Furthermore, many deep web sources update data frequently. Even if we obtain partial data statistics ahead of time, they may not hold true at the time when we are responding to a user query.

In this paper, we have developed and evaluated two new *adaptive sampling* algorithms for the deep web scenario. They are the *Adaptive Neighborhood Sampling* (ANS) algorithm and the *Two Phase adaptive Sampling* (TPS) algorithm. Both of them have two advantages over HDSampler. First, both of them can provide accurate estimation for aggregation queries on skewed data, without requiring any knowledge of hidden data distribution or other statistics. Second, besides being applicable on skewed data, TPS algorithm incurs significantly lower sampling costs compared to HDSampler. In this paper, we considered three types of aggregation queries: **Average (AVG)**, **Maximal (MAX)** and **Minimal (MIN)**. In addition, **SUM** queries can be answered by combining the results from an **AVG** query using our methods and a size aggregate obtained using a recently proposed method [9].

The basic idea underlying the ANS and TPS algorithms is as follows. Similar to the existing *unequal probability sampling* algorithms, ANS and TPS select a random sample *biased* towards the *skew causing* data. However, without knowledge of hidden data distribution, ANS and TPS sample skew causing data by utilizing *neighborhood proximity*. We observe that skew causing data, which also has a *low frequency*, usually forms *clusters* with respect to certain attribute(s). For example, in household income data, very high income households usually form clusters. i.e., their household heads may have certain occupations such as doctor or lawyer, or the physical locations of these households may be clustered. Unlike existing algorithms that handle each sampled unit independently, in ANS and TPS, we aggressively consider *neighboring units* for sampled units. For this, when *upper quantile* data (likely but not necessarily the skew causing data) is sampled, ANS and TPS explore its *neighborhood*, so as to increase the selection probability of the skew causing data.

The overall contributions of our work can be summarized as follows. We have developed two novel sampling algorithms, with mathematical analysis, to approximately answer aggregation queries on deep web sources. These algorithms yield accurate estimation on skewed data, without requiring prior knowledge of the data distribution. In addition, the TPS algorithm has lower sampling costs. Our experiments show that for data with a moderate or a large skew, ANS and TPS yield more accurate estimates, outperforming HDSampler by a factor of 4. For data with only a small skew, TPS method only incurs one-third of the sampling cost of HDSampler.

2. BACKGROUND

In this section, we first review HDSampler, a recently developed system for obtaining simple random samples from hidden databases [10, 8]. Then, we give an overview of *adaptive cluster sampling*, an existing method for sampling skewed datasets [26].

2.1 HDSampler and Its Estimators

HDSampler performs a *random walk* through a tree. Each level of the tree represents an input attribute and the database tuples exist at some of the leaves. Each path from root to a leaf represents a specific assignment of values to input attributes. HDSampler performs a random *drill-down* by adding randomly selected input attributes as query predicates, until a tuple is selected. Because a tuple at higher level of the tree is more likely to be sampled than one at lower levels, the retrieved tuples are post-processed by *rejection sampling* to enforce equal selection probability.

For a data source D with Y being the attribute of interest, we have $\mathcal{S} = \{y_1, y_2, \ldots, y_n\}$ as the SRS selected using HDSampler. Here, n is the sample size. Each y_i in $\mathcal{S}$ is a *sampled unit*. Using $\mathcal{S}$, the *estimates* of average, and the maximum values of Y can be computed as $\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n}$, and $\hat{y}_{max} = Max_{(i=1\ldots n)}\{y_i\}$. They are the *estimators* of HDSampler.

To evaluate the accuracy of an estimator, we use the metric *Absolute Error Rate (AER)*. For a variable with true value θ and estimated value $\hat{\theta}$, the AER of the estimator $\hat{\theta}$ is $AER(\hat{\theta}) = \frac{|\hat{\theta}-\theta|}{\theta}$.

2.2 Adaptive Cluster Sampling

The basic idea of the ANS and TPS algorithms is inspired by *Adaptive Cluster Sampling (ACS)* [26]. ACS was originally motivated by the problem of sampling *rare* and *clustered* population, such as rare species like shrimp, which is in high abundance in certain spatial areas. We may want to estimate the total number of such animals within a certain area which is partitioned into *grids*. The animal population is highly skewed, implying that most grids do not have any animals, whereas, some areas have a high density. We are likely to see two key features. First, it is hard to know their distribution in advance over the entire area. Second, they often form *clusters*. Thus, if we find them in one spatial area, we are more likely to find them in neighboring areas. With such properties, SRS usually does not perform well.

ACS is proposed for such cases, and works in two steps as follows. First, a SRS is selected, denoted as the *initial sample*. If any unit (grid) in the initial sample satisfies a pre-defined *limit condi-*

tion, such as the number of animals in the unit is greater than a threshold, the unit is called a *unit of interest*. Then, for each such unit of interest in the initial sample, its *neighboring* units are further sampled. If any neighboring unit also satisfies the limit condition, it is added to the sample, and this process is repeated. The idea is that because animals often form clusters, if we examine the *neighboring units* of a *unit of interest*, we are likely to obtain more *units of interest* into the sample. The neighborhood is usually determined by *spatial adjacency*. A unit of interest with all its neighboring units that also satisfy the limit condition form a *network*. Thus, in ACS with an initial sample of size n, we will eventually have n networks $\mathcal{S} = \{s_1^*, s_2^*, \dots, s_n^*\}$, where s_i^* is the i^{th} network.

Let us apply ACS to Example 2 in Section 1. We define the records with values no less than 10 as the *units of interest*, and neighborhood is determined based on spatial adjacency, implying that records 10 and 1000 are neighbors. Using the ACS estimators in [26], the ACS networks and ACS estimated averages are shown in Table 1. The unit(s) enclosed within brackets {} represent(s) a network. In each network, the unit with a $*$ is the initial unit of interest, and other units are neighboring units. We observe that for the initial samples s_2, s_3, and s_4, ACS gives much better estimates than HDSampler, which is based on SRS. This is because in these three samples, the *skew causing data* is included. However, if the initial sample is s_1, the ACS estimate is the same as HDSampler.

From this example, ACS appears like a promising approach for sampling hidden databases. However, ACS also has three limitations. First, the performance of ACS depends on the initial sample. If there is no unit of interest in the initial sample, ACS degrades to SRS. Second, ACS has not been developed or analyzed for cases with a sampling cost limit. In the deep web scenario, since queries are executed in an interactive setting with a user, it is necessary to consider a *cost limit*. This could further include: 1) how many units in the initial sample are allowed? and 2) how many neighboring sampled units are allowed? Third, the initial sample in ACS is selected as a SRS. As stated in Section 1, obtaining SRS on a deep web source has a high cost.

Our ANS method overcomes the first two limitations of ACS, and the TPS method addresses the first limitation of ACS and also provides a practical solution for the third limitation.

3. NEW SAMPLING ALGORITHMS

In this section, we first state several definitions to formulate the our sampling problem. Then, we describe the ANS and TPS algorithms and develop estimator for each of them, for computing the average (AVG) aggregation function. Towards the end of this section, we develop estimator for the maximum (MAX) and minimum (MIN) function.

3.1 Problem Formulation

Consider a deep web source D with N (unknown) tuples $\{r_1, \dots, r_N\}$ over a set of m attributes $Attr = \{A_1, \dots, A_m\}$. Attributes can be numerical or categorical. $IN = \{A_1, \dots, A_s\}$ is the input attribute set of D. The *attribute of interest* is a numerical attribute A_q, where $A_q \in Attr$ and $A_q \notin IN$. For an aggregation query on A_q which requires data enumeration, we select a sample $\mathcal{S} \in D$, to estimate the result $AGG(A_q)$.

3.1.1 Unit of Interest

For a record r_i, we use y_i, the value of r_i's attribute of interest, to represent r_i. Then, we denote a unit y_i to be a *unit of interest*, if y_i satisfies the *limit condition* $y_i > \eta$. Here, η is the *upper limit value* that is initialized based on domain knowledge (η will be adapted in our algorithm). For dataset with negative values on the attribute of interest, the limit condition $y_i < \eta'$ is also considered.

