TRCS01-19

Query Estimation By Adaptive Sampling

Yi-Leh Wu Divyakant Agrawal Amr El Abbadi

Department of Computer Science
University of California, Santa Barbara, CA 93106
Email:{ywu, agrawal, amr}@cs.ucsb.edu

Abstract

The ability to provide accurate and efficient result estimations of user queries is very important for the query optimizer in database systems. In this paper, we show that the traditional estimation techniques with data reduction points of view do not produce satisfiable estimation results if the query patterns are dynamically changing. We further show that to reduce query estimation error, instead of accurately capturing the data distribution, it is more effective to capture the user query patterns. In this paper, we propose query estimation techniques that can adapt to user query patterns for more accurate estimates of the size of selection or range queries over databases. [1]

1 Introduction

Query size estimation plays an important role in the query optimizer of a database management system. For example, depending on the size of the intermediate results, a particular query execution plan may be favored over another. Query estimates can also be used by the database system to give user feedback on the possible result sizes and the running time needed before queries are executed. A common approach in relational databases is to store, for each attribute, summary information that can be used to estimate the resulting size of a query. Histogram techniques [Poo97, APR99, PG99, PIHS96, PI97, JKM+98] are widely studied to serve this purpose of summarizing information. Ideal histograms should consist of buckets such that the enclosed domain region for each bucket has near uniform tuple density distribution to the reduce estimation error of selectivity as suggested by [APR99, PIHS96]. However, several problems arise with the construction of ideal histograms. First, most histograms are static in the sense that once the histogram buckets are built they remain fixed regardless of changes in data distribution caused by user updates. [GMP97, DIR00] propose reconfiguration techniques for single attribute histograms to remedy this problem. Second, most histogram techniques construct their buckets only with respect to the data distribution without considering how the histograms will be used in estimating user query results. Ignoring user query patterns assumes that user queries are randomly distributed throughout the domain, which is rarely the case in practice.

To remedy the second problem, [CR94] uses query feedbacks from the query processor to adjust the model functions of the data distributions. However, for real datasets it it rare that the underlying data

[1]This work was partially supported by NSF grants EIA-9818320, IIS-98-17432, EIA-9986057 and IIS-99-70700.

distributions can be described by a linear combination of known and smooth probability distributions. Other techniques have been proposed to build histograms without examining data distributions, by just examining the query result feedback from the query processor [AC99, BCG01]. Buckets are allocated where needed most as indicated by the workloads. Chaudhuri et. al. propose an optimization based sampling approach to minimize the error in answering queries for a given workload [CDN01]. However, their assumption that the given training workloads are *similar* to the actual workloads may not hold in some situations because user query patterns may change over time. Ganti et al. [GLR00] propose the *Icicle* approach, which is a weighted tuple sampling technique that can tune itself to dynamic workloads.

In this paper, we propose novel techniques that can adapt to user query patterns for more accurate size estimates of range queries in databases. Our proposed techniques differ from previous work in several aspects. First, unlike [AC99, BCG01] which ignore data distributions and build the histogram solely upon user query feedback, we try to accurately capture the data distribution then fine tune our estimation with query feedback. Second, our proposed methods do not assume that the workload distribution is relatively static as in [CDN01] or that the data distribution is relatively static as in [GLR00]. Third and most importantly, unlike the histogram techniques [AC99, BCG01] (based on partitioning the frequency distribution function) or the sampling techniques [CDN01, GLR00] (based on tuple sampling of the relation), our estimation techniques are based on sampling of the cumulative distribution function (cdf). We show that, by using samples on the cdf, the proposed techniques are more accurate and less expensive to maintain compared to previous work.

This paper is organized as follows. In Section 2 we introduce our terminology and discuss how to apply different estimation techniques to estimate range queries. In Section 3, we propose techniques that can adapt the sample sets of the estimator to user query patterns and thus can provide more accurate range query estimation. Experiments and results are presented in Section 4. Section 5 concludes this paper.

2 Background

In this section, we begin by defining the query result estimation problem. For illustration purpose, we limit our discussions to range query estimation on a single attribute domain. Note however, that the techniques we discuss throughout this paper are not limited to single attribute domains (one dimensional space) but can also be applicable to multiple attribute domains. We then discuss the categories of current estimation techniques and the differences among them. Finally, we discuss related work for the adaptive query estimation problem.

2.1 Preliminaries

Given a relation R with an attribute X, we are interested in estimating the size of a range query on attribute X. We adopt the notation develop in [PIHS96, MVW98] where information about the attribute domain is summarized and used in estimating the size of the query. We assume that the discrete domain D of attribute X is $\{d_0, d_1, d_2, ..., d_{N-1}\}$ which contains all the possible values of X. A range query is specified using a range predicate of the form $d_a \leq X \leq d_b$, where d_a and d_b are constant values in the discrete domain D. To calculate the size of this query, without actually executing it, a frequency distribution function is needed. The *frequency* f_i of value d_i for attribute X is the number of tuples $t \in R$ with $t.X = d_i$. If no tuples t in R have $t.X = d_i$, then $f_i = 0$; i.e., zero frequency. The *frequency distribution function* (fdf) is an ordered set of value pairs $fdf = \{\langle d_0, f_0\rangle, \langle d_1, f_1\rangle, \langle d_2, f_2\rangle, ..., \langle d_{N-1}, f_{N-1}\rangle\}$. Given the fdf, the size of a range query $q(d_a \leq X \leq d_b)$ can be calculated as the number of tuples $t \in R$ with $d_a \leq t.X \leq d_b$, which is equal to $\sum_{i=a}^{b} f_i$.

The main challenge in query size estimation is to use a limited amount of storage to approximate the fdf. This approximation is then used to estimate the number of tuples that are in the range specified by the query. In the following sections we categorize existing estimation techniques and highlight the differences among them.

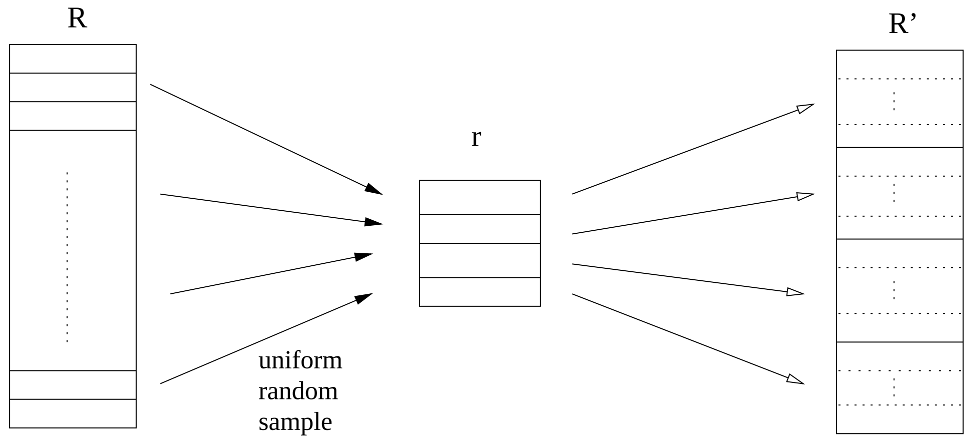
2.2 Categories of the Estimation Techniques

We categorize different estimation techniques by the information they try to summarize. To describe the distribution of attribute X in relation R, we can summarize the relational table R itself, summarize the fdf of attribute X, or summarize the cdf of the attribute X. Note that, with respect to attribute X, the set of tuples in relation R and the fdf and cdf of X are equivalent; i.e., they carry the same information and one can be derived from the other.

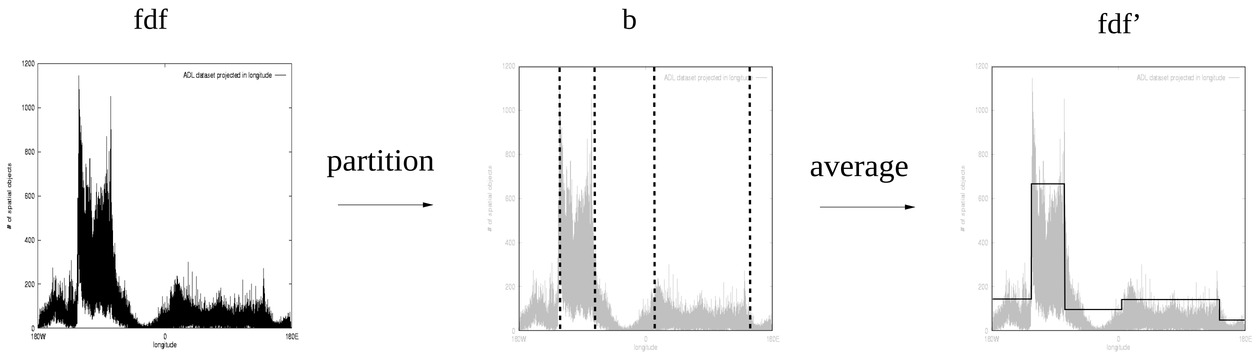
The first category of the estimation techniques is tuple sampling (e.g., [HS92, HS95, GM98, HHW97, HH99, LNS90]). The tuple sampling technique summarizes a relation R by taking uniform samples from the tuples in R. As shown in Figure 1(a), the summarized version of relation R is the sample set r. Intuitively, when a query is posed to the estimator, the estimator logically materializes the estimated relation R' from the sample set r and processes the query on relation R' to get an estimation. Let the number of tuple in R be $|R|$, the number of sample tuples in r be $|r|$ and $|R|/|r| = n$; i.e., the sample set r is $1/n$ the size of relation R. When relation R', which is the approximation of R, is materialized, each tuple in sample set r is replicated n times in R'.

Instead of summarizing the relation R directly, histogram based techniques typically summarize the fdf. To summarize the fdf, as shown in Figure 1(b), bucket boundaries are decided in the attribute domain according to certain criteria. For example, the equi-width histogram evenly partitions the attribute domain and the MaxDiff histogram [PIHS96] puts the bucket boundaries on the attribute values that have the maximum difference with their neighbors. Other proposed histogram techniques try to minimize the sum of variance in each bucket and claim optimality under different classes of user queries [IP95, PI97, JKM$^+$98, APR99]. After the bucket boundaries are chosen information regarding each bucket are recorded, e.g., the bucket boundaries and mean frequency of the attribute domain bounded by the bucket. As is the case in tuple sampling, when a query is posed to the estimator, the estimator logically materializes the estimated fdf, namely the fdf' in Figure 1(b), from the bucket set b and processes the query on fdf' to get an estimation. Note that in fdf' the frequency values in each bucket are replaced by the mean frequency of that bucket. Intuitively, in order to reduce the estimation errors, the variance in each individual bucket should be minimized [APR99]. However, as in our example fdf in Figure 1(b), the fdf is a fast changing function and even with the optimal bucket selection, if it can be decided, the variances in each bucket tend to be high.

