Bifocal Sampling for Skew-Resistant Join Size Estimation

Sumit Ganguly
Rutgers University
sumit@cs.rutgers.edu

Phillip B. Gibbons
Bell Laboratories
gibbons@bell-labs.com

Yossi Matias
Bell Laboratories
matias@bell-labs.com

Avi Silberschatz
Bell Laboratories
avi@bell-labs.com

Abstract

This paper introduces *bifocal sampling*, a new technique for estimating the size of an equi-join of two relations. Bifocal sampling classifies tuples in each relation into two groups, sparse and dense, based on the number of tuples with the same join value. Distinct estimation procedures are employed that focus on various combinations for joining tuples (e.g., for estimating the number of joining tuples that are dense in both relations). This combination of estimation procedures overcomes some well-known problems in previous schemes, enabling good estimates with no a priori knowledge about the data distribution. The estimate obtained by the bifocal sampling algorithm is proven to lie with high probability within a small constant factor of the actual join size, regardless of the skew, as long as the join size is $\Omega(n \lg n)$, for relations consisting of n tuples. The algorithm requires a sample of size at most $O(\sqrt{n} \lg n)$. By contrast, previous algorithms using a sample of similar size may require the join size to be $\Omega(n\sqrt{n})$ to guarantee an accurate estimate. Experimental results support the theoretical claims and show that bifocal sampling is practical and effective.

1 Introduction

Accurate and inexpensive estimation of database query sizes is useful for many purposes. Such estimates are used by query optimizers, to compare costs of alternate join plans. These estimates are useful in determining the resource allocation necessary to balance workloads on multiple processors in parallel or distributed databases. Finally, the estimate of query sizes is of interest by itself in some applications, such as financial audits and statistical studies.

There are several advantages of sampling-based estimation algorithms. Unlike parametric methods, there is no need to make assumptions about the fit of the data to an assumed distribution. Unlike histogram-based or nonparametric-based methods that rely on summary statistics, the sampling-based approaches do not rely on storing and maintaining such summary information about the data. Furthermore, in contrast to these approaches, sampling-based approaches always associate a statistical confidence (typically 95% or higher) to the estimates returned by the algorithm.

1.1 Previous work

The design of sampling-based estimation algorithms is a popular area of research [HÖT88, HÖT89, LN89, LN90, LNS90, HÖD91, HS92, LS92, LNSS93, HNSS93, HNS94, LN95, HNSS95]. Results in [LNS90, HÖD91, HS92, HNS94] and elsewhere demonstrate the practicality of estimation procedures based on sampling by showing that the time taken to compute the estimate is a small fraction of the time taken to compute the actual query.

Hou, Özsoyoğlu and Taneja [HÖT88, HÖT89] present initial work in this area. They present unbiased and consistent estimators for estimating the join size. Consistent estimators guarantee that as the sample size increases, the probability of error decreases. They also present an algorithm for cluster sampling. However, no bounds on the required sample size are presented.

Lipton and Naughton [LN90, LN95] and Lipton, Naughton and Schneider [LNS90] present algorithms based on sampling for estimating the sizes of various *select* or *join* queries (they also consider *transitive closure* and *general recursive Datalog* queries). The algorithms view the query as a collection of disjoint subqueries. Subqueries are selected at random and their sizes are computed. Termination occurs when either the sum of the subquery sizes is sufficiently large, or the number of samples taken is sufficiently large. The algorithms with high probability either estimate the query size to within some given percentage of its true value, or else guarantee that the query size is bounded by a specified value ("sanity bounds"). Since the number of sample subqueries taken adapts to the sizes of the subqueries taken thus far, these algorithms are known as *adaptive sampling algorithms*.

The adaptive sampling algorithms in [LN90, LNS90, LN95] assume knowledge of the maximum size of a subquery. Since this is not usually available, an upper bound for this quantity is used in the expression for the termination criterion of the adaptive sampling approach. As pointed out by Haas and Swami [HS92], this can lead to taking considerably more observations than necessary causing the sampling algorithm to be unduly expensive in some cases.

Haas and Swami [HS92] present improvements on the termination criterion of the adaptive sampling algorithms in [LN90, LNS90]. Specifically, they improve the case when the adaptive sampling algorithm would require many more observations than are actually necessary, simply due to assuming a high upper bound on the maximum size of a subquery. This leads, for example, to improved estimations of equi-join sizes. However, given an equi-join $R \bowtie R'$ between a non-skewed relation and a skewed relation, their algorithms do not perform very well: "[When] R' is highly skewed (Zipf) and relation R has relatively little skew ...coverage deteriorates. ...[A good] estimation algorithm is hard for this type of query" [HS92].

Hou, Özsoyoğlu and Dogdu [HÖD91] also present improvements on the termination criterion of adaptive sampling algorithms by using a pilot sample to compute an estimate for the mean of the sample sizes and the variance of the sample sizes. The drawback to this procedure, as pointed out by Haas and Swami, is that there is no theoretical guidance as to the appropriate size of the pilot sample.

A supplemental study to sampling-based methods regarding the termination criterion was presented by Ling and Sun [LS92]. Finally, further refinements were given by Haas, Naughton, Seshadri and Swami [HNSS93], and by Haas, Naughton and Swami [HNS94].

In [HNSS93], Haas *et al.* categorize sampling algorithms for join size estimation into six groups based on whether the unit of sampling is an individual tuple (t) or a memory page worth of tuples (p), and whether sampled tuples are compared (i) with all tuples in the join relation(s) with the same join value, e.g. when the appropriate indexes are provided (denoted *t_index* or *p_index*), (ii) with all tuples occurring in samples taken from these join relation(s) (*t_cross* or *p_cross*), or (iii) with only tuples in the most recent sample taken from these join relation(s) (*t_indep* or *p_indep*). They present results comparing these groups based on the variance obtained on different distributions, as well as improved sampling algorithms.