3.1.2 Data Record Neighborhood

In ACS, the neighborhood is determined based on spatial adjacency. Here, we define neighborhood in the deep web scenario. For a data source D with m attributes, we consider D as an *m-dimensional* space, and each record r is a point, denoted as $r = \{a_1, \dots, a_s, y, \dots, a_m\}$. a_i is the value of the attribute A_i. The first s values correspond to the input attributes, and y is the value of the attribute of interest. Intuitively, the neighborhood of r is the set of data records that are *adjacent* to r with respect to the input attributes. Specifically, we take data records that have a different value for at most one of the input (or the first s) dimensions. This is denoted as

$$Neighborhood(r) = \{r' = \{a_1, \dots, neighborhood(a_j),$$
$$\dots, a_s, *, \dots, *\} | 1 \leq j \leq s\}$$

where, $*$ denotes values for non-input attributes of r'. If A_j is a numerical attribute, $neighborhood(a_j) = \{a_j, a_j \pm unit\_length\}$, where $unit\_length$ is chosen depending upon the scale of A_j.

For a categorical attribute A_j, we assume its values can be *categorized* through domain knowledge. For example, the `occupation` attribute can be categorized into *high wages*, *median wages*, and *low wages*. Thus, in computing the neighborhood of a_j for attribute A_j, $neighborhood(a_j)$ contains all values in the same category as a_j.

Then, a neighboring unit r' of r falls in either of the following two types: 1). **Immediate Neighbor**, r' has the *same* input attribute values as r but different values on other attributes, 2). **Collateral Neighbor**, r' has *at most* one input attribute value different from that of r, but this value is still *adjacent* to the corresponding value for r. As an example, if we have a data record in a 2-dimension space $r = (a_1 = 2, y = 4)$, then the neighborhood of r contains $(2, *)$, $(1, *)$,$(3, *)$, of which the first one is an *immediate neighbor* and the other two are *collateral neighbors*. For a record r, because its *immediate neighbors* have the same input attribute values as r, they are returned in one result set with r. Therefore, we could consider immediate neighbors being available free of additional sampling cost. This provides our algorithms a critical advantage on sampling costs. Obtaining each collateral neighbor of r incurs the same sampling cost as that for obtaining r.

3.2 Adaptive Neighborhood Sampling (ANS)

In this section, we describe how our ANS algorithm address the first two limitations of ACS (Section 2.2), which are first, the performance of ACS depends on the initial sample, and second, ACS has not been developed for cases with a time-limit, or a *termination* condition on sampling.

The ANS algorithm selects sampled units in two steps. First, ANS selects an SRS as the *initial sample* using HDSampler. Second, for each sampled *unit of interest* r_i in the initial sample, the units *neighboring* r_i are further sampled to form the network corresponding to r_i. Formally, the method is shown as Algorithm 3.1. We detail these two steps of the ANS algorithm in the following paragraphs.

3.2.1 Selecting Initial Sample in ANS

The first limitation of ACS is that if there is no unit of interest in the *initial sample*, ACS degrades to random sampling. To overcome this limitation, an intuitive solution is to ensure that a certain number of *units of interest*, say k, are included in the *initial sample*. For this, we perform random sampling until k *units of interest* are observed in the initial sample. However, in order to achieve k units of interest in the initial sample, the final size of the initial sample can be arbitrarily high. Therefore, we must include appropriate *termination rules* while obtaining the initial sample, to limit the costs.

In a real query system, a user may require an answer to his/her query should be obtained within a specified time limit, though the user may be flexible with respect to the accuracy of the results obtained. Based on profile or history information, we may know the *average network delay* for one sampled unit. Similarly, we may also know the *average neighborhood size* of a data record. Then, given a cost limit C (in terms of time), we could decide the corresponding maximal size of the initial sample allowed, which is denoted as n.

ANS Termination Rules for Obtaining Initial Sample: In ANS, to obtain the initial sample, we use HDSampler to select a list of random sampled units until either of the following two *termination rules* are satisfied: 1). we have observed k number of units of interest in the current initial sample, or, 2) the size of the current initial sample reaches n. The first termination rule implies that if we have already obtained the desired number of units of interest in the initial sample, we could stop sampling even before reaching the limit n. The second rule implies that if we cannot obtain k units of interest in the initial sample within the limit n, we need to just proceed with the initial sample already obtained.

Algorithm 3.1: ANS(k, n, η)

```
/*The first step, obtain initial sample*/
$of\_interest = 0$ /*number of units of interest in initial sample*/
$initialsize = 0$; /*size of initial sample*/
/*consider the two termination rules*/
while $initialsize < n$ AND $of\_interest < k$
  $r = HDSampler()$ /*find a random sampled unit*/
  add $r$ to the initial sample
  $initialsize++$
  if $r$ is a unit of interest
    $of\_interest++$;
/*the second step, explore the neighborhood*/
foreach unit of interest $r$ in the initial sample
  $networksize = 0$
  $Neighbors = FindNeighbor(r)$ /*find the neighboring units of r*/
  foreach $nr \in Neighbors$
    if $nr.y > \eta$
      /*nr satisfies the limit condition*/
      add $nr$ to the network of $r$
      $networksize++$
      add $FindNeighbor(nr)$ to $Neighbors$
      /*adding neighbors in a recursive manner*/
  /*adaptively adjust the value of η*/
  if $networksize > upperthreshold$
    /*too many neighbhoring units added*/
    $\eta = \eta + unit\_length$
  else
    if $networksize < lowerthreshold$
      /*too few neighbhoring units added*/
      $\eta = \eta - unit\_length$
```

3.2.2 Selecting Neighborhood Sample in ANS

In the original ACS method as introduced in Section 3.1, if a sampled unit r is a unit of interest, i.e., the value of the attribute of interest of r is greater than the *upper limit value* η, then r's neighbors that also satisfy the limit condition are also included in the final sample. The original ACS algorithm assumes a fixed η value for the entire sampling process.

Clearly, the choice of the value of η is important for the process. If η is too small, too many neighboring units will be added to the final sample. This could involve extra sampling costs since collateral neighbors have sampling cost. If η is too large, we end up with too few neighboring units, which in turn could negatively impact the estimator. In practice, it is hard to find a good value for η at the beginning. To make our ANS algorithm more flexible and robust, we have developed a method to adaptively adjust η to control the number of neighboring units that are added to the final sample during the execution of the algorithm.

The method works as follows. After neighborhood sampling for a unit of interest r_i in the initial sample is finished, if the current η has introduced too many neighboring units into the neighborhood of r_i, we will increase the value of η. Thus, for other units of interest left in the initial sample, the number of neighborhood units added to their network could be decreased. If the current η introduces very few neighboring units into the network of r_i, we will decrease η.

3.3 Estimators for ANS Algorithm

In this section, we develop the AVG estimator for the ANS algorithm. We assume that the number of sampled units in the initial sample is n, and the number of units of interest in the initial sample is l. As introduced in Section 3.2.2, the ANS algorithm obtains a set of sampled networks, which is $\mathcal{S} = \{s_1^*, s_2^*, \dots, s_n^*\}$, where s_i^* is the i^{th} network formed by the i^{th} unit in the initial sample.

Developing such estimator involves several challenges, since the algorithm can stop with either of the two termination conditions, and the value of η can be dynamically modified. Thus, we first develop the estimator for the case that a fixed η value is used in the ANS algorithm. Then, we modify the estimator for the case when different η values are used in the algorithm.

3.3.1 Estimators for Fixed η

In designing the estimator for the ANS algorithm with a fixed η value, we need to consider two cases, corresponding to the two termination rules. We first design estimator for the sample selected based on each termination rule independently, then we combine them together and provide theoretical analysis.

Estimator for AVG Under 1st Termination Rule: Using the first termination rule, the initial sample $\mathcal{S} = \{s_1^*, s_2^*, \dots, s_n^*\}$ contains k units of interest and the size of the initial sample is smaller than or equal to the limit n. An intuitive AVG estimator will be the *sample average*, i.e., $\bar{y} = \frac{\sum_{i=1}^{n} \bar{y}_i^*}{n}$, where $\bar{y}_i^*$ is the average value of the attribute of interest in the network s_i^*. However, under the first termination rule, the size of the initial sample depends on when the k^{th} unit of interest is sampled. Thus the initial sample size n is not *fixed*. As a result, the above estimator, designed for the cases with fixed sample sizes, is *biased* [19].

To address this problem, we have developed an alternative estimator. In the data source D, suppose the percentage of units of interest with respect to the entire data set is β. Then, an unbiased estimator for the population average $\bar{y}$ for the attribute of interest using the sample obtained under the first termination rule is

$$\bar{y}_{tr1} = \hat{\beta}\bar{y}_1^* + (1 - \hat{\beta})\bar{y}_2^*$$

Here, $\bar{y}_1^* = \frac{\sum_{i \in \mathcal{S}_1} \bar{y}_i^*}{k}$, where $\mathcal{S}_1$ is the set of networks formed by the k units of interest in the initial sample. Also, $\bar{y}_2^* = \frac{\sum_{i \in \mathcal{S}_2} \bar{y}_i^*}{k}$, where, $\mathcal{S}_2$ is the set of networks formed by ordinary units (i.e. units not of interest) in the initial sample. $\hat{\beta}$ is the estimator of β, which can be approximated by $\hat{\beta} = \frac{k-1}{n-1}$ [20].