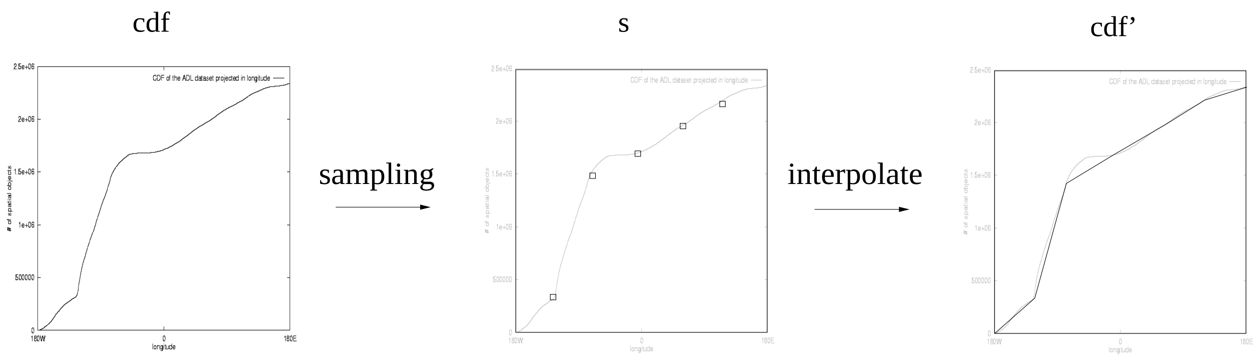
In [WAE01], we proposed the *golden estimator* technique which is based on the cumulative frequency distribution (cdf). The cdf is the ordered set of value pairs $\{\langle d_0, c_0\rangle, \langle d_1, c_1\rangle, \langle d_2, c_2\rangle, ..., \langle d_{N-1}, c_{N-1}\rangle\}$ where c_i is the cumulative frequency of value d_i, which is the number of tuples $t \in R$ with $t.X \leq d_i$; i.e., $c_i = \sum_{j=1}^{i} f_j$. For a range query $q(d_a \leq X \leq d_b)$, the cdf can be used to compute the number of tuples as $c_b - c_a$.



(a) Tuple sample technique



(b) Histogram technique



(c) Golden estimator technique

Figure 1: The differences among estimation technique

The cumulative frequency distribution can be easily obtained from the frequency distribution function. The intuition of the golden estimator technique is that instead of approximating the fdf, which is usually difficult, we approximate the cdf. As shown in Figure 1(c), the cdf is a relatively smooth function compared to the fdf (to be more exact, a non-decreasing function) in a small scale and intuitively easier to approximate. To approximate the cdf, the golden estimator takes random samples from the cdf and acquires the sample set s as shown in the middle of Figure 1(c). As is the case in the above techniques, when a query is posed to the estimator, the estimator logically materializes the estimated cdf, namely the cdf' in Figure 1(c), from the sample set s and processes the query on cdf' to get an estimation. Interpolation is used between sample pairs to approximate the cdf.

However, these three estimation techniques only focus on how to accurately capture data distributions. Moreover, most of the estimation techniques assume that the data are relatively static and the query workloads are unknown and usually ignored. In fact, most experimental evaluations use a uniform distribution of query workloads. In the following section, we discuss a new class of estimation techniques that eliminate these static assumptions to some degrees, and can adapt to query workloads.

2.3 Related Work on Adaptive Query Estimation

Recently, proposals have been made to extend existing query estimation techniques to accommodate the dynamic natures of data distributions and/or the query patterns [GMP97, DIR00, CR94, AC99, BCG01, CDN01, GLR00]. In this section, we briefly review some of the techniques.

For the category of tuple sampling that summarize the relation R, Ganti et al. [GLR00] propose *Icicle*, a new class of tuple sampling technique that can tune itself to dynamic workloads. The intuition of their technique is that the probability of a tuple included in an Icicle is proportional to the importance of that tuple for answering queries; i.e., an Icicle consist of more tuples from a subset of the relation that is used frequently to answer queries in the workload.

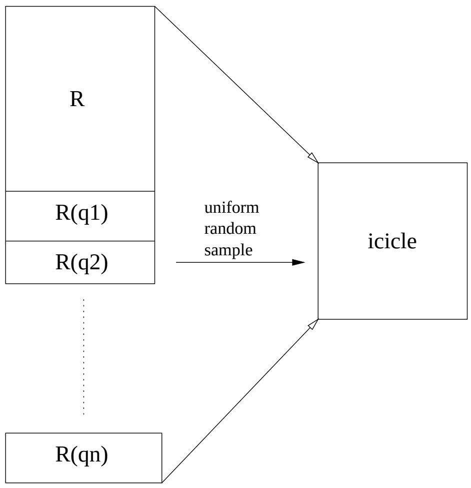


Figure 2: Logical Relation of an Icicle

In Figure 2, R is the base relation and a workload Q consists of n queries, $\{q_1, q_2, ..., q_n\}$. $R(q_i)$ is a subset of tuples in R that constitutes the answer set of query Q_i. A logical view of an Icicle is a uniform random sample of a multiset of tuples L, which is the union of tuples in R and all the set of tuples that were required to answer queries in the workload, which is $R(Q)$. [GLR00] shows that if an average query q_i is focused with respect to workload Q, then the answer of q_i can be answered more accurately than the original tuple sampling technique.

For the category of histogram techniques that summarize the fdf, Bruno et al. [BCG01] propose the *STHole*, a "workload-aware" histogram technique that can tune itself to user query workloads. STHole histograms allow nested histogram buckets to capture regions with near uniform data distributes. To construct a STHole histogram, one can start with an empty histogram or a single bucket histogram that covers the entire domain. The actual result of each query q in the workload is intercepted from the query processor and is used to refine the STHole histogram. By computing the overlapping regions of query q with each histogram bucket, one can refine the histogram by "drilling holes" or zooming into the buckets that cover the query region. If the total number of buckets exceeds the fixed storage constraint, one can merge buckets with similar characteristics, which results in the smallest penalty on the query estimation accuracy.

Instead of the tuple sampling technique and its variations (e.g., the *Icicle* [GLR00]) and the modified histogram techniques (e.g., the *STHole* [BCG01]) discussed above, we address the adaptive query estimation problem by proposing techniques that sample the user query patterns and use the cdf of the data distribution for more accurate estimate of the size of selections or range queries in databases.

3 Adaptive Sampling

In this section we first demonstrate the performance of estimation technique that focuses on data reduction in estimating queries on real dataset. We also demonstrate the impact of user query patterns on the estimation accuracy. We show that capturing the query distribution is as important as capturing the data distribution if we want to reduce the average estimation errors. Several techniques that can capture user query distributions are proposed later in this section.

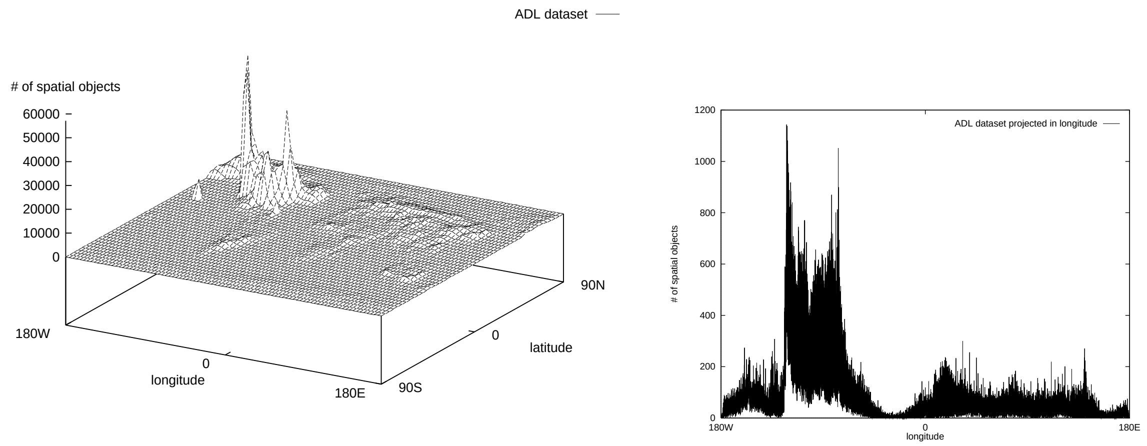
3.1 User Query Patterns

We first review one of the estimation technique discussed in Section 2, namely the golden estimator technique [WAE01], which can accurately capture the underlying data distribution. However, we also show by experiments that, if we want to reduce the estimation errors, capturing the query distribution is more important than capturing the data distribution.

All queries used throughout this section are *prefix* queries (also know as Type A query as defined in [VWI98]). In short, a prefix query q on relation R with attributes X is defined as

$$q_p(x \leq d_m) = \sum_{i=0}^{m} f_i \tag{1}$$

where f_i is the frequency of tuples as described in Section 2.1. The reason for using prefix queries is that every range query can be translated into a linear combination of prefix queries [WAE01]. Let the actual result of range query q be s, the absolute difference between the actual result size s and s', $|s - s'|$ is referred to as the *absolute error*. The *relative error* E_{rel}, which evaluates the error as related to the result size is defined as $E_{rel}(q) = \frac{|s-s'|}{s}$. The error measurement is the *average relative error* for the query set Q; i.e.; if