1.2 Contributions of this paper

In this paper, we introduce *bifocal sampling*, a new technique for estimating the size of an equi-join of two relations. Bifocal sampling classifies tuples in each relation into two groups, sparse and dense, based on the number of tuples with the same join value. Distinct estimation procedures are employed that focus on various combinations for joining tuples (e.g., for estimating the number of joining tuples that are dense in both relations). The individual estimation procedures we use are variants on types *t_index* and *t_cross*. This combination of estimation procedures overcomes some of the problems in previous schemes, enabling good estimates with no a priori knowledge about the data distribution required. Unlike the stratified sampling of Haas and Swami [HS92], the classification of tuples into groups is not known a priori, but must be inferred by the sampling algorithm.

Our bifocal sampling algorithm requires a sample of size at most $O(\sqrt{n} \cdot \lg n)$[1] from relations consisting of n tuples. Except for rather small join sizes, of size $o(n \lg n)$ (for which no existing sampling algorithms give good estimates), the estimate computed by the algorithm is provably within a small constant factor of the actual size of the join with high probability, regardless of the skew.

Our experimental results support the theoretical claims and show that the bifocal sampling algorithm is both practical and effective. The experiments compare bifocal sampling with two generic sampling algorithms, one of type *t_index* and one of type *t_cross*. All algorithms take the same number of samples. As an example of how bifocal sampling overcomes problems with previous algorithms, we present experimental results for Haas and Swami's "hard" scenario described above of an equi-join, $R \bowtie S$, for little-skewed R and highly-skewed S. (This scenario was also recognized to be hard in [HNS94].) We obtain excellent results while the two generic algorithms obtain rather poor results. Since previous adaptive sampling algorithms of type *t_index* (or type *t_cross*) differ primarily in their termination criteria, the results are relevant to all previous adaptive sampling algorithms of this type. In particular, previous adaptive sampling algorithms either terminate with fewer samples, and hence obtain no better results than the generic algorithms, or terminate with more samples, and hence run slower than our bifocal sampling algorithm.

The advantages of bifocal sampling over previous methods can be summarized as follows:

- No assumption is being made on the distribution of the data. For example, our analysis does not depend on the Central Limit Approximation used in many previous works, and the algorithm does not require a priori knowledge of the classification of tuples into sparse and dense.

[1] Throughout this paper, $\lg n$ denotes the base two logarithm of n.

- The algorithm guarantees estimates within a small constant factor for all cases in which the join is of size $\Omega(n \lg n)$, regardless of data distribution and skew.
- The sample size is always at most $O(\sqrt{n} \lg n)$.
- For every input, the probability of failing to provide the promised estimate with the above sample size is at most $n^{-\alpha}$, for any prespecified constant α.

As part of our algorithm, we perform *t_index*-type sampling on both join relations. It remains for future work to explore whether this algorithm can be effectively extended to use only *p_cross*-type sampling, for cases where indices are not available.

The rest of the paper is organized as follows. Section 2 presents background material on previous sampling algorithms and their problems. The bifocal sampling algorithm and its analysis are given in Section 3. Preliminary experimental results are described in Section 4.

2 Sampling algorithms revisited

In this section, we review previous sampling approaches and the problems they have estimating join sizes in the presence of certain types of skew. First, we consider adaptive sampling algorithms of type *t_index*: each sampled tuple in R is compared with all tuples in S, and the result is scaled as appropriate. We describe some of the difficulties such algorithms have in estimating join sizes. Then, we consider sampling algorithms of type *t_cross*: a random sample is taken from R and S, the size of their join is computed, and the result is scaled as appropriate. We show that such algorithms suffer from similar difficulties.

2.1 Adaptive sampling

We consider the general model for adaptive sampling proposed by Lipton and Naughton [LN90] and refined by several subsequent works (e.g. [LNS90, HÖD91, HS92, HNSS93, HNS94]). The problem is to estimate the size, A, of a given query. Conceptually, we view the query as a disjoint union of n subqueries. The sampling algorithm, of type *t_index*, repeatedly chooses a random subquery, computes the size of its output and adds it to the current sum A^*. After k such subqueries, where k is determined by a specified termination condition, the algorithm terminates and estimates the size of the query as $(n/k) \cdot A^*$.

Figure 1 depicts a generic adaptive sampling algorithm for estimating the size of an equi-join $R \bowtie S$. The algorithm takes five input parameters n, b, ϵ, p, m. The parameter n, the number of tuples in R, is the size of the population from which to sample. The parameter b is an upper bound on the output size of a subquery, that is, an upper bound on the program variable x. The parameter m is an upper bound on the number of iterations of the while loop; it guards against the sampling algorithm being too expensive. The function f in the procedure specifies the termination condition. It depends only on the desired confidence and inaccuracy parameters, p and ϵ respectively, and is greater than one.

```
Algorithm Adaptive-Sampling(n, b, ε, p, m)
   // n is the number of tuples in R.
   // b is an upper bound on the size of a subquery.
   // ε bounds the relative inaccuracy of the estimate.
   // p is the confidence on the estimates returned.
   // m is the maximum sample size.
U := NONE;   A* := 0;   k := 0;
while A* < b · f(p, ε) and k < m do begin
   Select a random tuple τ from R;
   Determine the number, x, of tuples in S
      that join with τ;
   A* := A* + x;
   k := k + 1;
end;
Â := nA*/k;
if A* < b · f(p, ε) then U := g(A*, n, b, p, m);
return (Â, U);
```

Figure 1: A generic adaptive sampling algorithm for estimating the size of an equi-join $R \bowtie S$. Note that the values for parameters b and m can themselves be determined by sampling or by known statistics on the data. Also, different adaptive sampling algorithms vary in the choice of function f for the termination condition and function g for the sanity bound, $\mathcal{U}$.

The algorithm terminates by returning a 2-tuple, $(\hat{A}, \mathcal{U})$. Each run of the algorithm falls into one of two cases:

1. The preferred case, characterized by the algorithm terminating with $A^* \geq b \cdot f(p, \epsilon)$. In this case, $\hat{A}$, the algorithm's estimate for A, is guaranteed to be within ϵA of the value for A with probability p.
2. The remaining case is called the case of *sanity bounds*. In this case, $\hat{A}$ may or may not be a good estimate for A, and the only claim is that $\mathcal{U} = g(A^*, n, b, p, m)$ is an upper bound on A with high probability.