LEMMA 3.1. *The estimator $\bar{y}_{tr1}$ is an unbiased estimator under the first termination rule.*

PROOF. In a data source D with N records, suppose the total number of units of interest is M, we know that $\beta = \frac{M}{N}$. We use

$\mathcal{U}_{\mathcal{N}-\mathcal{M}}$ to represent all ordinary units in D, and use $\mathcal{U}_{\mathcal{M}}$ to represent all units of interest in D. Let f_i to be the number of times the i^{th} unit from the set $\mathcal{U}_{\mathcal{N}-\mathcal{M}}$ appears in the sample, and we have $f_i \sim binomial(n-l, \frac{1}{N-M})$. g_i is the number of times the i^{th} unit from the set $\mathcal{U}_{\mathcal{M}}$ appears in the sample, and we have $g_i \sim binomial(l, \frac{1}{M})$. Then the expected value of $\bar{y}_{tr1}$ can be obtained as

$$E[\bar{y}_{tr1}] = E[\hat{\beta}\bar{y}_1^* + (1-\hat{\beta})\bar{y}_2^*] = E[\frac{\hat{M}\bar{y}_1^*}{N} + \frac{(N-\hat{M})}{N}\bar{y}_2^*]$$
$$= E[\frac{N-\hat{M}}{N-M}\frac{\sum_{i\in\mathcal{U}_{\mathcal{N}-\mathcal{M}}} y_i}{N} + \frac{\hat{M}}{M}\frac{\sum_{i\in\mathcal{U}_{\mathcal{M}}} y_i}{N}] = \bar{y}$$

Thus, $\bar{y}_{tr1}$ is an unbiased estimator under the first termination rule. $\square$

Estimator for AVG Under 2nd Termination Rule: Using the second termination rule, the initial sample $\mathcal{S} = \{s_1^*, s_2^*, \ldots, s_n^*\}$ has a *fixed* sample size n. As a result, the *sample average estimator* can be used here as an unbiased estimator. An unbiased estimator for the population average $\bar{y}$ for the attribute of interest using the sample obtained under the second termination rule is

$$\bar{y}_{tr2} = \frac{1}{n}\sum_{i=1}^{n}\bar{y}_i^*$$

where $\bar{y}_i^*$ is the average value of the attribute of interest in the network s_i^*.

Combined Estimator for AVG: We use l to represent the total number of units of interest in the initial sample. The estimator for $\bar{y}$ for ANS algorithm considering two termination rules with fixed η value is as follow.

$$\bar{y}_{ANS} = \begin{cases} \hat{\beta}\bar{y}_1^* + (1-\hat{\beta})\bar{y}_2^* & l = k, \\ \frac{1}{n}\sum_{i=1}^{n}\bar{y}_i^* & l < k. \end{cases}$$

When considering the estimators under each termination rule independently, both of them are unbiased. However, if the two termination rules are jointly considered, the size of the initial sample from ANS is *not fixed*, whereas, one part of the combined estimator requires a *fixed* sample size. As a result, the combined estimator is biased. Lemma 3.2 computes the expectation of $\bar{y}_{ANS}$ showing the bias of this estimator.

LEMMA 3.2. *$E[\bar{y}_{ANS}] = \bar{y} \times Pr[l \le k]$, where $Pr[l \le k]$ is the probability that the actual number of units of interest in the final initial sample being smaller or equal to k.*

The proof of Lemma 3.2 is similar as Lemma 3.1. The detailed proof can be found in [27].

Next, lemma 3.3 shows that if k is small, the bias of $\bar{y}_{ANS}$ is small. In our experiment, we show that for relatively small k ($k = 8$), the biased estimator has good performance.

LEMMA 3.3. *For a relatively large data set, if k is small, the bias factor in $\bar{y}_{ANS}$ tends to be 1, i.e., $Pr[l \le k] \to 1$. Therefore, the bias of the estimator $\bar{y}_{ANS}$ is very small.*

PROOF. We give a quick sketch of the proof here. The variable s, the actual number of units of interest in the final initial sample, follows the hypergeometric distribution $s \sim hypergeometric(N, M, n)$. When $M \ll N$ and $k \ll M$, we have

$$Pr[s \le k] = \sum_{k=0}^{s}\frac{\binom{k}{M}\binom{n-k}{N-M}}{\binom{n}{N}} \xrightarrow{M\ll N, k\ll M} \frac{\binom{n}{N}}{\binom{n}{N}} = 1$$

In our scenario, since the skew causing data usually only accounts for a very small portion of the entire data, we have $M \ll N$. Furthermore, because it is hard to sample the skew causing data, we have $k \ll M$ for a modest sized sample. Both of the above two conditions hold in our scenario, and thus, the result from Lemma 3.3 can be applied to our case. $\square$

In statistics, a *biased estimator* is not necessarily a poor or unacceptable estimator. If the bias is not severe, at least under some realistic conditions, and the biased estimator has other desirable properties, we still favor a biased estimator over an unbiased estimator without the same desirable properties. Lemma 3.3 shows that under certain conditions, which often hold in practice, the bias of the combined estimator described above is very small. This is further validated by our experiments. Furthermore, the combined estimator has two very desirable properties. First, it supports two practical termination conditions, whereas the original estimator for adaptive clustering sampling introduced in Section 2.2 does not consider any termination condition. Second, as we will show through our experiments, the estimation accuracy of the combined ANS estimator is better than that of the SRS estimator used by HDSampler.

3.3.2 Estimators for Multiple η

When we use different η values in ANS, the final sample is *post-stratified* into a list of *stratums* based on different η values. Therefore, we modify the estimator developed for fixed η using post-stratification. The estimators are modified as follows.

Estimator for AVG with Multiple η: The estimator of the population average using post-stratification in the ANS algorithm is

$$\bar{y}_{post} = \sum_{h=1}^{H}\frac{n_h}{n}\bar{w}_h$$

where n_h is the total number of units in the initial sample that belongs to the h^{th} stratum. n_h can be obtained after post-stratification. $\bar{w}_h$ is the estimated average value using all the sampled units in the h^{th} stratum. Since all the sampled units in one stratum share the same η value, $\bar{w}_h$ can be computed using the estimator $\bar{y}_{ANS}$ shown earlier. The fraction $\frac{n_h}{n}$ shows the percentage of the sampled units in the initial sample falls into the h^{th} stratum, and it is used as the weight for the estimated average from the h^{th} stratum.

3.4 Two Phase Adaptive Sampling (TPS)

In the ANS algorithm, we still need to use HDSampler to select an initial sample, which can result in high *sampling costs*. In this section, we present another algorithm, which is the TPS algorithm.

The basic idea of TPS is as follows. We consider all data records in a data source D as being partitioned into a set of sub-spaces based on the value of input attributes. For example, if a data source D has a single categorical input attribute with 5 distinct values, we could consider the data in D are partitioned into 5 sub-spaces. Numerical input values can be partitioned according to ranges.

The key observation is that a large set of the records from a sub-space can be retrieved with a single query over a deep web source (probably with clicking the “next” button). Most data sources even allow a range query for a numerical input attribute. Given an overflowing query (sub-space), HDSampler considers it as invalid and continue to drill down. In contrast, TPS will use several data records from its answer set, but with more advanced estimators. In this way, with the same sampling cost, TPS could have more sampled units than HDSampler. In other words, TPS can have a much lower sampling cost for a fixed sample size.

The algorithm proceeds as follows.

1. Suppose there are M sub-spaces in D. We randomly choose m sub-spaces. This can be done by random assigning valid values or ranges for the input attributes and issuing the corresponding queries to obtain the answer set. The sampling cost for this stage can be viewed as being proportional to m.

2. For the i^{th} sampled sub-space, we select a *first-phase* random sample of size n_{i1}. Since the data records in the i^{th} sampled sub-space have been delivered to the system, selecting these samples does not cause any additional queries, and thus can be viewed as free of sampling costs. One exception may be that we need to click the "next" button in the web browser to obtain additional page(s), but we consider this a part of the sampling cost in step 1.