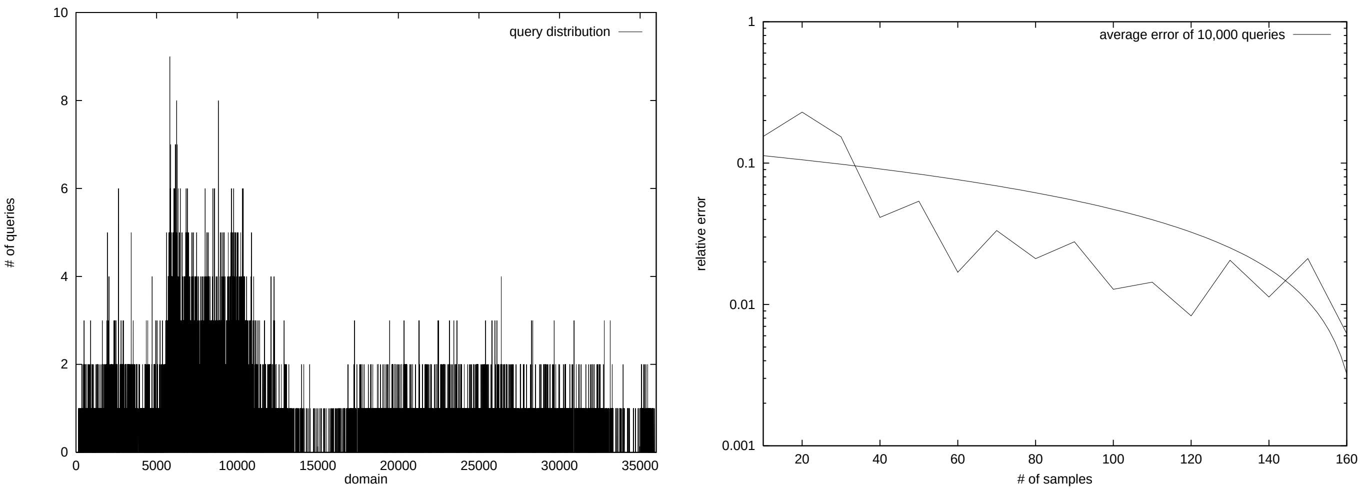


(a) (b)

Figure 3: The ADL Dataset

we assume $Q = \{q_1, q_2, ..., q_m\}$, then the average relative error is defined as

$$\overline{E}_{rel}(Q) = \frac{\sum_{i=1}^{m} E_{rel}(q_i)}{m} \tag{2}$$

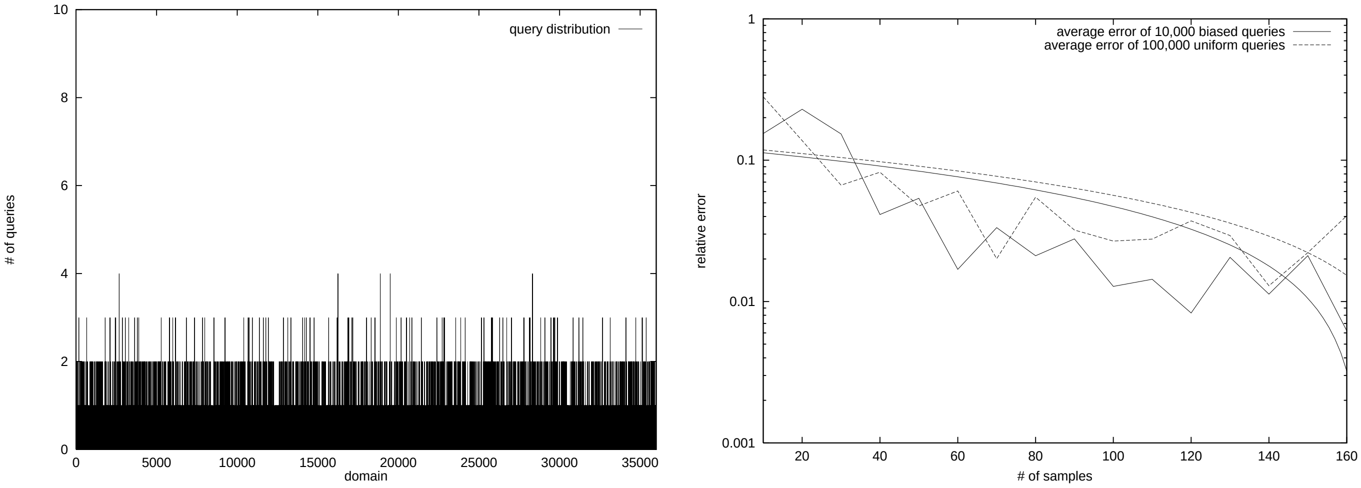


(a) (b)

Figure 4: Experiment results of the ADL dataset with biased distributed query set

The dataset we use throughout this paper is a real dataset taken from the Alexandria Digital Library (ADL) [Smi96] at the University of California at Santa Barbara. The dataset contains 2,339,771 digital map objects. Figure 3.(a) depicts the distribution of the centers of each ADL map object. In Figure 3, the x axis is the longitude, y axis is the latitude, and the z axis is the number of map objects centered in these coordinates. Each cell in the grid represents an area of 4 degree longitude by 2 degree latitude. Note that the peaks in Figure 3.(a) generally fall in the area with range between [40W:130W] and [0N:60N], which roughly corresponds to North America. A big portion of the ADL collection is composed of spatial objects covering North America. Figure3.(b) is the one dimensional ADL data projection on the x-axis; i.e., data distribution on the longitude attribute.

Since we do not have a priori knowledge of the distribution of user queries, we use a form of biased query sets. The position of each prefix query q is biased with respect to the distribution of the data in R, as discussed in [APR99]; i.e., the query distribution is the same as the data distribution. Figure 4.(a) shows a sample query set with 10,000 prefix queries points. Figure 4.(b) shows the average relative error when applying the golden estimator techniques with different storage space requirements; i.e., 10 buckets to 160 buckets. Figure 4.(b) also shows the regression result of the errors. Note that since the y-axis is in log scale, the regression line is therefore a curve line.



(a) (b)

Figure 5: Experiment results of the ADL dataset with uniformly distributed query set

s discussed above, we are interested in knowing when there is no a priori knowledge of the user query distribution, how will these distributions affect the accuracy of estimates. For example, Figure 5.(a) shows another sample query set with 10,000 prefix queries points that follow a uniform distribution with respect to the data domain. We then estimated the query set size using the golden estimator approach with different storage settings. The dashed lines in Figure 5.(b) show the estimation errors and their regression result. The solid lines in Figure 5.(b) are the average relative errors (and the regression result) when applying the golden estimator techniques on the biased query set as depicted in Figure 4.(b). As depicted in Figure 5.(b), the result shows that the average relative errors are generally greater when compared to the results in Figure 4.(b) for biased query set. This suggests that not only the data distributions but also the distributions of query sets can affect the accuracy of estimation. In the following sections, we propose techniques to capture the user query distributions accurately and later show the effectiveness of the proposed techniques on decreasing estimation errors.

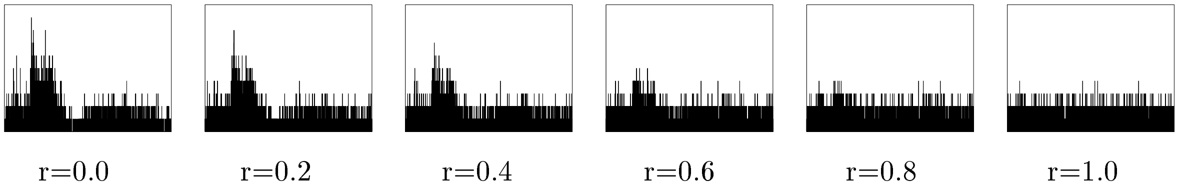
3.2 Estimation Technique with Training Workloads

```
PROCEDURE Static_Sampling_Estimator (TrainingWorkload, cdf_data)
BEGIN
    querySampleSet = NULL;
    finalSampleSet = NULL;
    compute cdf_query of TrainingWorkload;
    invoke the GoldenEstimator with cdf_query and put the result in querySampleSet;
    FOR EACH sample ∈ querySampleSet
        add ⟨samplePosition, cdf_data(samplePosition)⟩ to finalSampleSet;
    END; (* FOR EACH *)
    RETURN finalSampleSet;
END; (* Static_Golden_Estimator *)
```

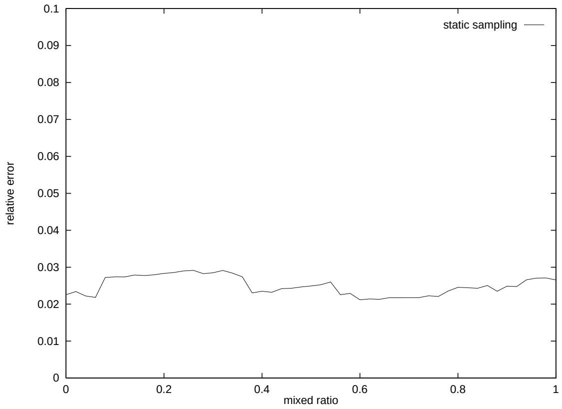
Figure 6: The Algorithm for Training Workload Sampling

If the query distribution is known a priori or training workloads are available then we can adapt the golden estimator technique [WAE01] to capture the query distributions. Instead of picking sample points with respect to the data distribution, we find sample points using the query distribution. First we compute the cdf of the input query set cdf_{query}. Next we take a uniform sample with respect to the domain of cdf_{query} and get the query sample set $querySampleSet$. The probability of a query $q_i(d_i)$ being represented in $querySampleSet$ is proportional to the total number of queries $q_j(d_j) \in Q$, where $d_i = d_j$, and $i \neq j$. For each sample points we store the cdf, which is the cumulative data frequency in cdf_{data}, of that query point and use the sample set for estimating user queries. The algorithm of the training workload sampling algorithm is shown in Figure 6.

We now conduct experiments to show that the training workload sampling algorithm is a feasible technique for adapting query patterns and yields good estimates. We start with experiments on one dimensional dataset. The one dimensional dataset we used is the ADL dataset as discussed in Section 3.1 with the center of each spatial object projected on the longitude attribute with 1/100 degree precision; i.e., 36,000 cells along longitude direction with each cell 1/100 degree. The one dimensional dataset is shown in Figure 3.(b). Figure 7.(a) shows a variety of query sets with distributions mixed from uniform distribution and ADL data distribution in longitude dimension. We mix the ADL data distribution and uniform random distribution with different mix ratio of the ADL data distribution and uniform distributions; i.e., from left to right in 0%, 20%, 40%, 60%, 80% and 100% samples follow the uniform random distribution and 100%, 80%, 60%, 40%, 20%, and 0% samples follow the ADL data distribution, respectively. For example, the second from the left figure of Figure 7.(a) depicts a sample set with mix ration $r = 0.2$; i.e., 20% samples that follow the uniform random distribution and 80% samples that follow the ADL data distribution. Figure 7.(b) depicts the average relative errors when applying the static sampling estimator with 100 static samples on each query set, each contains 100,000 prefix queries. Note that although the experiment on the proposed technique uses only prefix queries, but if the input queries are Type B, we can simply replace each Type B query (e.g., $Q_{TypeB}(i \leq x \leq j)$) with two Type A queries (e.g., $Q_{TypeB}(i \leq x \leq j) = Q_{TypeA}(x \leq j) - Q_{TypeA}(x \leq i)$) and then perform the static sampling algorithm as discussed above to capture query patterns. The experiment result suggests that the static sampling algorithm yields stable estimation errors irrespective to data



(a) query sets with mixed distributions



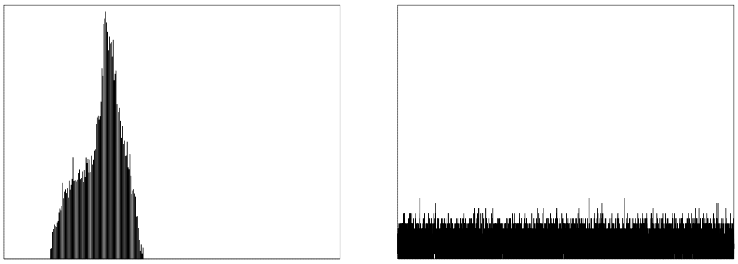
(b) estimation error for different query distributions

Figure 7: Experiment Results of the Static Sampling Algorithm Using One Dimensional Dataset

distributions or query distributions in one dimension.