The generic algorithm given above depicts the simplified termination criteria used, e.g., in [LNS90]. More refined termination criteria, see e.g. [HNSS93], are functions of not only confidence and inaccuracy parameters, but also the observed standard deviation in the subquery sizes. While these refined termination criteria can result in fewer samples, they do not change the quality of the estimate for a given number of samples. Thus, for simplicity, we consider the generic algorithm in the discussions that follow, although many of the problems addressed apply as well to these more refined algorithms.

2.2 A problem with adaptive sampling

Previous adaptive sampling algorithms have an inherent problem with estimating join sizes that are not sufficiently large. Consider the problem of estimating the join size A to within an inaccuracy ϵ and confidence p, using at most m samples. If $A < (nb/m) \cdot f(p, \epsilon)$, then it is expected that $A^* < b \cdot f(p, \epsilon)$ and hence it is likely that the adaptive sampling algorithm will not terminate with a good estimate after taking m samples. For example, if $m = \sqrt{n}$, and $b = \Theta(n)$, then A should be at least $\Omega(n\sqrt{n})$ in order to allow accurate estimation. For join sizes below this threshold, it is possible that the adaptive sampling algorithm is unable to accurately estimate it using m samples, as the following example demonstrates.

Example 1: Consider the equi-join between two relations, $R(B) = \{1, 2, \ldots, n\}$ and $S(B, C) = \{(1, 1), (1, 2), \ldots, (1, n)\}$. The equi-join of these two relations is equal to S, and therefore has n tuples. Consider the problem that occurs when adaptive sampling is used such that m random tuples are chosen from R. If the tuple with join attribute value 1 is chosen, then the final sum A^* is at least n (it is larger than n if this tuple is chosen more than once). Thus the estimate $\hat{A} = nA^*/m$ is at least n^2/m, causing a tremendous overestimate, unless $m = \Omega(n)$. On the other hand, if this tuple is not chosen in the sample, then the estimate is 0, which is tremendously inaccurate as well. Thus, with a sample of size $o(n)$, the estimate is always highly erroneous.

Suppose instead that the adaptive sampling algorithm samples tuples from S. In this case, for any sample size m, the sum A^* is m and the estimate is $nA^*/m = n$, which is accurate. The number of samples required from S prior to termination in the preferred case is $m = b \cdot f(p, \epsilon)$. If we had a priori knowledge that $b = 1$, then the sampling algorithm would terminate after taking $f(p, \epsilon)$ samples. However, if we replace b by a high upper bound, say n, then the sampling algorithm would require $f(p, \epsilon) \cdot n$ samples. Since $f(p, \epsilon) > 1$, it follows that the entire relation would be accessed several times over. Clearly, this is undesirable. Thus a more refined termination criterion that accounts for the distribution of subquery sizes (e.g. [HÖD91, HS92, HNSS93]) is useful here. □

2.3 Dual sampling

Example 1 shows that for join queries, one way to improve the estimate of the adaptive sampling algorithm is to sample from both relations. This may be done as follows. Let R and S be the two relations for which we wish to estimate the size of their join on some attribute. First, run adaptive sampling on R while keeping a limit m on the number of tuples sampled. Two outcomes may result, either the algorithm terminates with a statistical guarantee for the estimated join size or the algorithm is unable to do so and returns a sanity bound instead. In the first case, we terminate with an estimate for the join size. Otherwise, we run the adaptive sampling algorithm on S, keeping a limit m on the number of tuples sampled. Again, two outcomes may result. If a statistical guarantee is obtained, then we can terminate with an estimate. Otherwise, we obtain a sanity bound. In the second case, we return the smaller of the two sanity bounds.

Let b_R be the maximum number of tuples in R that join with any one tuple in S. Similarly, let b_S be the maximum number of tuples in S that join with any one tuple in R. Then, dual sampling allows us to estimate join sizes A such that $A > n \cdot \min(b_R, b_S)/m$.

However, if only very coarse upper bounds are available on either b_R or b_S, problems may arise. In Example 1, if n is used as an upper bound on b_R, and $m = \sqrt{n}$, the estimation algorithm would return a sanity bound although it has the correct answer.

A more serious problem is that the actual values for b_R and b_S may indeed both be high due to highly-skewed data. In particular, the problem example above for adaptive sampling (Example 1) can be extended to thwart dual sampling, as the following example demonstrates.

Example 2: Consider the equi-join on attribute B between the following two n-tuple relations, R and S:

$$\begin{aligned} R(B,C) &= \{(1,c_1),(1,c_2), \\ &\quad (2,c_1),(2,c_2),(2,c_3),\ldots,(2,c_{n/2}), \\ &\quad (3,c_1),(4,c_1),\ldots,(n/2,c_1)\} \\ S(B,D) &= \{(1,d_1),(1,d_2),\ldots,(1,d_{n/2}), \\ &\quad (2,d_1),(2,d_2), \\ &\quad (n/2+1,d_1),(n/2+2,d_1),\ldots,(n-2,d_1)\} \end{aligned}$$

The reader may verify that the equi-join of these two relations has $2n$ tuples, and that $b_R = b_S = n/2$. As argued above, dual sampling allows us to estimate join sizes A such that $A > n \cdot \min(b_R, b_S)/m$, which is $n^2/(2m)$ in the particular case of this example. Thus adaptive sampling would require more than $m = n/4$ samples in order to guarantee a good estimate. Clearly, this is undesirable. □

Scenarios such as the previous example can occur, for example, whenever both relations have a Zipf distribution on the join attribute but the peaks of the two Zipf distributions are on distinct values. Note that the adaptive sampling algorithm is not effective here even though the exact values of b_R and b_S are assumed to be known. Therefore, for such scenarios any implementation of the adaptive sampling algorithm (with or without dual sampling) would not be effective, regardless of the method used to estimate upper bounds for b_R and b_S, and regardless of the choice of $f(b, \epsilon)$.

Thus, dual sampling, though superficially attractive, does not solve the problem of skewed data in previous adaptive sampling algorithms.

2.4 Cross sampling

In sampling algorithms of type *t_cross* [HNSS93], a random sample is taken from both R and S, the join size of the samples is computed, and the result is scaled as appropriate. In particular, if samples of size m are taken from relations of size n, and if the join size of the two samples is A^*, then the size of $R \bowtie S$ is estimated as $(n/m)^2 \cdot A^*$. Such cross sampling algorithms are useful whenever indices for the join predicate are not available.