3. For the i^{th} sampled sub-space, if any unit in the first-phase sample satisfies the limit condition, we select a *second-phase* random sample from the sub-space with the size n_{i2}. Otherwise, we do not select the second-phase sample, i.e., $n_{i2} = 0$. These are also obtained from the data records obtained originally, and thus, we view second-phase sampling as not imposing any additional costs. The underlying idea is that based on *neighborhood proximity*, if any unit in the first-phase sample is a unit of interest, it is likely that this sub-space contains other units of interest.

Then, for the i^{th} sampled sub-space, the final sample is s_i with the size $n_i = n_{i1} + n_{i2}$. We denote $s_i = \{y_1, y_2, \ldots, y_{n_i}\}$. We use N_i to denote the total number of data records in the i^{th} sub-space, which is the total number of matched records of a query that most data sources often return. We first try to find the summation estimator for the i^{th} sampled sub-space.

Summation Estimator for the i^{th} Sub-space: Due to the use of overflowing queries in TPS (similar to [11]), the selection probability for the units in the TPS sample for the i^{th} sub-space is not uniform. Thus, an intuitive estimator for summation, $\hat{t_i} = \frac{N_i}{n_i}\sum_{j=1}^{n_i} y_j$, is biased.

In the following, we first propose a trivial unbiased estimator then, we use the **Rao-Blackwell Theorem** to improve the *efficiency* of the trivial unbiased estimator.

Considering the first two drawn units y_{i1} and y_{i2} in the TPS sample for the i^{th} sub-space only, we have a trivial unbiased estimator for the sub-space SUM for the i^{th} sub-space as.

$$\hat{t_i} = y_{i1} + \frac{y_{i2}(1-p_{i1})}{p_{i2}}$$

where p_{i1} and p_{i2} are the probabilities of choosing y_{i1} and y_{i2} in the sample for the i^{th} sub-space respectively.

We use two indicator variables, I_{ij}, which is 1 if y_{ij} is the first sampled unit in the sample s_i and 0 otherwise, and $I^{'}_{ij'}$ which is 1 if $y_{ij'}$ is the second sampled unit in the sample s_i and 0 otherwise. Then, we can write the above estimator as

$$\hat{t}_{i\_trivial} = \sum_{j=1}^{N_i}\sum_{j'>j}^{N_i} y_{ij}I_{ij} + \frac{y_{ij'}(1-p_{ij})}{p_{ij'}} I^{'}_{ij'}$$

Next, we improve the above estimator using the Rao-Blackwell Theorem.

THEOREM 3.4. (Rao-Blackwell Theorem [25]) *Let T be any unbiased estimator of a parameter ϕ, and let W be sufficient for ϕ. Define*

$$T_w = E[T|W] = \eta(W)$$

Then T_w is an unbiased estimator of T, and T_w is more efficient than T.

Based on the Rao-Blackwell theorem, we improve the estimator $\hat{t}_{i\_trivial}$. We define s_i to be the final sample we selected for sub-space i and s_i is a sufficient statistic for $\hat{t}_{i\_trivial}$. Applying the Rao-Blackwell theorem, we have a new estimator

$$\hat{t_i} = E[\hat{t}_{i\_trivial}|s_i] = \sum_j \sum_{j',j'>j} (y_{ij} + \frac{y_{ij'}(1-p)}{p})\frac{P(s_i,t_i)}{P(s_i)}$$

$$= \sum_j \sum_{j',j'>j} \frac{y_{ij}P(s_i,t_i)}{P(s_i)} + \frac{y_{ij'}(1-p)P(s_i,t_i)}{pP(s_i)}$$

We have

$$P(s_i,t_i) = P(s_i|t_i)P(t_i) = P(s_i|j,j^{'})p\frac{p}{1-p}$$

As a result, we have

$$\hat{t_i} = \sum_j \sum_{j',j'>j} \frac{P(s_i|j,j^{'})}{P(s_i)}(\frac{p^2 y_{ij}}{1-p} + py_{ij^{'}})$$

We denote q_i to be the total number of units in the sample s_i for the i^{th} sub-space, which are the units of interest, and we have the following result for $\frac{P(s_i|j,j^{'})}{P(s_i)}$ as shown in Figure 1.

Estimator for AVG for Data Source D: Based on the summation estimator for the i^{th} sub-space in D, the AVG estimator for D using TPS is as follows.

$$\bar{y}_{TPS} = \frac{\sum_{i=1}^m \hat{t_i}}{\sum_{i=1}^m N_i}$$

3.5 Estimating MAX and MIN

Using any of the above proposed algorithm, we obtain a sample of size n as $\mathcal{S} = \{r_1, r_2, \ldots, r_n\}$. The attribute of interest of r_i is denoted as y_i. Then the estimator for the population maximum $\hat{y}_{max}$, and population minimum $\hat{y}_{min}$ are $\hat{y}_{max} = Max_{i=1}^n\{y_i\}$, and $\hat{y}_{max} = Min_{i=1}^n\{y_i\}$.

THEOREM 3.5. (Chebyshev Inequality [25]) *If a random variable X has a finite mean μ and a finite variance σ^2, then for any $\epsilon \geq 0$, we have $Pr[|X-\mu| \geq \epsilon] \leq \frac{\sigma^2}{\epsilon^2}$*

We consider the attribute of interest as a random variable X, using Theorem 3.5, for MAX estimation $\hat{y}_{max}$, we could find a bound on the probability that $|X-\mu| \geq \hat{y}_{max}-\mu$. In Theorem 3.5, we set $\epsilon = \hat{y}_{max} - \mu$, we could get $Pr[X \geq \hat{y}_{max}] \leq \frac{\sigma^2}{(\hat{y}_{max}-\mu)^2}$. In other words, we could say that $\hat{y}_{max}$ is at least the $1-\frac{\sigma^2}{(\hat{y}_{max}-\mu)^2}$ quantile of the values of the attribute of interest. We could build similar bound for the minimum estimator as well.

4. EVALUATION

In this section, we evaluate the sampling algorithms we have developed. First, we evaluate each of the sampling algorithms individually. Second, we compare our new sampling algorithms with HDSampler.

4.1 Data Sets

We have used 8 data sets, including six synthetic datasets and two real data sets.

Synthetic Data sets: The synthetic datasets were generated using MINITAB[2], a statistical software package. We generated six datasets, corresponding to data skew values 0, 1, 3, 5, 7, and 9, and refer to them as `synskew0`, `synskew1`, `synskew3`, `synskew5`,

[2]http://www.Minitab.com

$$\frac{P(s_i|j,j^{'})}{P(s_i)} = \begin{cases} \frac{N_i(N_i-1)}{n_{i1}(n_{i1}-1)} & n_{i2}=0, \\ \frac{N_i(N_i-1)}{(n_{i1}+n_{i2})(n_{i1}+n_{i2}-1)} & n_{i2}>0 \text{ and } q_i >= n_{i2}, \\ \frac{N_i(N_i-1)(n_{i1}+n_{i2}-2)!}{(n_{i1}+n_{i2})!-n_{i2}!\frac{(n_{i1}+n_{i2}-q_i)!}{(n_{i2}-q_i)!}} & n_{i2}>0 \text{ and } q_i < n_{i2} \text{ and } j \text{ or } j^{'} \text{ satisfies neighborhood condition,} \\ \frac{N_i(N_i-1)\{(n_{i1}+n_{i2}-2)!-n_{i2}!\frac{(n_{i1}+n_{i2}-q_i-2)!}{(n_{i2}-q_i)!}\}}{(n_{i1}+n_{i2})!-n_{i2}!\frac{(n_{i1}+n_{i2}-q_i)!}{(n_{i2}-q_i)!}} & n_{i2}>0 \text{ and } q_i < n_{i2} \text{ and neither } j \text{ nor } j^{'} \text{ satisfies the condition.} \end{cases}$$

Figure 1: $\frac{P(s_i|j,j^{'})}{P(s_i)}$ factor for the TPS algorithm

`synskew7`, and `synskew9`, respectively. Each dataset has 1000 data records and 7 attributes (columns). The first attribute is the attribute of interest and the other 6 attributes are used as *input attributes* to query the data. Each of the 6 *input attributes* has a different *neighborhood proximity error (NPE)*, detailed in Section 4.2, with respect to the attribute of interest. The NPEs for the 6 synthetic datasets range from 0.46 to 1.72, and *larger* NPE value indicates *weaker* neighborhood proximity.

US Census Dataset: The Census dataset comprises of the 2002 US Economic Census data on Wholesale Trade Product Lines listed by the Kind of Business. This dataset can be downloaded at American FactFinder[2]. This dataset is referred to as `IBQ`. After removing free text, we have 6 attributes and 24984 data records. We use the `sales` attribute as the attribute of interest, and the `number of establishments` as the input attribute. The skew of the `sales` data in `IQB` is 8, and the NPE value is 0.67, implying that this dataset has good neighborhood proximity.