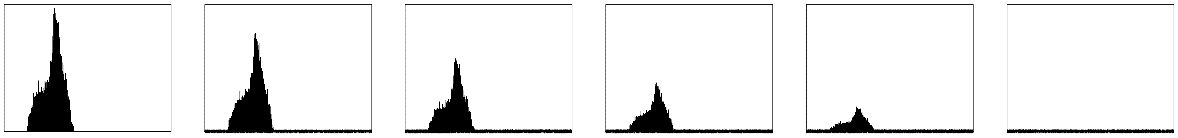
We present another experiment to show how the query distribution affects the estimation error. Figure 8(a) shows the two query sets used in this experiment. The first query set has clustered distribution; i.e., all the queries in the set are within a limited range of the domain, as shown in the left in Figure 8(a). The other query set has queries that are uniformly distributed in the entire domain, as shown in the right of Figure 8(a). Figure 8(b) shows the sample sets used by the estimator to generate estimates. We mix the clustered distribution and uniform random distribution with different percentage of samples follow the two distributions as discussed earlier in this section. For example, the second from the left figure of Figure 8.(b) depicts a sample set with mix ration $r = 0.2$; i.e., 20% samples that follow the uniform random distribution and 80% samples that follow the clustered distribution. Figure 8.(c) shows the estimation errors with different combinations of query and sample sets. The solid line shows the average relative error of 10,000 queries that follow the clustered query distribution as depicted in the left of Figure 8(a), with 200 data samples that follow different distributions as depicted in Figure 8.(b). The dash line in Figure 8.(c) shows the average relative error of 10,000 queries that follow the uniform query distribution as depicted in the right of Figure 8(a), with 200 data samples that follow different distributions.

First note that the average relative errors of uniform random queries are larger than the clustered queries with any sample distributions. Second note that when the query distribution is uniformly distributed, a uniformly distributed sample set, which is with mix ration $r = 1.0$ on the right of x-axis, produces the least error compared to other sample distributions. However, if the queries are clustered, then a clustered sample set, which is with mix ratio $r = 0.0$ on the left of x-axis, results in the least error. Note that when the sample distribution deviates from the query distribution, the error increases. This can be observed from the trend of the results of clustered queries on the bottom of Figure 8.(c), as the sample set gets more uniformly



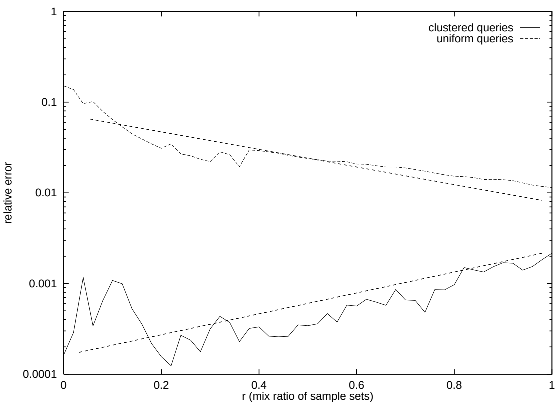
clustered queries   uniform queries

(a) query distributions



r=0.0   r=0.2   r=0.4   r=0.6   r=0.8   r=1.0

(b) mixed sample distributions



(c) relative errors

Figure 8: Co-relation between query and sample distribution

distributed toward the right of x-axis, the error gradually increases. The same trend can be observed from the results of the uniformly distributed queries, which is on the top of Figure 8.(c). As the sample distributions deviate from the query distribution toward the left of x-axis (samples get more clustered), the estimation error also increases. The result suggests a co-relation between the query distribution and the sample distribution. Third, as discussed above, the best sample set that produces the least estimation error with respect to the uniformly distributed query set is the uniformly distributed sample set (the right most one along the x-axis), which results an average relative error of 1 to 2%. However, the best sample set that produces the least estimation error with respect to the clustered distributed query set is the clustered distributed sample (the left most one along the x-axis), which result an average relative error ranges from 0.01% to 0.1%. The least estimation error produced by evaluating the clustered query set is an order of magnitude smaller than the one with uniformly distributed query set. This is because in the best case, the sample set used to evaluate the clustered query set is also clustered; i.e., the 200 samples are put in the same limited range as the queries. Hence with more samples in this region we can answer queries in this same region more accurately. This result suggests that the query distributions can greatly affect the estimation accuracy. Moreover, this results suggest that capturing the query distribution is important when the main objective is to reduce the average estimation errors.

The technique proposed in this section assumes that the query patterns are known a priori and the corresponding training workloads are available. In the following section, we generalize the problem to capture unknown user query patterns dynamically.

3.3 Dynamic Sampling the User Queries

```
PROCEDURE Dynamic_Sampling_Estimator (query)
BEGIN
    STATIC sampleSet is initialized with random samples according to data distribution;
    STATIC noQueriesProcessed = 0; (* the number of queries processed so far *)
    noQueriesProcessed++;
    skip = pick a random number between 0 and noQueriesProcessed;
    IF skip ≤ |sampleSet| THEN (* try add query to sampleSet *)
        transform query into prefix range query
        IF query ∉ sampleSet THEN (* add if no duplicate *)
            randomly kick one sample out of sampleSet
            add ⟨query, cdf_q⟩ to sampleSet;
        END; (* IF query ∈ sampleSet*)
    END; (* IF skip ≤ |sampleSet| *)
    RETURN sampleSet;
END; (* Dynamic_Sampling_Estimator *)

PROCEDURE Query_Processing (queryStream)
BEGIN
    WHILE queryStream is open
        query = get next query q from queryStream;
        Query_Processor(q);
        Dynamic_Sampling_Estimator(q);
    END; (* WHILE *)
END; (* Query_Processing *)
```

Figure 9: The Algorithm for Dynamic Sampling

In Section 3.2 we proposed a static sampling algorithm that captures the user query distribution with a representative set of user queries. However, in real world applications, a priori information of user query patterns may not always be available. Furthermore, computing and maintaining the cdf of an input query set is expensive in terms of computation time and storage space. Finally, the user query patterns may change over time and therefore the sampled data may need to be adapted dynamically.

In this section, we propose a dynamic sampling algorithm to overcome the above drawbacks of the static sampling algorithm. The algorithm works as follows. Initially, without a priori information of query patterns, we randomly generate a sample set according to the data distribution with the hope that the user queries follow the same distribution. Over time, we randomly replace a sample query in the sample set by an actual query point with the actual query result. The actual query result can be computed by the estimator or it can be piggybacked from the query processor.

The technique we employed to gather sample queries is the *reservoir sampling* algorithms [Vit85], which generates a uniform sample set from the input query stream. The basic ideas of the reservoir algorithms are to select a sample of size $\geq n$, from which a true random sample of size n can be generated from the input sources without a priori knowledge of the size of the input sources. Our proposed technique relied on this property because the input query streams are continuous and we do not know the size of the input as in the case of training workload algorithm discussed in Section 3.2. In summary, the algorithm takes an input stream of user queries and generates a uniform sample set of the input queries given so far. Yet another consideration, since the storage of the sample set is usually small and fixed, we want the sample set to contain as much information as possible. As a result, if two or more queries are identical, storing identical queries (which have the same results) into the sample set does not increase the estimation accuracy . In other words, by storing duplicate queries, we decrease the utilization of the limited storage space. For this reason we discard duplicate queries in the sample set and hence the resulting set may not be a true uniform sample of the input query stream.

We now discuss how to process the query result feedback. In previous discussion we assumed that the actual query result can be computed by the estimator or it can be piggybacked from the query processor. However, user queries in practice are general range queries (Type B queries in [VWI98]) whereas each sample used in the dynamic sampling estimator is a prefix query (Type A) result. We need to accommodate general range query result feedback from the query processor into prefix query result samples; i.e., to tune the interpolation on cdf from range query feedback.

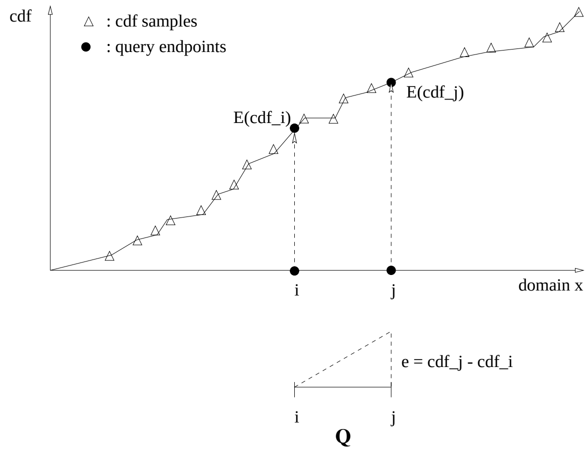
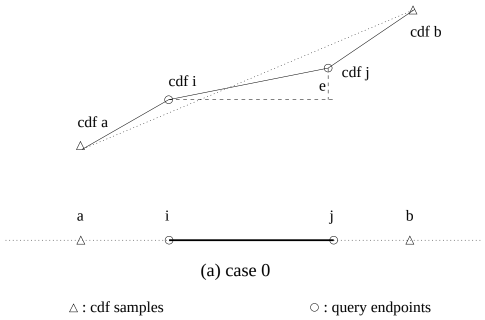


Figure 10: Processing Range Query Feedbacks

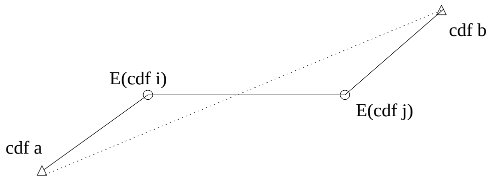
The problem is formally defined as follows. As depicted in Figure 10, let the user range query be $Q(i \leq x \leq j)$ on attribute x with domain range $[0..n-1]$ and $0 \leq i \leq j \leq n-1$. Let the actual query result of Q, from the query processor, be e. We want to estimate cdfs at i and j, which are $E(cdf_i)$ and $E(cdf_j)$, respectively, in order to update the interpolated cdf.