Figure 2 depicts a generic cross sampling algorithm for estimating the size of an equi-join $R \bowtie S$, in which R and S may be of different size.

Cross sampling algorithms suffer from many of the same difficulties as described above for adaptive sampling. On Example 1 above, if the tuple with join attribute value 1 is chosen (at least once) when sampling from R, then $A^* \geq m$, and thus the estimate $\hat{A} = (n/m)^2 \cdot A^*$ is at least n^2/m, causing a tremendous overestimate, unless $m = \Omega(n)$. On the other hand, if this tuple is not chosen when sampling from R, then the estimate is 0, which is tremendously inaccurate as well. Thus, with a sample of size $o(n)$, the estimate is always highly erroneous. Likewise on Example 2 above, if a tuple with value 1 is chosen when sampling from R or a tuple with value 2 is chosen when sampling from S, then (the reader may verify that) with high probability A^* is $\Omega(m)$, and hence $\hat{A}$ is $\Omega(n^2/m)$. This is a tremendous overestimate, unless $m = \Omega(n)$. On the other hand, if no such tuples are chosen when sampling, then the estimate is 0, which is tremendously inaccurate as well.

Algorithm Cross-Sampling(n_1, n_2, m)

  // n_1 is the number of tuples in R.
  // n_2 is the number of tuples in S.
  // m is the sample size for each relation.

Select m random tuples from R, and let R^* be the resulting random sample;
Select m random tuples from S, and let S^* be the resulting random sample;
Let A^* be the number of tuples in $R^* \bowtie S^*$;
$\hat{A} := n_1 \cdot n_2 \cdot A^*/m^2$;
return $\hat{A}$;

Figure 2: A generic sampling algorithm of type *t_cross* for estimating the size of an equi-join $R \bowtie S$.

3 Bifocal sampling

In this section, we present the new bifocal sampling algorithm, designed to better address the problems mentioned in Section 2.

Consider estimating the size of an equi-join $R \bowtie S$. To simplify the descriptions that follow, assume that the join predicate is of the form $R.J = S.J$ for some attribute J, and that each relation has the same number of tuples, n. The results in this section can readily be extended to handle the more general cases.

Bifocal sampling classifies tuples in each relation into two groups, sparse and dense, based on the number of tuples with the same join attribute value. Distinct estimation procedures are employed that focus on various combinations for joining tuples (e.g., for estimating the number of joining tuples that are dense in both relations), thereby obtaining better results than previous approaches for certain types of skew, and provably good results regardless of the skew. Note that no a priori knowledge of the data distribution is assumed; in particular, the classification of tuples into sparse and dense is not known a priori.

We will use the following definitions.

Definition 3.1 *For a set T of tuples and a join attribute value v, define $\mathsf{mult}_T(v)$ to be the number of tuples in T with value v.*

Definition 3.2 *A join attribute value v is defined to be* dense *in R if $\mathsf{mult}_R(v) \geq \sqrt{n}$, and defined to be* sparse *in R if $\mathsf{mult}_R(v) < \sqrt{n}$. A tuple in a relation R is defined to be* dense *(*sparse*) if its join attribute value v is dense (sparse) in R.*

The output of the join $R \bowtie S$ consists of one output tuple for each pair of tuples (τ_R, τ_S) such that τ_R is a tuple in R, τ_S is a tuple in S, and both tuples share the same join attribute value. We can view these pairs as partitioned into disjoint sets, one set per join attribute value.

Definition 3.3 *For each join attribute value v, define* SubJoin(v) *to be the set of all pairs (τ_R, τ_S) such that τ_R is a tuple in R, τ_S is a tuple in S, and both tuples share the same value v.*

Each SubJoin can be classified into one of four groups:

Definition 3.4 *SubJoin(v) is a* dense-dense SubJoin *if v is dense in both R and S. SubJoin(v) is a* dense-sparse SubJoin *if v is dense in R and sparse in S. SubJoin(v) is a* sparse-dense SubJoin *if v is sparse in R and dense in S. SubJoin(v) is a* sparse-sparse SubJoin *if v is sparse in both R and S.*

Figure 3 depicts our bifocal sampling algorithm. The algorithm takes four input parameters n, m_1, m_2, δ, as described in the figure. It employs two procedures, a dense-dense estimation procedure (Dense_Dense_Estimation, depicted in Figure 4) and a sparse-any estimation procedure (Sparse_Any_Estimation, depicted in Figure 5). The dense-dense estimation procedure estimates the size of all dense-dense SubJoins. The sparse-any estimation procedure estimates the sum of the sizes of all sparse-dense and sparse-sparse SubJoins. The algorithm estimates the join size, A, with one application of dense-dense estimation and two of sparse-any estimation, with the roles of R and S reversed. This accounts for all SubJoins by possibly accounting twice for some sparse-sparse SubJoins. We show later that for any relations R and S such that $R \bowtie S$ has an $\Omega(n \lg n)$ output size, the bifocal sampling algorithm with $O(\sqrt{n} \lg n)$ samples estimates the output size to within a small constant factor with high probability.

Bifocal sampling enables better estimates than previous approaches by focusing separately on sparse and dense tuples. In particular:

- Any dense-dense SubJoin has many tuples in each relation and hence, with high probability, will be represented by a proportionate number of such tuples in any sample. Thus, dense-dense SubJoins can be estimated with high confidence by a join performed on a small sample from each relation.

- Although sparse-dense SubJoins are likely to be missed when sampling on the sparse side (since there are few such tuples), they are not likely to be missed when sampling on the dense side (since there are many such tuples). Recall Example 1. Thus a dual sampling approach, as discussed in Section 2.3, can be applied to effectively estimate sparse-dense and dense-sparse SubJoins. As for sparse-sparse SubJoins, individual SubJoins may be missed, but the aggregate contribution to the join size of sparse-sparse SubJoins can be effectively estimated as long as the total contribution of such SubJoins is $\Omega(n)$.