Yahoo! Auto Data set: The Yahoo! Auto data set, denoted as `Auto`, comprises of the data crawled from a subset of a real-world hidden database at *http://autos.yahoo.com/*. Particularly, we download the data on used Ford cars from any model between 2000 and 2009 and located within 50 miles of a zipcode address. This yields a dataset with 1146 data records. We consider the `price` attribute as the attribute of interest. The data skew of the `price` data in `Auto` is 0.7. The NPE value of `Auto` is 0.31, which also indicates good neighborhood proximity.

4.2 Evaluation Metrics

In our experiments, four metrics are used.

Absolute Error Rate (AER): AER, as defined in Section 2.1, captures the estimation accuracy, with a small AER value indicating higher accuracy.

Sample Size: We consider two types of samples. As stated in Section 3.1.2, immediate neighbors of a sampled data record are free of sampling cost. We denote all these cost-free samples as *associated samples*, and all other cost-involving samples as *direct samples*. The total sample size is the sum of the sizes of the direct sample and the associated sample.

(Projected) Sampling Cost: An initial study shows that the average network transmission cost for the Yahoo! Auto web-site is about 250ms for obtaining one direct sampled unit. We use this measurement to project the actual sampling costs for our sampling algorithms.

Neighborhood Proximity Error: To quantify neighborhood proximity for a particular data source, we define the metric *Neighborhood Proximity Error (NPE)*. This is computed by evaluating NPE of the attribute of interest with respect to the input attributes. We partition the data into clusters based on data record neighborhood as introduced in Section 3.1.2. Then, NPE is computed as a function of the *intra-cluster* variation and *inter-cluster* variation. The details of computing NPE can be found in [27].

[2]http://factfinder.census.gov/

4.3 Performance of HDSampler on Synthetic Skewed Datasets

In this experiment, we use HDSampler to select a SRS of size 50 (5% of total data records) to estimate the average and maximal values of the attribute of interest for each of the 6 synthetic datasets. We examine the AER of the estimates on data with different skews. We observe that with the increase of data skew, the estimation accuracy obtained using HDSampler degraded dramatically. For the data sets with skew of 3, 5, 7 and 9, the AERs of the AVG (MAX) estimates obtained by HDSampler are 40% (20%), 50% (30%), 70% (46%), and 88% (87%), respectively. This confirms that a random sampling approach like HDSampler is not appropriate for skewed data.

We conducted another experiment. Using the same datasets, we measure the least number of direct sampled units needed to achieve an AER of 10% using HDSampler. We observe that for the datasets with skew of 3, 5, 7 and 9, the necessary sample size needed by HDSampler is 22%, 50%, 70% and 90%, respectively, of the total dataset size.

4.4 Evaluation of ANS Algorithm

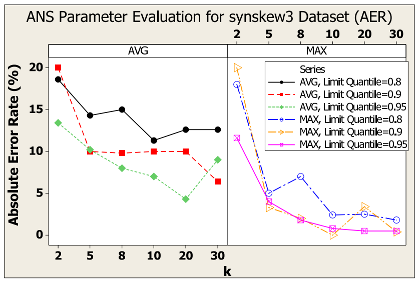
In this section, We evaluate the ANS algorithm, separately considering some variants, with different termination conditions and/or fixed/variable upper limit values.

4.4.1 Parameter Evaluation

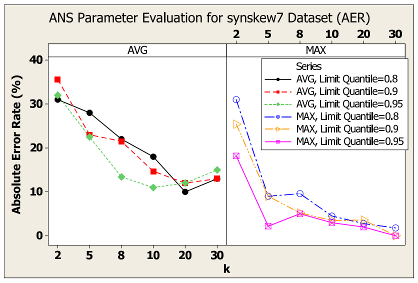
For this experiment, we consider the cost limit to be infinite. This implies that only the first termination rule of ANS is used. In this case, there are two important parameters in the ANS algorithm, which are k, the number of *units of interest* in the initial sample, and the *upper limit value* η.

Impact of Parameters on AER: We examine the effect of the two parameters on the AER of the estimators. For each data set, we vary k from 2 to 30 and η from 80% upper quantile of the data to 95%. We record the AERs for the AVG and MAX estimates. The results for the datasets `synskew3`, `synskew7`, and `IBQ` are shown in Figure 2. Results for other data sets are similar and omitted due to lack of space. In Figure 2, the x-axis is the k value and the y-axis shows the AER values.

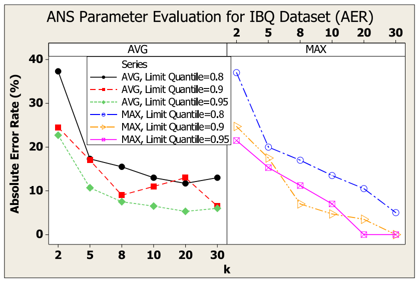
From Figure 2, we have the following observations. For both AVG and MAX estimates, ARE decreases with the increase of k. The greatest decrease in AER happens when k increases from 2 to 8, and for k>8, either the decrease rate in AER slows down, or the AER stays still or even has a small increase. For AVG estimates, for different upper limit values and datasets with different skew, a relatively low AER of 20% can be achieved at $k = 8$. For upper limit value to be 0.95, and $k = 8$, we achieve an AER around 10% for the AVG estimates for all datasets. For MAX estimates, we



(a)

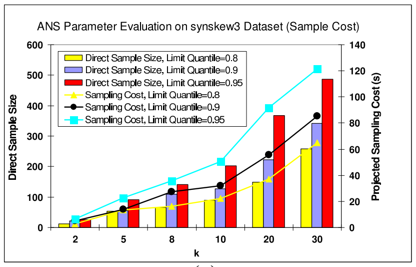


(b)

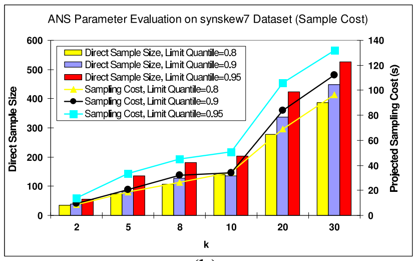


(c)

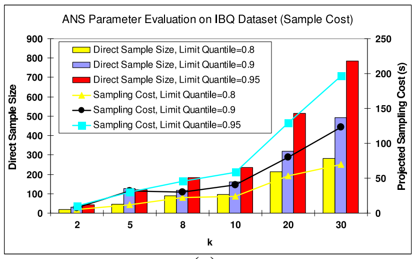
Figure 2: The Effect of k and η on AER - ANS Algorithm: (a) synskew3, (b) synskew7, (c) IBQ (skew=8)



(a)



(b)



(c)

Figure 3: The Effect of k and η on Sampling Cost - ANS Algorithm: (a) synskew3; (b) synskew7; (c) IBQ (skew=8)

have similar observations. This shows that while small k impact the algorithm negatively, very large k are not necessary. This is because for data with a large skew, small k cannot effectively find the data that causes the skew. But, once such data can be identified, accuracy is not improved by larger values of k. Another observation is that when k>8, AER of MAX estimation is lower than the AER of AVG estimation for most cases. The ANS algorithm can identify data with large values, and as MAX estimation depends only on the maximal value, the algorithm is more effective in estimating it. The results also show an expected result that for both AVG and MAX estimation, the AER values become lower with the increase of upper limit quantile values.

Impact of Parameters on Sampling Costs: In this experiment, we vary the value of k and the upper limit quantile. We record the *direct sample size* and *(projected) sampling cost*. The results for `synskew3`, `synskew7` and `IBQ` are shown in Figure 3. In Figure 3, the x-axis is the k value, the left y-axis shows the direct sample size (bar chart), and the right y-axis shows the (projected) sampling cost in term of seconds (line chart).

From Figure 3, we observe that with the increase of k and the upper limit quantile, both the direct sample size and sampling costs increase. For all upper limit quantile cases, the sampling costs for k>10 exceed 40 seconds. Based on the results from the above two experiments, in our following experiments, with the ANS algorithm, we set $k = 8$ and upper limit quantile to be 0.95.

Impact of Data Skew: Besides k and η, data skew is another factor which impacts the performance of the ANS algorithm. In this experiment, we show the AER values of the ANS algorithm on data with different skew. The results are shown in Figure 4. We observe that for both AVG and MAX estimation, AER increases moderately (i.e. the accuracy decreases) with the increase in data skew. When $k = 8$, the AERs of the ANS algorithm are always lower than 19% for different data skew which illustrates the effectiveness of our ANS algorithm.