Although the exact cdf_i and cdf_j are unknown and to be estimated, we do know that the exact result $Q(i \leq x \leq j) = cdf_j - cdf_i$ by definition as discussed in Section 2.2. If both cdf_i and cdf_j are in the sample set, then no action is needed. If one of cdf_i and cdf_j is in the sample set, then we can calculate the other without introducing any error. However, if both cdf_i and cdf_j are not in the sample set, we need to estimate them with interpolation. There are three cases:

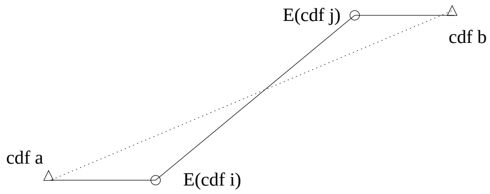
- Case 0: There is no sample in query range $[i, j]$.



(a) general case



(b) e = 0



(c) e = $cdf_b - cdf_a$

Figure 11: Range Query Feedbacks: Case 0

As depicted in Figure 11.(a), the dotted line is the original interpolated cdf. Let cdf_a be the largest cdf sample such that $a \leq i$ and cdf_b be the smallest cdf sample such that $j \leq b$. The only true values we know are cdf_a, cdf_b and e, which give us the number of tuples with attribute value x in ranges $[a, b]$ and $[i, j]$ are

$$Q(a \leq x \leq b) = cdf_b - cdf_a,$$
$$Q(i \leq x \leq i) = cdf_j - cdf_i,$$

respectively. We do not know how tuples are distributed with attribute value x in range $[a, i]$ and $[j, b]$. However, the total number of tuples in ranges $[a, i]$ and $[j, b]$ is

$$Q(a \leq x \leq i) + Q(j \leq x \leq b) = (cdf_b - cdf_a) - (cdf_j - cdf_i) = (cdf_b - cdf_a) - e$$

Since we do not know the real distribution of tuples in ranges $[a, i]$ and $[j, b]$, we assume that tuples are uniformly distributed. As a result, we assign the number of tuples in ranges $[a, i]$ and $[j, b]$ with respect to their ranges; i.e., the estimated number of tuples in range $[a, i]$, $\hat{Q}(a \leq x \leq i)$, and in range $[j, b]$, $\hat{Q}(j \leq x \leq b)$ are:

$$\hat{Q}(a \leq x \leq i) = ((cdf_b - cdf_a) - e) \times \frac{i - a}{(i - a) + (b - j)},$$
$$\hat{Q}(j \leq x \leq b) = ((cdf_b - cdf_a) - e) \times \frac{b - j}{(i - a) + (b - j)},$$

respectively. And the estimated $E(cdf_i)$ and $E(cdf_j)$ can be computed as follows:

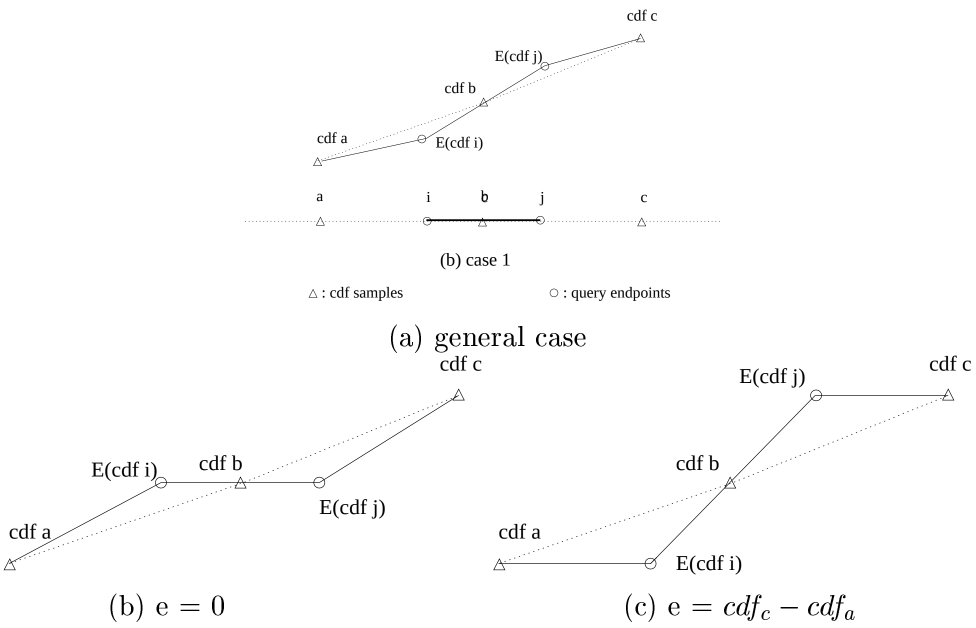
$$E(cdf_i) = cdf_a + ((cdf_b - cdf_a) - e) \times \frac{i-a}{(i-a)+(b-j)},$$
$$E(cdf_j) = cdf_b - ((cdf_b - cdf_a) - e) \times \frac{b-j}{(i-a)+(b-j)},$$

respectively. As a result, note that in Figure 11.(a), the slopes $S_{(cdf_a, cdf_i)}$ between cdf_a and $E(cdf_i)$ is the same as the slopes $S_{(cdf_j, cdf_b)}$ between $E(cdf_j)$ and cdf_b. The solid line in Figure 11.(a) is the estimated cdf after inserting $E(cdf_i)$ and $E(cdf_j)$ by interpolation.

Figure 11.(a) shows the general case for processing range query feedbacks when there is no sample in the query ranges. It is interesting to show the interpolation results for the extreme cases. For example, when the query result $Q(i \le x \le j) = 0$, we know that there exist no tuples with attribute value x in range $[i, j]$ and thus the interpolation between $E(cdf_i)$ and $E(cdf_j)$, is a horizontal line, as shown in Figure 11.(b). Another extreme case is when the query result $Q(i \le x \le j) = cdf_b - cdf_a$, we know that all tuples with attribute value x in range $[a, b]$ are also in $[i, j]$. And thus there is no tuple with ranges $[a, i]$ and $[b, j]$ and the interpolations between sample pair cdf_a and $E(cdf_i)$ and sample pair $E(cdf_j)$ and cdf_b are both horizontal lines, as shown in Figure 11.(c).

We now consider the next case where there is a sample in the range $[i, j]$.

- Case 1: There is one sample in query range $[i, j]$ as depicted in Figure 12.



(a) general case

(b) e = 0

(c) e = $cdf_c - cdf_a$

Figure 12: Range Query Feedbacks: Case 1

Similar to case 0, we use the uniform assumption to estimate $E(cdf_i)$ and $E(cdf_j)$. As depicted in Figure 12.(a), the dotted line is the original interpolated cdf. Let a be the largest sample such that $a \le i$, b be the sample between i and j and c be the smallest sample such that $j \le c$, as shown in

Figure 12.(a). The only difference between case 1 and case 0 is that the interpolated line between $E(cdf_i)$ and $E(cdf_j)$ needs to pass the true value cdf_b. We use cdf_b as an anchor points and assign the number of tuples in ranges $[i, b]$ and $[b, j]$ with respect to their ranges; i.e., the estimated number of of tuples in range $[i, b]$, $\hat{Q}(i \leq x \leq b)$, and in range $[b, j]$, $\hat{Q}(b \leq x \leq j)$ are

$$\hat{Q}(a \leq x \leq i) = e \times \frac{b-i}{j-i}$$
$$\hat{Q}(j \leq x \leq b) = e \times \frac{j-b}{j-i},$$

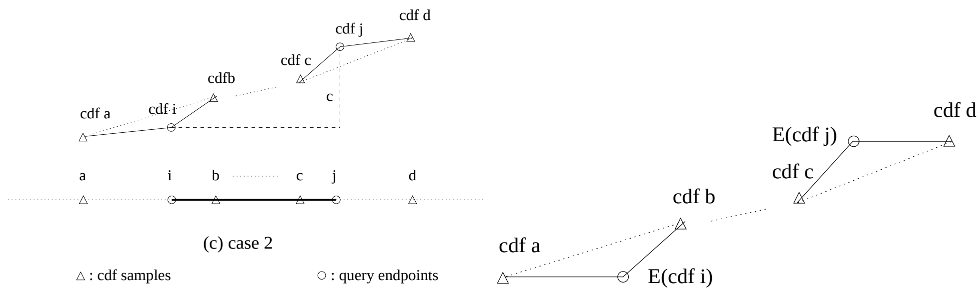
respectively. And the estimated $E(cdf_i)$ and $E(cdf_j)$ can be computed by

$$E(cdf_i) = cdf_b - e \times \frac{b-i}{j-i}$$
$$E(cdf_j) = cdf_b + e \times \frac{j-b}{j-i}$$

respectively. As a result, note that in Figure 12.(a), the slopes $S_{(cdf_i, cdf_j)}$ between $E(cdf_i)$ and $E(cdf_i)$ remains constant and the interpolated line between $E(cdf_i)$ and $E(cdf_i)$ passes through cdf_b. The solid line in Figure 12.(a) is the estimated cdf after inserting $E(cdf_i)$ and $E(cdf_j)$ by interpolation.

Figure 12.(b) and 12.(b) show the extreme cases similar to case 0. In Figure 12.(b), when the query result $Q(i \leq x \leq j) = 0$, we know that there exist no tuples with attribute value x in range $[i, j]$ and thus the interpolation between $E(cdf_i)$ and $E(cdf_j)$, is a horizontal line passing through cdf_b. Similarly, when he query result $Q(i \leq x \leq j) = cdf_c - cdf_a$, the interpolations between sample pair cdf_a and $E(cdf_i)$ and sample pair $E(cdf_j)$ and cdf_c are both horizontal lines, as shown in Figure 11.(c).

- Case 2: There are two or more cdf samples in query range $[i, j]$. Let a be the largest sample such that $a \leq i$, b be the smallest sample such that $i \leq b$, c be the largest sample such that $c \leq j$ and d be the smallest sample such that $i \leq d$, as shown in Figure 13.(a).