- By eliminating dense tuples from consideration, the sparse-any estimation procedure can assume a small upper bound, b, on the maximum number of tuples that join with any one tuple (i.e., the size of the join subquery), thereby guaranteeing good estimates with fewer samples. The bound b for sparse tuples is $\sqrt{n}$, whereas the bound considering all tuples could be as high as n (recall Examples 1 and 2).

In the remainder of this section, we present an analysis of our bifocal sampling algorithm. In Section 3.1, we first present several definitions used in the analysis. We then outline the rationale behind the bifocal sampling algorithm by over-viewing the analysis used to bound

---

Algorithm Bifocal-Sampling(n, m_1, m_2, δ)

// n is the number of tuples in each relation.
// m_1 is the sample size for the 1st procedure.
// m_2 is the sample size for the 2nd procedure.
// δ is a threshold used in the 1st procedure.

$\mathcal{U}$:= NONE;
$\hat{A}_d := \textit{Dense\_Dense\_Estimation}(n, m_1, \delta)$;
$\hat{A}_{s_1} := \textit{Sparse\_Any\_Estimation}(R, S, n, m_2)$;
$\hat{A}_{s_2} := \textit{Sparse\_Any\_Estimation}(S, R, n, m_2)$;
$\hat{A} := \hat{A}_d + \hat{A}_{s_1} + \hat{A}_{s_2}$;
if $\hat{A} < n \lg n$ then $\mathcal{U} := n \lg n$;
return $(\hat{A}, \mathcal{U})$;

Figure 3: Our bifocal sampling algorithm for estimating the size of an equi-join $R \bowtie S$. Provably good estimates can be obtained for sample sizes m_1 at most $O(\sqrt{n} \lg n)$ and m_2 at most $O(\sqrt{n})$, and taking δ to be $\Theta(\lg n)$. (The constants in these bounds are determined by the analysis based on the desired accuracy.) In the experiments, we use $m_1 = (\sqrt{n} + \lg n) \cdot \lg n$, $m_2 = \sqrt{n} + \lg n$, and $\delta = \lg n$.

the estimation error. Section 3.2 then gives the analysis of the dense-dense estimation procedure, and finally Section 3.3 presents the analysis of the sparse-any estimation procedure.

3.1 Algorithm outline

We now discuss the two estimation procedures in somewhat more detail, in order to present a technical, but relatively high-level description of the two procedures and their performance guarantees. We begin with several definitions.

We denote a *probe* of a database to be the retrieval of all (zero or more) tuples from a given relation with a given key value. The *probe cost* is the number of probes to complete the procedure. We denote the *tuple cost* of a probe to be the number of tuples retrieved by a probe. We consider two cost models: a *strong cost model* in which we only account for the number of probes, e.g. the number of samples, and a *weak cost model* in which we account for the number of probes plus the sum of the tuple costs of all probes performed by the algorithm. The strong cost model is appropriate when the database includes an index structure that can compute efficiently $\mathsf{mult}_R(v)$ and $\mathsf{mult}_S(v)$ for a given value v; otherwise the weak cost model may be more appropriate.

We extend the definition of sparse and dense given above to apply to a general parameter m, and define as well the class of *very-dense* tuples/values for the purpose of the analysis.

Dense_Dense_Estimation(n, m_1, δ)

// n is the number of tuples in each relation.
// m_1 is the sample size for each relation.
// δ is a threshold.

$A^* := 0$;
Select m_1 random tuples from R, and
let R^* be the resulting random sample;
Select m_1 random tuples from S, and
let S^* be the resulting random sample;
Let V^* be the set of join attribute values appearing
in tuples of both R^* and S^*;
For each value $v \in V^*$, determine $\mathsf{mult}_{R^*}(v)$;
For each value $v \in V^*$, determine $\mathsf{mult}_{S^*}(v)$;

// Compute the number of tuples in $R^* \bowtie S^*$
// that appear to be in dense-dense SubJoins:
For each value $v \in V^*$ do begin
if $\mathsf{mult}_{R^*}(v) \geq \delta$ and $\mathsf{mult}_{S^*}(v) \geq \delta$ then
$A^* := A^* + \mathsf{mult}_{R^*}(v) \cdot \mathsf{mult}_{S^*}(v)$;
end;
$\hat{A}_d := (n/m_1)^2 \cdot A^*$;
return $\hat{A}_d$;

Figure 4: The procedure for estimating the number of pairs in dense-dense SubJoins.

Sparse_Any_Estimation(R, S, n, m_2)

// n is the number of tuples in each relation.
// m_2 is the sample size for each relation.

$A^* := 0$;
Select m_2 random tuples from S, and
let S^* be the resulting random sample;

// If determining $\mathsf{mult}_R(v)$ is slow for dense v then
// use probabilistic elimination to suppress dense v:
if avoid_dense_mult then begin
Select m_2 random tuples from R, and
let R^* be the resulting random sample;
For each join attribute value v appearing in R^* do
Remove from S^* all tuples with value v;
end;

// Compute the estimate on the (remaining) values
// that are sparse in R:
For each tuple τ in S^* do begin
Determine the number, x, of tuples in R
that join with τ;
if $x < n/m_2$ then $A^* := A^* + x$;
end;
$\hat{A}_s := nA^*/m_2$;
return $\hat{A}_s$;

Figure 5: The procedure for estimating the number of pairs in either sparse-sparse or sparse-dense SubJoins. Two variants of the procedure are shown. If determining $\mathsf{mult}_R(v)$ is slow for dense v then the avoid_dense_mult flag should be set so as to probabilistically suppress dense v prior to any determination of $\mathsf{mult}_R(v)$.

Definition 3.5 *Let m be a positive integer. A join attribute value v is defined to be* m-dense *in R if* $\mathsf{mult}_R(v) \geq n/m$, *and otherwise defined to be* m-sparse *in R. An (m-dense) join attribute value v is defined to be* very m-dense *in R if* $\mathsf{mult}_R(v) \geq 2n \lg n/m$. *A tuple in a relation R is defined to be* m-sparse, m-dense, very m-dense *if its join attribute value v is m-sparse, m-dense, very m-dense, respectively, in R.*

Since we use the same m throughout this section, we will subsequently refer to tuples and values as simply sparse, dense, and very dense.