4.4.2 *Evaluation of Bias*

Lemma 3.3 shows that the bias of the ANS estimator is very small when k is small, such as $k = 8$. Here, we validate this aspect. We use the `IBQ` data set, and consider two cases, corresponding to cost limits of 200 and 500 direct sampled units, respectively. We focus on the AVG estimate in this experiment. For each case, we show the AERs of the AVG estimate and the direct sample sizes with respect to the different values of k. The results are shown in Figure 5. The results for other datasets are similar. In Figure 5, the x-axis is the k values, the y-axis on the left panel shows the AERs of the AVG estimate, and the y-axis on the right panel shows the direct sample sizes.

For the cost limit of 200, on the left panel in Figure 5, AER first decreases to about 7% when $k = 8$, then when k>10, AER increases (bias occurs) moderately. From the corresponding chart of the direct sample size on the right panel in Figure 5, for the cost limit of 200, when $k = 10$, we reached the cost limit of 200. We have similar observation for cost limit of 500 case. AER begins to increase moderately when $k = 30$ (left panel), which corresponds to where the cost limit of 500 is reached (right panel). Figure 5 shows that when the cost limit is reached, bias indeed occurs. However, as long as the k value is relatively small w.r.t. the dataset size, the increased AER value is still reasonable, and we can still achieve good estimation using the ANS algorithm.

4.4.3 *ANS with Post-stratification: Trade off between Sample Size and AER*

In Section 3.2.2, we show that if an upper limit value η brings

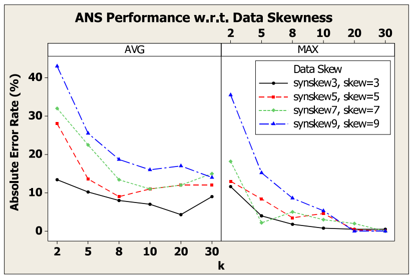


Figure 4: ANS Performance w.r.t. Data Skew, Upper Limit Quantile=0.95

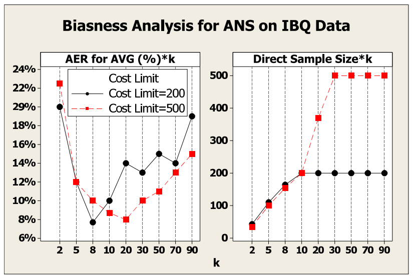


Figure 5: Analysis of Bias for ANS on `IBQ` Data, Cost Limit=200/500

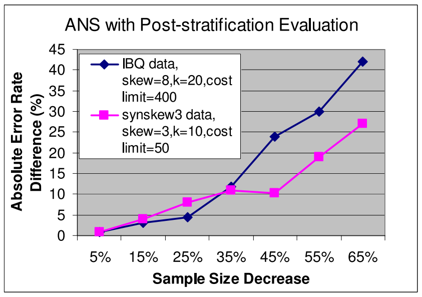


Figure 6: ANS Algorithm With Post-stratification

too many neighbors into the sample, we increase η, resulting in a decreased sample size and lowered sampling cost. There is a trade-off between reducing sample size and estimation accuracy. In this experiment, we evaluate the ANS algorithm with dynamically adjusting η values (post-stratification), using `IBQ` and `synskew3`. The results for other data sets are similar. By adjusting η by different values, we control the percentage of the sampled units saved. Then, for each case, we record the difference of the AER values of the AVG estimate between using post-stratification and without post-stratification (i.e., using a fixed η). The results are shown in Figure 6.

In Figure 6, we observe that the differences of AER between using post-stratification and without post-stratification are always within 10% (small decrease in accuracy), even if we saved a relatively large (35%) percentage in sample size. This shows the effectiveness of ANS with post-stratification. If we significantly reduce the sample size (over 50%), the estimation accuracy is lowered significantly, as we will expect.

4.5 Evaluation of TPS Algorithm

In this subsection, we evaluate the TPS algorithm.

Effect of Initial Selected Sub-space Size: In this experiment, we evaluate the performance of TPS w.r.t. the number of sub-spaces selected. Intuitively, the more sub-spaces selected, the better the estimation accuracy but higher sampling cost. We vary the number of selected sub-spaces, and record the AER values of the AVG and MAX estimates as well as the direct sample sizes. The results for `synskew1`, `synskew3`, `IBQ` and `Auto` are shown in Figure 7. The results for other data sets are similar. In Figure 7, the x-axis is the percentage of selected sub-spaces, the left y-axis is the AER values and the right y-axis shows the direct sample sizes.

From Figure 7, we observe that the AER for both AVG and MAX estimates decreases (better accuracy), and sample size increases, with the increase of selected sub-space size. When selected sub-space size is 30%, we have relatively low AERs (below or around 15%) and small sample size (20 samples, below 1% of the total data set size). This shows that with small sampling cost, TPS could achieve good estimation accuracy. In our following experiments, we set the selected sub-space size to be 30%.

Effect of Data Skew: Here, we measure the AERs of the AVG and MAX estimates using TPS on synthetic datasets with different skew levels. The results are shown in Figure 8. Similar to the ANS algorithm, the AER of both AVG and MAX increases when there is a larger data skew. However, for our chosen selected sub-space size, which is 30%, the AERs are always smaller than 17%. This shows that TPS works well even when there is a significant skew.

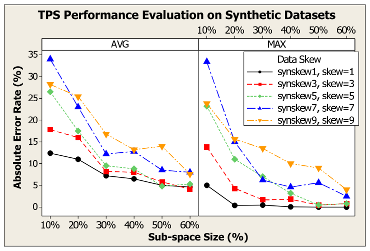


Figure 8: TPS Performance w.r.t. Data Skew

4.6 Comparison of Three Sampling Algorithms

In this section, we compare the performance of our two new sampling algorithms, ANS and TPS, with HDSampler.

4.6.1 Absolute Error Rate Comparison

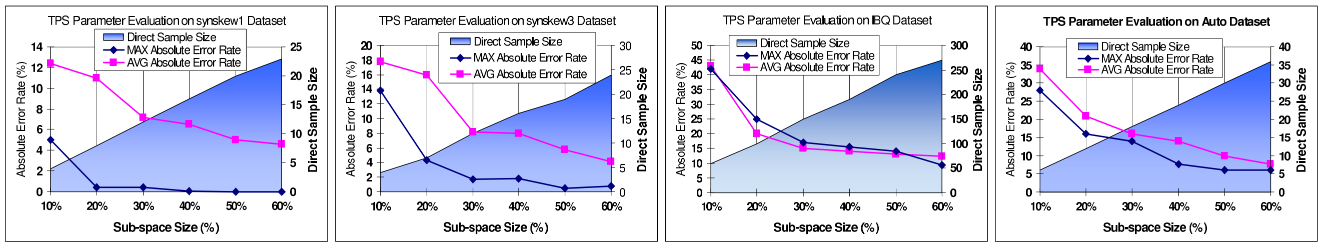
We first compare the estimation accuracy of the three sampling algorithms. We fix the sampling cost, and compare the AER values of both AVG and MAX estimates of the three algorithms. For synthetic dataset, we fixed the sampling cost to be 5% of the dataset size. The result are shown in Figures 9.

From Figure 9, we observe that for small skew (skew $\leq$ 1), the AERs of the three sampling algorithms are comparable. However, with the increase in data skew, the estimation accuracy of HDSampler degraded severely. In these cases, ANS and TPS algorithms outperform HDSampler, with AER being better by a factor of 4 on the average. In particular, the AER values of ANS and TPS are always below 20%. The performance of ANS and TPS are close to each other.

For data sets `IBQ` and `Auto`, since their data skew is fixed, we vary the sample size and compare AER values. The results are shown in Figures 10 and 11.

For `IBQ` (Figure 10), for both MAX and AVG, ANS and TPS outperform HDSampler for all direct sample size cases by a factor of 3 on the average. For `Auto`, ANS and TPS outperform HDSampler for MAX, with average AER being lower by a factor of 3. This shows the advantage of using ANS and TPS for obtaining the large value data, even when the skew is small. For AVG estimation on `Auto`, since the data skew is small, the performance of the three algorithms is comparable. However, as Section 4.6.4 shows, the TPS method has lower sampling cost than HDSampler on small skew data. As a result, we still favor our proposed method.