(a) general case

(b) e = $cdf_d - cdf_a$

Figure 13: Range Query Feedbacks: Case 2

The dotted line through cdf_a, cdf_b, cdf_c and cdf_d is the original estimate by interpolation. The solid line is the estimate after inserting $E(cdf_i)$ and $E(cdf_j)$ by interpolation. Note that since we do not have information regarding the difference between point pair cdf_i and cdf_b and point pair cdf_c and cdf_j, we assume the tuple values are uniformly distributed in these two ranges. Thus the slopes $S_{(cdf_i, cdf_b)}$ between $E(cdf_i)$ and cdf_b is equal to the slopes $S_{(cdf_c, cdf_j)}$ between cdf_c and $E(cdf_j)$.

As depicted in Figure 12.(a), the dotted line is the original interpolated cdf. The total number of tuples in ranges $[i, b]$ and $[c, j]$ is

$$Q(i \leq x \leq b) + Q(c \leq x \leq j) = (cdf_j - cdf_i) - (cdf_b - cdf_c) = e - (cdf_b - cdf_c),$$

which is similar to case 0. With the uniformity assumption, we assign the number of tuples in ranges $[i, b]$ and $[c, j]$ with respect to their ranges; i.e., the estimated number of tuples in range $[i, b]$, $\hat{Q}(i \leq x \leq b)$, and in range $[c, j]$, $\hat{Q}(c \leq x \leq j)$ are:

$$\hat{Q}(i \leq x \leq i) = (e - (cdf_c - cdf_b)) \times \frac{b - i}{(b - i) + (j - c)},$$
$$\hat{Q}(c \leq x \leq j) = (e - (cdf_c - cdf_b)) \times \frac{j - c}{(b - i) + (j - c)},$$

respectively. And the estimated $E(cdf_i)$ and $E(cdf_j)$ can be computed as follows:

$$E(cdf_i) = cdf_b - (e - (cdf_c - cdf_b)) \times \frac{b - i}{(b - i) + (j - c)},$$
$$E(cdf_j) = cdf_c + (e - (cdf_c - cdf_b)) \times \frac{j - c}{(b - i) + (j - c)},$$

respectively. As a result, in Figure 13.(a), the slopes $S_{(cdf_i, cdf_b)}$ between $E(cdf_i)$ and cdf_b is the same as the slopes $S_{(cdf_c, cdf_j)}$ between cdf_c and $E(cdf_j)$. The solid line in Figure 13.(a) is the estimated cdf after inserting $E(cdf_i)$ and $E(cdf_j)$ by interpolation.

Note that the extreme case that $Q(i \leq x \leq j) = 0$ cannot exist in case 2 with more than three samples in range $[i, j]$ because it implies more than two samples with the same cdf. The reason is that in the sampling process, the duplicate samples with the same cdf will be discarded. Figure 12.(b) shows the opposite extreme case when the query result $Q(i \leq x \leq j) = Q(a \leq x \leq d)$, which results the interpolations between sample pair cdf_a and $E(cdf_i)$ and sample pair $E(cdf_j)$ and cdf_d are both horizontal lines.

By analyzing the above three cases, we demonstrate how to incorporate user query feedbacks for updating the sample set of the dynamic sampling estimator discussed in Section 3.3. We conduct extensive experiments to verify the performance of the proposed dynamic sampling estimator. The experiment results are shown in Section 4.

4 Experimental Evaluation

In this section, we conduct experiments to show the feasibility of the proposed techniques to adapt user query patterns. First we discuss the error metric used for comparing the performance of different estimation techniques. Later we conduct experiments to show how the dynamic sampling algorithm adapts to user query patterns which are static (do not change over time) but unknown beforehand. We compare the performance of the proposed technique with the Icicle technique [GLR00], which is a tuple sampling technique that biases the sampling process according to the history of user queries, and the STHole technique [BCG01], which is a histogram technique that refines histogram buckets according to query feedback. We then conduct experiments to show how the proposed dynamic sampling algorithm adapts to user query patterns which are changing dynamically. All experiments in this section are on the real ADL dataset discussed in Section 3.1.

4.1 The Error Metric

One of the most commonly used error metrics to compare estimation techniques is the relative error. The definition of the relative error is $E_{rel}(q) = \frac{|s-s'|}{s}$, where actual result of range query q is s and the estimation result of q is s', has many problems. First problem is that $E_{rel}(q)$ is undefined when s is zero. Also, if s is very small the error is amplified because of small s. For example, let the estimated result $s' = 10$. If the actual result is $s = 19$, the absolute error is $|s - s'| = |19 - 10| = 9$ and the relative error is $\frac{9}{19} = 47\%$. But if the actual result is $s = 1$, the absolute error is $|s - s'| = |1 - 10| = 9$, which is the same as before, but the relative error becomes $\frac{9}{1} = 900\%$, which increases more than 19 times. The above example shows how small actual result values amplify the relative error while the absolute error remains unchanged. Other error metrics have been proposed to overcome this problem with limited success. For example, in [VWI98], Vitter et al. proposed to modify the definition of relative error as

$$E_{rel}(q) = \frac{|s - s'|}{max\{1, s\}}.$$

This modification eliminates the problem that the relative error is undefined when $s = 0$. But it does not solve the amplification problem. In the same paper, the notion of *combined error* is proposed to moderate the amplification problem by defining the combined error as

$$e^{comb} = MIN(\alpha \times e^{abs}, \beta \times e^{rel}).$$

The idea is that when the relative errors are amplified we report the absolute errors instead of the amplified relative errors. Note that in such scenarios the absolute errors are usually small. But the resulting error metric is ambiguous to the users.

In this paper, we do not propose a new error metric for the accuracy of query estimation. Instead, we propose to constrain the relative error when estimating the query result. To the best of our knowledge, this is the first effort that tries to bound the relative error directly instead of proposing new error metrics to compensate the amplification problem of the relative error definition. We first define $Est_{max}(q)$ to be the upper bound of the estimated result $Est(q)$ of query q generated by arbitrary estimation techniques. Similarly, we define $Est_{min}(q)$ to be the lower bound of the estimated result $Est(q)$; i.e., $Est_{min}(q) \leq Est(q) \leq Est_{max}(q)$. It can be shown that, the relative error of estimation $Est(q)$ can be constrained to be no more than δ, where $\delta \geq 100\%$, if we let $Est(q) = Est_{min}(q)$ when $Est_{max}(q) \geq (\delta + 1) \times Est_{min}(q)$.

Theorem 4.1 *Given a query q, the relative error $E^{rel}(q)$ of the estimated result $Est(q)$ cannot exceed δ, where $\delta \geq 100\%$, if the estimated result $Est(q) = Est_{min}(q)$ when $Est_{max}(q) \geq (\delta + 1) \times Est_{min}(q)$.*

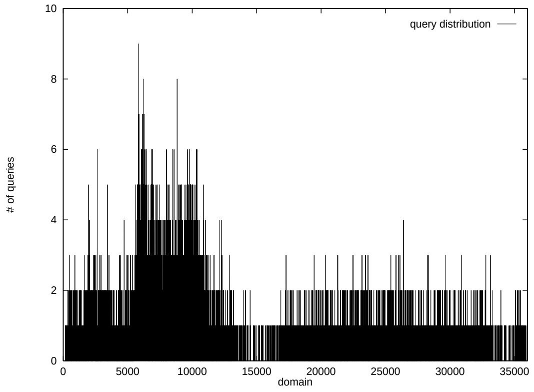
Proof. Let $Act(q)$ be the actual result of query q, then $Est_{min}(q) \leq Act(q) \leq Est_{max}(q)$. The relative error $E^{rel} = (|Act(q) - Est(q)|)/Act(q)$.

- worst case scenario 1: $Act(q) = Est_{max}(q)$ and $Est(q) = Est_{min}(q)$

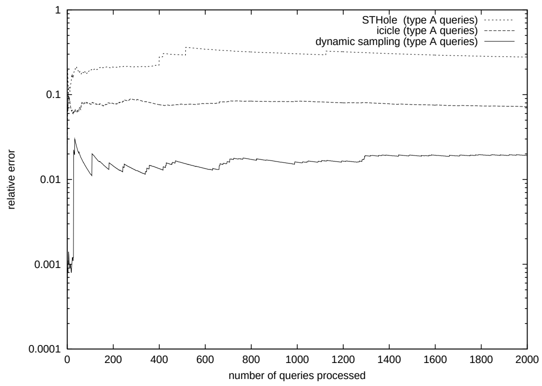
  The relative error is

  $$E^{rel} = \frac{|Act(q) - Est(q)|}{Act(q)} = \frac{|Est_{max}(q) - Est_{min}(q)|}{Est_{max}(q)} \leq 100\%$$

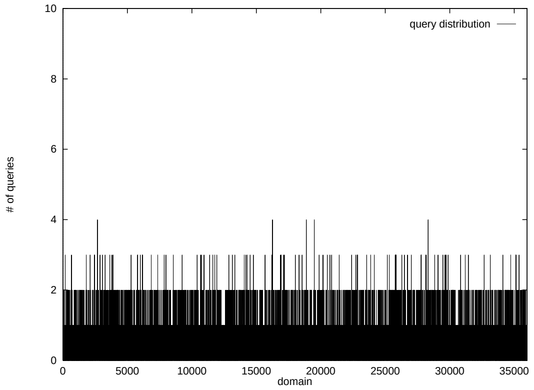
  This suggest that the relative error cannot exceed 100% in the worst case.