In the full paper, we present the complete analysis of our bifocal sampling algorithm, showing that the algorithm estimates the join size to within an error factor close to 2 with probability at least $1 - 1/n^\alpha$, for any prespecified constant α. In what follows, we provide a sketch of this analysis, for the case where $\alpha = 1$.

The first estimation procedure, Dense_Dense_Estimation, is used to account for all dense-dense SubJoins. A sampling algorithm of type *t_cross* is used, with the novelty that values that appear less than δ times in either sample are ignored. It is shown that the number of pairs in each dense-dense SubJoin is estimated within a small error factor with high probability. Furthermore, the total (scaled) contributions of other SubJoins to the total (scaled) estimate is bounded above by a small constant times the total sum of their sizes, with high probability.

The second estimation procedure, Sparse_Any_Estimation, is used to account for all sparse-sparse and sparse-dense SubJoins. A sampling algorithm of type *t_index* is used, with the novelty that dense tuples in R (and possibly other tuples in R) are ignored. It is shown that the number of pairs in each such SubJoin is estimated within a small error factor with high probability. By repeating this sparse-any procedure for $S \bowtie R$, we obtain an estimate for dense-sparse SubJoins in $R \bowtie S$.

The total estimate is obtained by combining the three estimates. Each SubJoin is guaranteed to be reflected in at least one of the three estimates with high probability. The total estimate is provably within a factor close to 2 of the actual join size, with high probability.

The sparse-any estimation procedure determines x ($= \mathsf{mult}_R(v)$) for each join attribute value v appearing in tuples of S^*. The tuple cost for determining $\mathsf{mult}_R(v)$ is $\mathsf{mult}_R(v)$, which may be as large as n. For the weak cost model, which accounts for tuple cost, we would like to avoid determining $\mathsf{mult}_R(v)$ for dense v. (This is not a concern for the strong cost model.) We introduce a simple *probabilistic elimination* technique that neutralizes the effect of dense values v on the tuple cost. This technique can be incorporated into the algorithm as desired by setting the "avoid_dense_mult" flag: for the weak cost model, the flag should be set, while for the strong cost model, the flag need not be set.

The probabilistic elimination technique takes a random sample R^* of R and then eliminates from S^* all tuples in SubJoins that have at least one tuple in R^*. Perhaps surprisingly, even though this simple technique does not actually identify which SubJoins and tuples *ought* to be eliminated, the overall elimination is satisfactory with high confidence. We select the size, m_2, of R^* to satisfy the following three requirements:

(i) With high probability, all tuples whose value is very dense in R are eliminated.

(ii) Let τ be a tuple in S with join attribute value v such that SubJoin(v) is sparse-sparse or sparse-dense and not eliminated by R^*. The expected contribution of each such τ to the total estimate is within a small error of $\mathsf{mult}_R(v)$.

(iii) Some dense tuples in R might not be eliminated by R^*. Nevertheless, each dense tuple in R has constant expected contribution to the total tuple cost.

The analysis will show that it suffices to have the size m_2 be $\sqrt{n}$.

Requirement (i) enables us to assume that the maximum contribution of a single tuple is bounded by $2n \lg n/m$. This is helpful both for bounding the tuple cost of determining multiplicities in the algorithm, as well as for bounding the tuple cost error using tail inequalities. Requirement (ii) guarantees that the expected total contribution of all sparse-sparse and sparse-dense SubJoins is within a small error of their actual total sum of sizes. Requirement (iii) guarantees that tuples that are not sparse in R (but are not necessarily very dense) would not significantly affect the expected total tuple cost.

3.2 The dense-dense estimation procedure

The algorithm for estimating the number of pairs in dense-dense SubJoins is given in Figure 4.

Let V be the set of join attribute values. For each value $v \in V$, let $\mathcal{E}(v)$ be the contribution of v to the total estimate, $\hat{A}_d$, of dense-dense SubJoins:

$$\mathcal{E}(v) = \begin{cases} 0 & \text{if } \mathsf{mult}_{R^*}(v) < \delta \text{ or } \mathsf{mult}_{S^*}(v) < \delta \\ (n/m_1)^2 \cdot \mathsf{mult}_{R^*}(v) \cdot \mathsf{mult}_{S^*}(v) & \\ \quad \text{otherwise.} & \end{cases}$$

Note that $\hat{A}_d = \sum_{v \in V} \mathcal{E}(v)$.

Let $E(v) = \mathsf{mult}_R(v) \cdot \mathsf{mult}_S(v)$; i.e., $E(v)$ is the number of pairs in SubJoin(v). Let V_d be the set of values v with dense-dense SubJoins; i.e., $\mathsf{mult}_R(v) \geq n/m$ and $\mathsf{mult}_S(v) \geq n/m$. Let A_d be the total number of pairs in dense-dense SubJoins; i.e., $A_d = \sum_{v \in V_d} E(v)$. Finally, let $\mathcal{E}_d$ be the contribution to $\hat{A}_d$ from SubJoins that are indeed dense-dense; i.e., $\mathcal{E}_d = \sum_{v \in V_d} \mathcal{E}(v)$.

The following lemma shows that $\mathcal{E}(v)$ is a good estimate for each dense-dense SubJoin(v):

Lemma 3.1 *Let $0 < \epsilon < 1$ be an arbitrary constant, and let $c = c(\epsilon)$ be an appropriately selected constant. Let $\delta = (1-\epsilon/3)c \lg n$, and let $m_1 = cm \lg n$. Then for each $v \in V_d$,*

$$\mathbf{Pr}\left(\mathcal{E}(v) \in (1 \pm \epsilon)E(v)\right) \geq 1 - 1/n^2 \ .$$

Proof. Let $X = \mathsf{mult}_{R^*}(v)$ and let $Y = \mathsf{mult}_{S^*}(v)$. Both X and Y are binomial random variables, with parameters $(m_1, \mathsf{mult}_R(v)/n)$ and $(m_1, \mathsf{mult}_S(v)/n)$, respectively. Then,

$$\mathbf{E}(X) = \frac{m_1 \mathsf{mult}_R(v)}{n} \quad \text{and} \quad \mathbf{E}(Y) = \frac{m_1 \mathsf{mult}_S(v)}{n} \ .$$