4.6.2 Estimation Variance Comparison



(a) (b) (c) (d)

Figure 7: The Effect of Selected Sub-space Size - TPS Algorithm: (a)synskew1; (b) synskew3; (c) IBQ (skew=8); (d) Auto (skew=0.7)

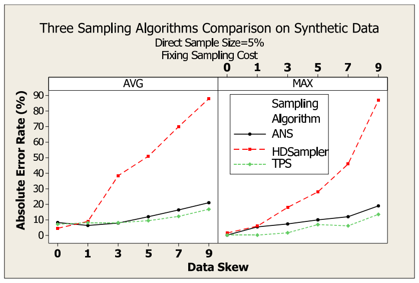


Figure 9: Compare AER for Three Algorithms on Synthetic Data

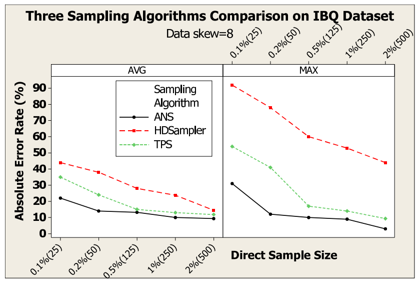


Figure 10: Comparing AER for Three Algorithms on `IBQ` (skew=8)

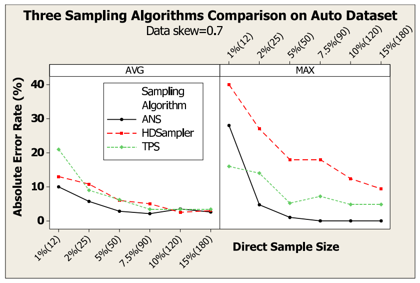


Figure 11: Comparing AER for Three Algorithms on `Auto` (skew=0.7)

Table 2: Estimation Variance Comparison

IBQ			synskew1		
HDSampler	ANS	TPS	HDSampler	ANS	TPS
1.06E+11	2.4E+10	1.09E+10	90	3	22

In this experiment, we compare the estimation variances with the use of these three sampling algorithms. Table 2 shows the comparison results on `IBQ` and `synskew1` datasets. The results on other datasets are similar and omitted due to lack of space. From Table 2, we can observe that our ANS and TPS methods can achieve smaller estimation variance than HDSampler on both large-skewed data (`IBQ`) and small-skewed data (`synskew1`).

4.6.3 The Effect of Neighborhood Proximity: HDSampler vs ANS

In this experiment, we evaluate the effect of neighborhood proximity on estimation accuracy of ANS. Each synthetic dataset has 6 input attributes with different *neighborhood proximity error* (NPE) with respect to the attribute of interest. We execute ANS with cost limit of 100 direct samples on synthetic data sets using different input attributes and examine the AERs. The result on `synskew5` is shown in Figure 12. The results on other synthetic datasets are similar. In Figure 12, the x-axis is the neighborhood proximity errors (larger error means weaker neighborhood proximity), and the y-axis is the AERs. The largest NPE in Figure 12 indicates the input attribute that is generated completely independent from the attribute of interest.

Since HDSampler doesn't explore neighboring units, we have horizontal lines for HDsampler. From Figure 12, we observe that with the increase of NPE, the estimation accuracy of ANS decreases. This is reasonable, because weaker neighborhood proximity implies that larger number of neighboring units selected are not skew causing data. However, the AERs of ANS, even for the largest NPE case, are consistently lower than HDSampler. This shows that even for data sets with weak neighborhood proximity, ANS still outperforms HDSampler.

4.6.4 Sampling Cost Comparison between HDSampler and TPS for Small Skew Data

In this experiment, we show that for small skew data, to achieve the same AER, TPS incurs lower sampling costs than HDSampler. We use `synskew1` and `Auto` (skew=0.7) here. We vary the AERs and record the least number of direct sampled units needed to achieve the same AER for TPS and HDSampler, as well as the total sample size (direct and associated samples) for TPS. The results for the two datasets are shown in Figure 13 and Figure 14.

In Figure 13 and 14, the numbers around each data point show the sampling cost in terms of number of seconds. We have the following observations. First, to achieve a low AER (2% or 5%), TPS requires only one third of the sampling cost of HDSampler. Second, although the sampling cost of TPS is lower than HDSampler, the total number of sampled units TPS uses are larger than HDSampler. This illustrates the key advantage underlying TPS. It should be noted that newer extensions to HDSampler [11] reduce some of the sampling costs, though we are currently unable to compare against these methods. The key distinctive aspect, however, of TPS is that it applies on skewed data, unlike HDSampler and its variants.

4.6.5 Effect of Query Selectivity

In this experiment, we vary the selectivity of queries from 10% to 100% and record the AERs for AVG and MAX estimates of the three algorithms. Due to lack of space, we omit the figure for this experiment. We have the following observations. First, for MAX estimation, the ANS and TPS algorithms outperform HDSampler

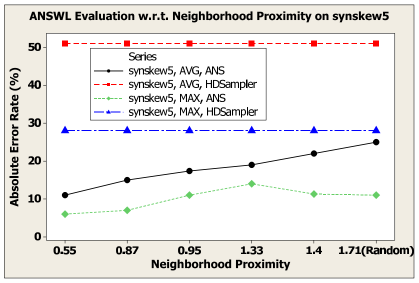


Figure 12: ANS Performance Evaluation w.r.t. Neighborhood Proximity on `synskew5`

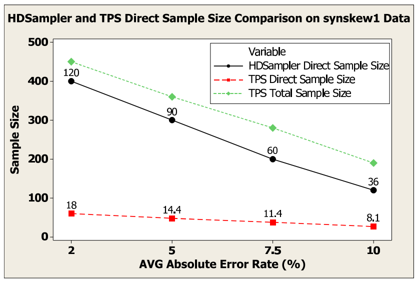


Figure 13: Sampling Cost Comparison for TPS and HDSampler Fixing AER Level on `synskew1`

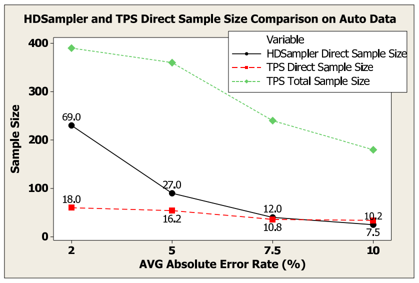


Figure 14: Sampling Cost Comparison for TPS and HDSampler Fixing AER Level on `Auto`(skew=0.7)

for all query selectivity values by a factor of 3 on the average. Second, for AVG estimation, the ANS algorithm consistently outperforms HDSampler by a factor of 2. The TPS algorithm outperforms HDSampler for moderate and large selectivity queries, but is comparable with HDSampler for low selectivity queries. This is because many selected sub-spaces in TPS are filtered out if the query has a low selectivity.

5. DISCUSSION

We now briefly discuss two issues with respect to the applicability of our proposed algorithms.

Customizing Algorithms by Adjusting Parameters: In practice, users may want aggregation results to be estimated with different levels of accuracy based on their needs and domain information. As shown in our experiments in Section 4.4.1 and 4.5, the estimation accuracy and the sample size of the ANS algorithms can be adjusted by parameters k and η. A smaller k and/or η can help decrease the sample size, which results in a coarser estimation, but with a faster response time. Conversely, a larger k and/or η may lead to a larger sample size, and finer estimation, with associated higher response time. Similarly, the TPS algorithm can also be adjusted by changing the *Initial Selected Sub-space Size* parameter.

Applicability to a Real Deep Web Integration System: To enable detailed evaluation, our work has not been based on real deep web data sources. However, our work has been driven by our experiences in developing two real deep web integration systems. In our prior work, we have developed SNPMiner [28]. SNPMiner was developed to integrate several widely used SNP databases and extract relevant information from these databases for end users. SNPMiner is a web-based search tool, and it has a convenient user interface. The system integrates eight biological databases, including general purpose databases and specialized databases for SNPs. SNPMiner includes modules for parsing information from each of these data sources, and a rich query planning support that could enable keyword queries.

Another system we developed integrated four popular web-sites for finding hotels online [21]. This system focused on comparing the hotel room offerings from different web-sites for different cities in the world, with the goal of developing an understanding of the key differences in what they offer.

Neither of the these two systems supported sampling based query processing. Such a support can be provided by extending these systems with the sampling methods presented in this paper. However, there are also several other practical challenges in developing a real deep web integration system. For example, in integrating travel related web-sources, we found that parsing HTML from these data sources can be quite challenging. Moreover, as the format of the web-sources changed frequently, these parsers had to be updated often.

6. RELATED WORK

We now compare our work with existing work on related topics, including online aggregation, approximate query answering, and hidden web sampling.