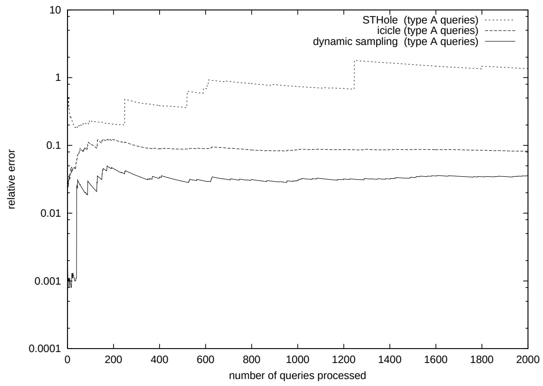
(a)



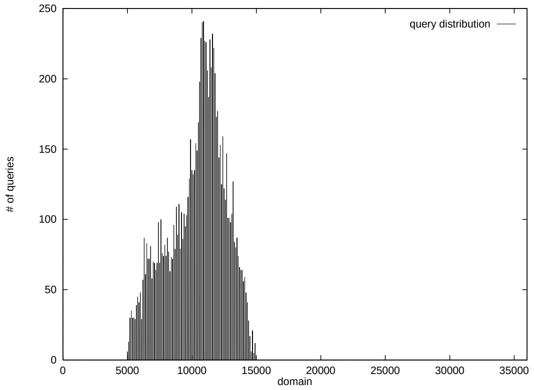
(b)



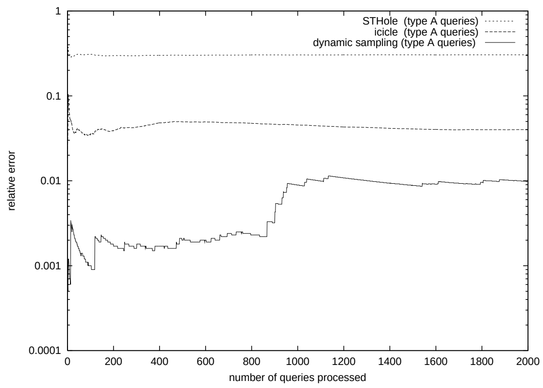
(c)



(d)



(e)



(f)

Figure 14: Average Relative Error with Different Query Patterns (Type A Query)

- worst case scenario 2: $Act(q) = Est_{min}(q)$ and $Est(q) = Est_{max}(q)$
  The relative error is

$$E^{rel} = \frac{|Act(q) - Est(q)|}{Act(q)} = \frac{|Est_{min}(q) - Est_{max}(q)|}{Est_{min}(q)}$$

  To constrain the relative error such that $E^{rel} \leq \delta$ we have

$$\begin{aligned} E^{rel} = \frac{|Est_{min}(q) - Est_{max}(q)|}{Est_{min}(q)} &\leq \delta \\ \frac{Est_{max}(q) - Est_{min}(q)}{Est_{min}(q)} &\leq \delta \\ Est_{max}(q) &\leq (\delta + 1) \times Est_{min}(q) \end{aligned}$$

  It suggests that when $Est_{max}(q) \leq (\delta + 1) \times Est_{min}(q)$, the E^{rel} cannot exceed δ.

By analyzing of the above two scenarios, we conclude that by letting

$$Est(q) = \begin{cases} Est_{min}(q) & if Est_{max}(q) \geq (\delta + 1) \times Est_{min}(q) \\ Interpolation(Est_{max}(q), Est_{min}(q)) & otherwise \end{cases}$$

the estimated result $Est(q)$ cannot exceed δ, where $\delta \geq 100\%$. □

Note that the error bound can be applied to any estimation technique that can produce maximum and minimum estimates; e.g., histogram techniques, the golden estimator technique. Our contribution here is to put a bound on the relative error to modulate the amplification problem and thus make the relative error a better metric when comparing different estimation techniques.

4.2 Experiments of the Dynamic Sampling Algorithm

We now evaluate the estimation accuracy of the proposed dynamic sampling algorithm when the user query patterns are static but unknown. We start with experiments on the same one dimensional dataset as discussed in the earlier sections with prefix queries (Type A queries [VWI98]). We compare the performance of the proposed dynamic sampling algorithm with the Icicle technique [GLR00], and the STHole technique [BCG01]. For each technique we assigned a fixed storage space. We assign a set of 100 tuples samples for the Icicle technique and a set of 100 cdf samples for the proposed dynamic sampling technique. For the STHole histogram technique, since for each bucket we have to maintain the starting position, the ending position, the number of tuples in the bucket, the parent bucket and a list of children buckets (not necessary if computed on-line), so at least four numbers needed to be stored for each bucket. As a comparison, only two numbers are needed for each Icicle sample (tuple ID and weight) and two numbers for the proposed dynamic sampling technique (domain value and cdf value). As a result, for a fixed amount of storage, only half the number of histogram buckets are assigned to the STHole technique for performance comparison, which is 50 histogram buckets. Figure 14.(a) depicts the user query pattern follows the data distribution of the ADL data set in one dimension. Figure 14.(c) depicts the user query pattern follows the uniform distribution of ADL data domain in one dimension. Figure 14.(e) depicts that the user query pattern is clustered in one dimension; i.e., the user queries focus on the [40W:130W] longitude, which is roughly the American continent. Figure 14.(b), 14.(d) and 14.(f), show the average relative errors with queries from

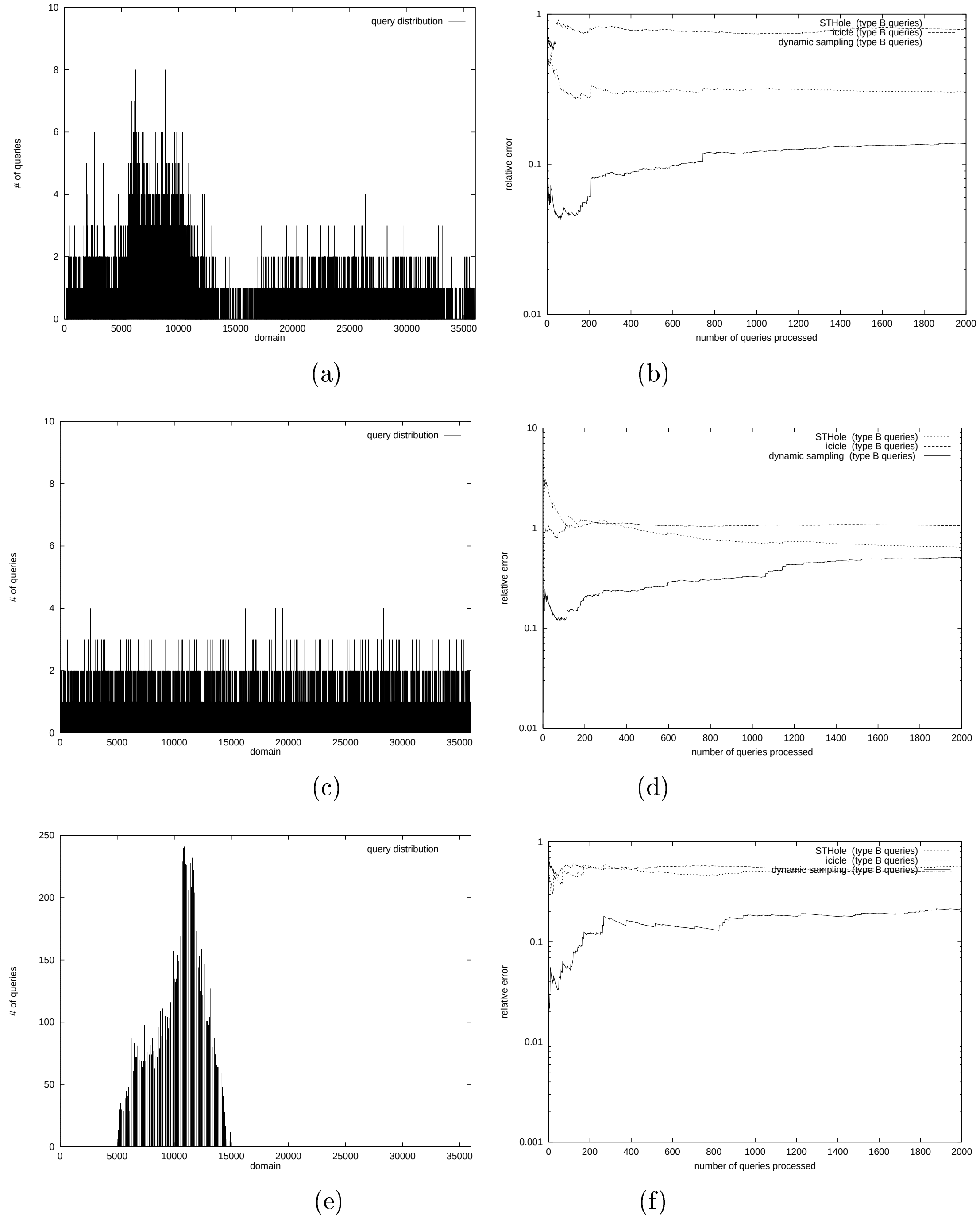


Figure 15: Average Relative Error with Different Query Patterns (Type B Query)

the distributions of Figure 14.(a), 14.(c) and 14.(e). For example, Figure 14.(b) shows that after processing 200 queries the dynamic estimator yields an average of 1.5% estimation error for the first 200 queries. The results show that the proposed dynamic sampling algorithm produces less average relative error compared to the Icicle technique and the STHole technique.

We now extend the same set of experiments to range queries (Type B queries) with 1% selectivity with respect to the domain. The amount of space used by each comparing techniques are the same as in the previous experiments, which is 100 samples for Icicle technique, 100 samples for the proposed dynamic sampling technique, and 50 histogram buckets for the STHole histogram technique. Figure 15.(a) depicts the centers of the user queries follow the data distribution of the ADL data set in one dimension. Figure 15.(c) depicts the centers of the user queries follow the uniform distribution of ADL data domain in one dimension. Figure 15.(e) depicts the clustered user query centers. Figure 15.(b), 15.(d) and 15.(f), show the average relative errors with queries in Figure 15.(a), 15.(c) and 15.(e), being processed, respectively. The results again show that the proposed dynamic sampling algorithm outperforms the Icicle technique and the STHole technique.