Let $\epsilon' = \epsilon/3$. Since $v \in V_d$, we have that

$$\mathbf{E}(X) \geq c \lg n \qquad \text{and} \qquad \mathbf{E}(Y) \geq c \lg n \ .$$

Therefore, by Chernoff bounds, for a sufficiently large $c = c(\epsilon)$,

$$\mathbf{Pr}\left(X \in (1 \pm \epsilon')\mathbf{E}(X)\right) \geq 1 - \frac{1}{2n^2} \ .$$

Similarly,

$$\mathbf{Pr}\left(Y \in (1 \pm \epsilon')\mathbf{E}(Y)\right) \geq 1 - \frac{1}{2n^2} \ .$$

Therefore, since $v \in V_d$, $X \geq (1-\epsilon')c \lg n$ and $Y \geq (1-\epsilon')c \lg n$ with probability $1 - 1/n^2$, and hence $\mathcal{E}(v) > 0$. Furthermore, since $(1+\epsilon) > (1+\epsilon')^2$ and $(1-\epsilon) < (1-\epsilon')^2$,

$$\mathbf{Pr}\left(X \cdot Y \in (1 \pm \epsilon)\left(\frac{cm \lg n}{n}\right)^2 Ev\right) \geq 1 - \frac{1}{n^2} \ ,$$

and hence

$$\mathbf{Pr}\left(\mathcal{E}(v) \in (1 \pm \epsilon)Ev\right) \geq 1 - \frac{1}{n^2} \ .$$

∎

We can now conclude that the aggregate estimate $\mathcal{E}_d$ for all dense-dense SubJoins is a good one, for the parameter settings of Lemma 3.1.

Lemma 3.2 *Let $0 < \epsilon < 1$ be an arbitrary constant. Then*

$$\mathbf{Pr}\left(\mathcal{E}_d \in (1 \pm \epsilon)A_d\right) \geq 1 - 1/n \ .$$

Proof. Since $A_d = \sum_{v \in V_d} E(v)$ and $\mathcal{E}_d = \sum_{v \in V_d} \mathcal{E}(v)$, Lemma 3.1 implies that

$$\begin{aligned}
&\mathbf{Pr}\left(\mathcal{E}_d \in (1 \pm \epsilon) A_d\right) \\
&\quad \geq \quad \mathbf{Pr}\left(\forall v \in V_d : \mathcal{E}(v) \in (1 \pm \epsilon) E(v)\right) \\
&\quad \geq \quad 1 - n \cdot \frac{1}{n^2} = 1 - 1/n \ .
\end{aligned}$$

∎

In the full paper, we also prove that, with high probability, the total contribution, $\hat{A}_d - \mathcal{E}_d$, of SubJoins that are not dense-dense to the estimate $\hat{A}_d$ is bounded above by a small constant times the total sum of their sizes, $\sum_{v \in V} E(v) - A_d$.

Implementation and complexity. Selecting the random samples takes $2m_1$ probes, which by the choice of m_1 in the analysis is $O(m \lg n)$ probes. The remaining steps of the algorithm can be implemented by first (semi-)sorting the sets R^* and S^*; these steps involve no probes. Thus the probe cost as well as the tuple cost of the algorithm is $O(m \lg n)$.

3.3 The sparse-any estimation procedure

The algorithm for estimating the number of pairs in either sparse-sparse or sparse-dense SubJoins is given in Figure 5. We analyze the algorithm for $m_2 = m = \sqrt{n}$.

Let A_s be the number of pairs in either sparse-sparse or sparse-dense SubJoins. Algorithm Sparse_Any_Estimation (from Figure 5) computes $\hat{A}_s$ as an estimate for A_s. We denote by $\hat{S}^*$ the set consisting of the (remaining) values in S^* after the (optional) probabilistic elimination step. Let $\mathcal{E}(\tau)$ be the contribution of a tuple τ in S, $\tau \in \text{SubJoin}(v)$, to the estimate $\hat{A}_s$; i.e., $\hat{A}_s = \sum_{\tau \in S} \mathcal{E}(\tau)$:

$$\mathcal{E}(\tau) = \begin{cases} \frac{n}{m_2} \cdot \mathsf{mult}_R(v) & \text{if } \tau \in \hat{S}^* \ \wedge \ \mathsf{mult}_R(v) < \frac{n}{m_2} \\ 0 & \text{otherwise.} \end{cases}$$

We first consider the accuracy and complexity for the strong cost model. We then analyze the performance for the weaker cost model, as a result of the probabilistic elimination step.

Lemma 3.3 *We have $\mathbf{E}(\hat{A}_s) = A_s$, and with high probability $\hat{A}_s = \Theta(A_s) + O(n \lg n)$. Also, the probe cost is $O(\sqrt{n})$.*

Proof. (sketch) If the value of a tuple $\tau \in S$ is dense in R then $\mathcal{E}(\tau) = 0$. Consider a tuple $\tau \in S$, $\tau \in \text{SubJoin}(v)$, such that $\mathsf{mult}_R(v) < n/m_2$. Since $\mathbf{Pr}(\tau \in \hat{S}^*) = m_2/n$, we have $\mathbf{E}(\mathcal{E}(\tau)) = \mathsf{mult}_R(v)$, and hence $\mathbf{E}(\hat{A}_s) = A_s$. Moreover, $\mathcal{E}(\tau)$ is a random variable taken from domain $[0, z]$, where $z = (n/m_2)^2 = n$. By Hoeffding bounds [Hoe63], $\hat{A}_s = \Theta(\mathbf{E}(\hat{A}_s))$ with probability $\exp(-\Theta(\mathbf{E}(\hat{A}_s/z)))$, which is high probability if $A_s = \Omega(n \lg n)$. The probe cost is easily seen to be $O(m)$. ∎

In analyzing the performance of the sparse-any algorithm for the weak cost model, denote by $\mathcal{C}$ the total tuple cost, and let n_d be the number of tuples in S whose value is dense in R.