Online Aggregation: Hellerstein *et al* [13] have developed techniques for online aggregation. Approximate answers for aggregation queries are generated and further refined when all data has been processed. Jermaine *et al* [15] proposed a online aggregation method for the DBO engine. We are considering a distinct problem. First, we focus on *sampling algorithms* to obtain a sample with sampling cost and developing *efficient estimators* to provide accurate estimates. Second, in our case, query is answered only using the sample, but in online aggregation, eventually, all data is processed and an exact answer is obtained. Finally, skewed data is not considered in the above online aggregation efforts.

Approximate Query Answering: To provide better estimates for aggregation queries on skewed data, efforts have focused on preprocessing the data. Histograms [24] and wavelets [4] can be precomputed and used. Chaudhuri *et al* [7, 2, 5, 6] have conducted extensive studies on approximate aggregation queries answering using workload information and biased samples. The approaches include partitioning the database into *fundamental regions* [6, 7] or *groups* [2], based on a given workload of queries. Each fundamental region or group is considered as a stratum. Then, a stratified sample is selected from each stratum based on some property of the stratum. The sample thus chosen is used to answer future queries which are similar to the workload initially provided. To handle skewed data, an *outlier indexing* technique has been developed [5]. Joshi *et al* designed sampling methods and estimators for low selectivity queries [17] and subset-based queries [18]. Wu *et al* [29] have developed a Bayesian method for estimating the extreme values in a dataset based on the learned characteristics of a previous workload. Palmer *et al* [23] have used *density biased sampling* to improve data mining queries on a large dataset. Jermaine *et al* [14, 16] proposed the APA (*approximate pre-aggregation*) and APA+ methods to answer aggregation queries on skewed data using an SRS combined with a small set of statistics about the data.

The above approaches cannot be applied in the deep web scenario. First, to build fundamental regions, stratums, histograms, wavelets, or gathering supplemental data statistics, at least one scan of the entire database is required. Without the access to the full data, as is the case in the deep web, this is not possible. Another distinct aspect of our approach is that we do not assume that we have a workload and future queries are going to be similar to the queries in the workload.

Hidden Web Sampling: The prominent work in this area is by Dasgupta *et al.* [10, 8], who have developed HDSampler. We have extensively compared our work with their method. More recent work from the same authors focuses on estimating the counts [9] and improving efficiency of the sampling process [11]. The work in [9] has not considered challenges associated with skewed data. The work in [11] considered methods to lower down *sample skew*, which is the data skew in the *sample*. It does not solve the problem when the dataset itself is skewed. In comparison, our work focuses on estimating accurate aggregations from *skewed datasets*. In the future, we will like to undertake a detailed comparison with the sampling process described in [11].

Afrati *et al* [1] developed an adaptive sampling algorithm for answering aggregation queries on web-sites with hierarchical structures. They assume that a hierarchical structure partitions a dataset into groups, and focus on adjusting the sample size assigned to each group based on the estimation error in each group. In our problem, no such hierarchical structure assumed. We focus on adaptively *selecting samples*, not *adjusting sample sizes*.

7. CONCLUSIONS

In this paper, we have developed two novel adaptive sampling algorithms, adaptive neighborhood sampling (ANS) and two phase adaptive sampling (TPS), to approximately answer deep web aggregation queries. Without any prior knowledge of hidden data's distribution, our algorithms can obtain accurate estimations with low sampling costs. Our detailed experimental evaluation has shown that 1) For data with a moderate or a large skew, ANS and TPS algorithms obtain an average estimation accuracy around 90% for AVG and 95% for MAX, which is about 4 times better than HDSampler. 2) The TPS algorithm further improves the sampling efficiency. 3) We confirm our theoretical analysis that the bias of the ANS estimator does not impact the estimation performance, and 4) By adaptively adjusting the upper limit value η in the ANS algorithm, we can reduce the sampling costs by one-third, with less than 10% decrease in the estimation accuracy.

8. REFERENCES

[1] F. N. Afrati, P. V. Lekeas, and C. Li. Adaptive-sampling algorithms for answering aggregation queries on Web sites. *Data and Knowledge Engineering*, 64:462–490, 2008.

[2] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *Proceedings of SIGMOD 2003*, pages 539–550, 2003.

[3] M. K. Bergman. The Deep Web: Surfacing Hidden Value. *Journal of Electronic Publishing*, 7, 2001.

[4] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim. Approximate query processing using wavelets. *VLDB Journal*, 10:199–223, 2001.

[5] S. Chaudhuri, G. Das, M. Datar, R. Motwani, and V. Narasayya. Overcoming limitations of sampling for aggregation queries. In *Proceedings of ICDE 1999*, pages 534–542, 1999.

[6] S. Chaudhuri, G. Das, and V. Narasayya. A robust, optimization-based approach for approximate answering of aggregate queries. In *Proceedings of SIGMOD 2001*, pages 295–306, 2001.

[7] S. Chaudhuri, G. Das, and V. Narasayya. Optimized Stratified Sampling for Approximate Query Processing. *ACM Transactions on Database Systems*, 32, 2007.

[8] A. Dasgupta, G. Das, and H. Mannila. Leveraging count information in sampling hidden databases. In *Proceedings of ICDE 2009*, pages 329–340, 2009.

[9] A. Dasgupta, X. Jin, B. Jewell, N. Zhang, and G. Das. Unbiased estimation of size and other aggregates over hidden web databases. In *Proceedings of SIGMOD 2010*, pages 855–866, 2010.

[10] A. Dasgupta, N. Zhang, and G. Das. A random walk approach to sampling hidden databases. In *Proceedings of SIGMOD 2007*, pages 629–640, 2007.

[11] A. Dasgupta, N. Zhang, and G. Das. Turbo-charging hidden database samplers with overflowing queries and skew reduction. In *Proceedings of Conference on Extending Database Technology (EDBT)*, 2010.

[12] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the deep web. *Communications of the ACM*, 50:94–101, 2007.

[13] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *Proceedings of SIGMOD 1997*, 1997.

[14] C. Jermaine. Robust estimation with sampling and approximate pre-aggregation. In *Proceedings of VLDB 2003*, pages 886–897, 2003.

[15] C. Jermaine, S. Arumugam, A. Pol, and A. Dobra. Scalable approximate query processing with the dbo engine. In *Proceedings of SIGMOD 2007*, pages 725–736, 2007.

[16] R. Jin, L. Glimcher, C. Jermaine, and G. Agrawal. New sampling-based estimators for olap queries. In *Proceedings of ICDE 2006*, page 18, 2006.

[17] S. Joshi and C. Jermaine. Robust stratified sampling plans for low selectivity queries. In *Proceedings of ICDE 2008*, pages 199–208, 2008.

[18] S. Joshi and C. Jermaine. Sampling-Based Estimators for Subset-Based Queries. *The VLDB Journal*, 18:181–202, 2009.

[19] F. Lan. Sequential adaptive designs to estimate abundance in rare populations., 1999.

[20] E. L. Lehmann. *Theory of Point Estimation*. Chapman and Hall, 1983.

[21] T. Liu, F. Wang, J. Zhu, and G. Agrawal. Differential Analysis on Deep Web Data Sources. In *Proceedings of the Workshop on Mining Multiple Information Sources (MMIS), held in conjunction with International Conference on Data Mining (ICDM)*, 2010.

[22] J. Madhavan, L. Afanasiev, L. Antova, and A. Y. Halevy. Harnessing the deep web: Present and future. In *Proceedings of CIDR 2009*, 2009.

[23] C. R. Palmer and C. Faloutsos. Density biased sampling: An improved method for data mining and clustering. In *Proceedings of SIGMOD 2000*, pages 82–92, 2000.

[24] V. Poosala and V. Ganti. Fast approximate query answering using precomputed statistics. In *Proceedings of ICDE 1999*, page 252, 1999.

[25] J. A. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, 1995.

[26] S. K. Thompson and G. A. F. Seber. *Adaptive Sampling*. Wiley, 1996.

[27] F. Wang. *SEEDEEP: A System for Exploring and Querying Deep Web Data Sources*. PhD thesis, Ohio State University, 2010.

[28] F. Wang, R. Jin, G. Agrawal, and H. Piontkivska. SNPMiner: A Domain Specific Integration Tool. In *Proceedings of IEEE Syposium on Bioinformatics and Bioengineering*, Oct. 2007.

[29] M. Wu and C. Jermaine. A bayesian method for guessing the extreme values in a data set. In *Proceedings of VLDB 2007*, pages 471–482, 2007.