4.3 Experiments on Dynamic User Query Patterns

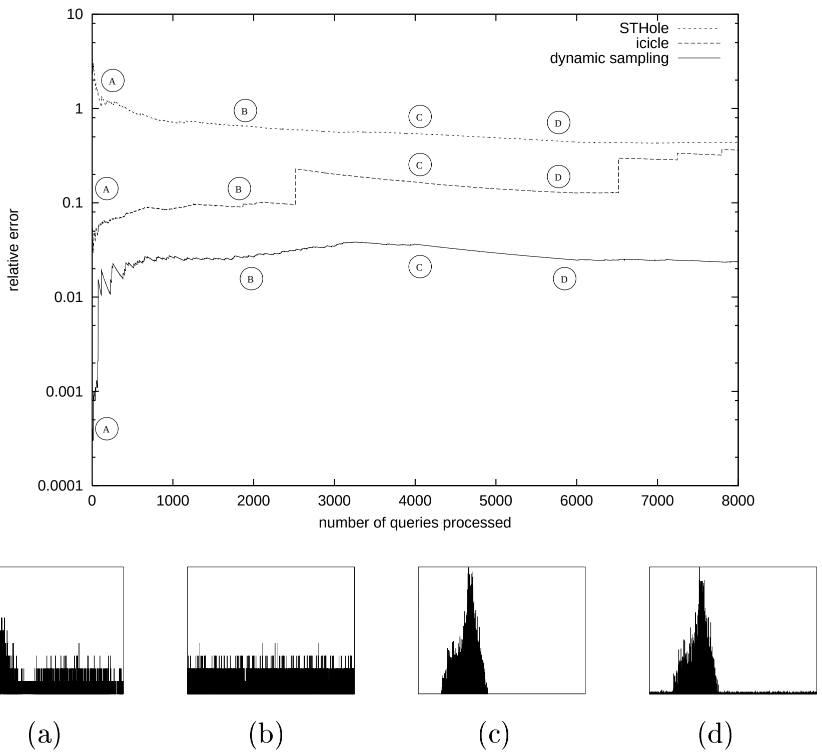


Figure 16: Average relative error with different query patterns

We now evaluate the estimation accuracy of the proposed dynamic sampling algorithm when user query patterns are dynamically changing. Figure 16 shows that a user query pattern that changes every 2,000 queries; i.e., the first 2,000 queries follow the distribution in Figure 16.(a) and the next 2,000 queries follow the distribution in Figure 16.(b) and so on. The results show that the proposed dynamic sampling algorithm with interpolation adapts better to user query pattern changes with less average relative error when compared to the Icicle technique and the STHole technique.

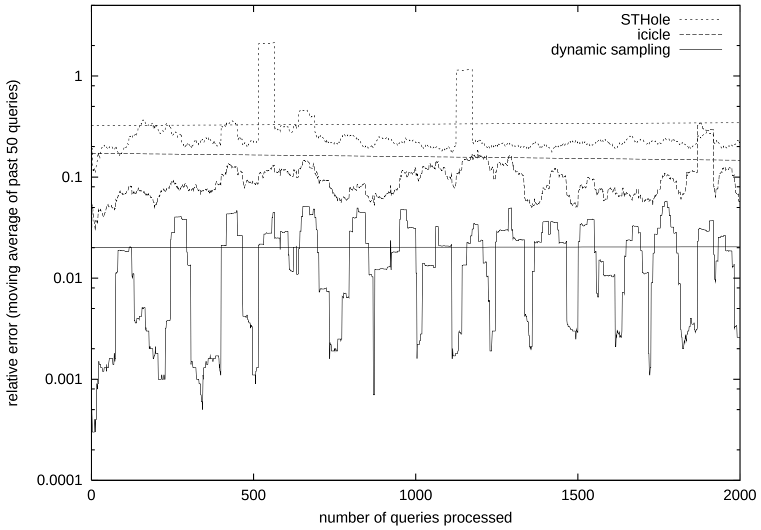


Figure 17: Moving average error with window size 50

Figure 16 shows the estimation error as the average of relative errors of all past queries since the estimator was initiated. However, users may be more interested in the recent trend of performance of the estimator. Figure 17 shows the estimation errors (with the same experiment settings as in Figure 16) as the moving average of window size 50; i.e., the average error of the past 50 queries. For example, the average error of queries q_{450} to q_{500} incurred in Icicle is about 8%, in STHole is about 20%, and in dynamic sampling is about 0.2%. In general, what we observe is that the maximum error produced by the dynamic sampling algorithm in terms of moving average is about 5% whereas the minimum goes to as little as 0.1%. The result shows that for the recent trend of estimation performance, the proposed dynamic sampling algorithm consistently produces less error than the Icicle and the STHole techniques.

5 Conclusion

Query size estimation is a critical component in DBMS for generating optimal query execution plans and feedback to the users. Commercially, most DBMSs employ histogram based techniques for approximation of data distributions. Many other technique for summarizing data distributions in database have been proposed; e.g., MaxDiff histogram technique [PIHS96], wavelet based technique [VWI98, VW99], etc. Our previous works, the golden estimator technique [WAE01] based on random sampling, demonstrate how to accurately capture the data distributions without any a priori knowledge of the actual underlying data distributions. However, as discussed in Section 3.1, while the main objective is to reduce the average estimation errors, our result strongly suggests that capturing the query distribution is more important than capturing the data distribution.

We propose a new technique which can be used for capturing user query patterns in database. Instead of tuple sampling technique and its variations (e.g., the *Icicle* [GLR00]) that summarizes the relations and the modified histogram techniques that summarizes fdf (e.g., the *STHole* [BCG01]) , we proposed techniques that sample the user query patterns and use the cdf of the data distribution for more accurate estimates of the size of selections or range queries in databases. The proposed technique is based on random sampling and therefore is highly efficient both in terms of computational and storage cost.

It is interesting to note that several variations can be made from the proposed algorithms in Section 3. For example, instead of generating a uniform sample over the entire duration of the query history, one can apply the caching algorithms (e.g., Least Recently Used (LRU)) to generate different flavors of samples that capture recent trends of the user queries. Hence, the dynamic sampling algorithm proposed in Section 3.3 may be viewed as a framework of adaptive estimation techniques on cdf.

Our future work will explore possible variations of the proposed algorithms and their applications for estimating other types of queries (join, intersection, exclusion, etc.) Also we plan to develop efficient techniques to address the problem of updating the cdf in a dynamic environment.

References

[AC99] Ashraf Aboulnaga and Surajit Chaudhuri. Self-tuning histograms: Building histograms without looking at data. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, pages 181–192, 1999.

[APR99] Swarup Acharya, Viswanath Poosala, and Sridhar Ramaswamy. Selectivity estimation in spatial databases. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, pages 13–24. ACM Press, 1999.

[BCG01] Nicolas Bruno, Surajit Chaudhuri, and Luis Gravano. Stholes: A multidimensional workload-aware histogram. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, Santa Barbara, CA, May 21-24, 2001*, pages 211–222. ACM Press, 2001.

[CDN01] Surajit Chaudhuri, Gautam Das, and Vivek Narasayya. A robust, optimization-based approach for approximate answering of aggregate queries. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, Santa Barbara, CA, May 21-24, 2001*, pages 295–306. ACM Press, 2001.

[CR94] Chung-Min Chen and Nick Roussopoulos. Adaptive selectivity estimation using query feedback. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27, 1994*, pages 161–172, 1994.

[DIR00] Donko Donjerkovic, Yannis E. Ioannidis, and Raghu Ramakrishnan. Dynamic histograms: Capturing evolving data sets. In *Proceedings of the 16th International Conference on Data Engineering, 28 February - 3 March, 2000, San Diego, California, USA*, 2000.

[GLR00] Venkatesh Ganti, Mong-Li Lee, and Raghu Ramakrishnan. Icicles: Self-tuning samples for approximate query answering. In *VLDB 2000, Proceedings of 26th International Conference on*

Very Large Data Bases, September 10-14, 2000, Cairo, Egypt, pages 176–187. Morgan Kaufmann, 2000.

[GM98] Phillip B. Gibbons and Yossi Matias. New sampling-based summary statistics for improving approximate query answers. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 331–342. ACM Press, 1998.

[GMP97] Phillip B. Gibbons, Yossi Matias, and Viswanath Poosala. Fast incremental maintenance of approximate histograms. In *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, pages 466–475. Morgan Kaufmann, 1997.

[HH99] Peter J. Haas and Joseph M. Hellerstein. Ripple Joins for Online Aggregation. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 287–298, Phildelphia, Pennsylvania, June 1999.

[HHW97] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. Online Aggregation. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 171–182, Arizona, 1997.

[HS92] Peter J. Haas and Arun N. Swami. Sequential sampling procedures for query size estimation. In *Proceedings of 1992 ACM SIGMOD international conference on Management of data*, pages 341–350, 1992.

[HS95] P. Haas and A. Swami. Sampling-based Selectivity for Joins Using Augmented Frequent Value Statistics. In *Proceedings of the 11th IEEE Conference on Data Engineering - ICDE 95*, pages 522–531, 1995.

[IP95] Yannis E. Ioannidis and Viswanath Poosala. Balancing Histogram Optimality and Practicality for Query Result Size Estimation. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 233–244, March 1995.

[JKM+98] H.V. Jagadish, Nick Koudas, S. Muthukrishnan, Viswanath Poosala andKen Sevcik, and Torsten Suel. Optimal Histogram with Quality Guarantees. In *Proceedings of the 24th VLDB Conference - VLDB 98*, New York, USA, 1998.

[LNS90] Richard J. Lipton, Jeffrey F. Naughton, and Donovan A. Schneider. Practical selectivity estimation through adaptive sampling. In *Proceedings of 1990 ACM SIGMOD international conference on Management of data*, pages 1 – 11, 1990.

[MVW98] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. Wavelet-Based Histograms for Selectivity Estimation. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 448–459, Seattle, 1998.

[PG99] Viswanath Poosala and Venkatesh Ganti. Fast approximate answers to aggregate queries on a data cube. In *11th International Conference on Scientific and Statistical Database Management,* *Proceedings, Cleveland, Ohio, USA, 28-30 July, 1999*, pages 24–33. IEEE Computer Society, 1999.

[PI97] Viswanath Poosala and Yannis E. Ioannidis. Selectivity Estimation Without the Attribute Value Independence Assumption. In *Proceedings of the 23rd VLDB Conference - VLDB 97*, pages 486–495, Ahtens, Greece, 1997.

[PIHS96] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. Shekita. Improved Histograms for Selectivity Estimation of Range Predicates. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, May 1996.

[Poo97] Viswanath Poosala. *Histogram-Based Estimation Techniques in Database Systems.* Ph. D. Dissertation, University of Wisconsin-Madison, 1997.

[Smi96] T. R. Smith. A digital library for geographically referenced materials. *Computers*, 29(5):54–60, May 1996.

[Vit85] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transaction Mathematical Software*, 11(1):37 – 57, March 1985.

[VW99] Jeffrey Scott Vitter and Min Wang. Approximate Computation of Multidimensional Aggregates of Sparse Data Using Wavelets. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 193–204, Phildelphia, June 1999.

[VWI98] Jeffrey Scott Vitter, Min Wang, and Bala Iyer. Data Cube Approximation and Histograms via Wavelets. In *Proceedings of Seventh International Conference on Information and Knowledge Management - CIKM'98*, pages 96–104, Bethesda, Maryland, November 1998.

[WAE01] Yi-Leh Wu, Divyakant Agrawal, and Amr El Abbadi. Applying the golden rule of sampling for query estimation. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, Santa Barbara, CA, May 21-24, 2001*, pages 449–460. ACM Press, 2001.