Lemma 3.4 (probabilistic elimination lemma) *We have $A_s/e \leq \mathbf{E}(\hat{A}_s) \leq A_s$, and with high probability, $\hat{A}_s = \Theta(A_s) + O(n \lg n)$. Also, the expected tuple cost, $\mathbf{E}(\mathcal{C})$, is at most $n_d/e + A_s/\sqrt{n}$.*

Proof. (sketch) Consider a tuple τ in S^* with join attribute value v; i.e., $\tau \in \text{SubJoin}(v)$.

$$\begin{aligned}
&\mathbf{Pr}\left(\tau \text{ is not eliminated from } S^*\right) \\
&\quad = \quad (1 - \mathsf{mult}_R(v)/n)^m \approx e^{-(m/n)\mathsf{mult}_R(v)} \ ,
\end{aligned}$$

assuming (without loss of generality) that $\mathsf{mult}_R(v) \leq n/10$. Thus, since $\mathbf{Pr}(\tau \in S^*) = m/n$,

$$\begin{aligned}
\mathbf{Pr}\left(\tau \in \hat{S}^*\right) &= \mathbf{Pr}\left(\tau \in \hat{S}^* | \tau \in S^*\right) \cdot \mathbf{Pr}\left(\tau \in S^*\right) \\
&\approx \frac{m}{n} \cdot e^{-(m/n)\mathsf{mult}_R(v)} \ .
\end{aligned}$$

Therefore, if $\mathsf{mult}_R(v) < n/m$ then

$$\begin{aligned}
\mathbf{E}\left(\mathcal{E}(\tau)\right) &= \mathsf{mult}_R(v) \cdot (n/m) \cdot \mathbf{Pr}\left(\tau \in \hat{S}^*\right) \\
&\approx \mathsf{mult}_R(v) \cdot e^{-(m/n)\mathsf{mult}_R(v)} \ .
\end{aligned}$$

But since v is sparse, $1/e < e^{-(m/n)\mathsf{mult}_R(v)} \leq 1$ and therefore $\mathsf{mult}_R(v)/e < \mathbf{E}\left(\mathcal{E}(\tau)\right) \leq \mathsf{mult}_R(v)$. (Hence requirement (ii) follows.) As before, Hoeffding bounds can be applied to establish that $\hat{A}_s = \Theta(A_s)$ with high probability, as long as $A_s = \Omega(n \lg n)$.

The probe cost is $O(m)$, as before. The tuple cost is $\sum \mathsf{mult}_R(v)$, where the sum is taken over join attribute values v appearing in tuples in $\hat{S}^*$.

Let $\mathcal{C}(\tau)$ be the contribution of a tuple τ in S to the tuple cost; i.e., $\mathcal{C} = \sum_{\tau \in S} \mathcal{C}(\tau)$. If τ with value v is in $\hat{S}^*$ then $\mathcal{C}(\tau) = \mathsf{mult}_R(v)$; otherwise $\mathcal{C}(\tau) = 0$. Therefore,

$$\mathbf{E}\left(\mathcal{C}(\tau)\right) \approx \mathsf{mult}_R(v) \cdot \frac{m}{n} \cdot e^{-(m/n)\mathsf{mult}_R(v)} \ .$$

We consider the tuples $\tau \in \hat{S}^*$, based on the density of their value v in R. If v is very dense then $e^{-(m/n)\mathsf{mult}_R(v)} \leq e^{-(m/n)(2n \lg n/m)} = n^{-2 \lg e}$ and therefore $\mathbf{E}\left(\mathcal{C}(\tau)\right) \leq m \cdot n^{-2 \lg e} = n^{.5 - 2 \lg e}$. Moreover, by Markov inequality, $\mathbf{Pr}\left(\mathcal{C}(\tau) > 0\right) \leq n^{.5 - 2 \lg e}$. Therefore, the probability that any tuple $\tau \in S^*$ that is very dense in R contributes to the tuple cost is at most $n^{1.5 - 2 \lg e}$. (Hence requirement (i) follows.)

If v is dense but not very dense then $1/n \leq e^{-(m/n)\mathsf{mult}_R(v)} \leq 1/e$ and therefore $\mathbf{E}\left(\mathcal{C}(\tau)\right) \leq 1/e$. (Hence requirement (iii) follows.) Thus the total expected contribution to the tuple cost of dense tuples in R is at most n_d/e.

Finally, for sparse v recall that $e^{-(m/n)\mathsf{mult}_R(v)} \leq 1$. Therefore, $\mathbf{E}(\mathcal{C}(\tau)) \leq \mathsf{mult}_R(v) \cdot (m/n)$, and hence the total expected contribution to the tuple cost of sparse tuples in R is $(m/n) \cdot A_s = A_s/\sqrt{n}$. ∎

Lemma 3.3 and Lemma 3.4 justify the use of a sanity bound $\mathcal{U} = n \lg n$:

Corollary 3.5 *If $\hat{A}_s \geq n \lg n$ then with high probability $A_s = \Theta(\hat{A}_s)$. Otherwise, with high probability $A_s = O(n \lg n)$.*

4 Experimental results

In this section, we report on preliminary experimental comparisons between our bifocal sampling algorithm, the generic adaptive sampling algorithm (described in Section 2.1), and the generic cross sampling algorithm (described in Section 2.4). In order to demonstrate how bifocal sampling overcomes problems with previous algorithms, we present experimental results for the scenario that was considered as "hard" by Haas and Swami [HS92], and which is described in Section 1, of an equi-join $R \bowtie S$ for little-skewed R and highly-skewed S. While the adaptive sampling algorithm and the cross sampling algorithm obtain rather poor results, as is anticipated, the bifocal sampling algorithm obtains excellent results.

We constrained all three algorithms to select the same number of samples from the database and then compute their respective estimates for the size of the join. The percentage error of each of the estimates was then computed by actually computing the join and counting the number of tuples in the output. Specifically, the percentage error is computed as the absolute value of the difference between the estimate and the actual join size, divided by the estimate. Since various implementations of adaptive sampling algorithms differ primarily in their termination criteria, the results are relevant to all those implementations. In particular, previous adaptive sampling algorithms either terminate with fewer samples, and hence obtain no better results than the generic algorithm, or terminate with more samples, and hence run slower than our algorithm.

The size of the equi-join of two relations depends only on the values of the join attribute in each of the input relations. It is therefore sufficient for the purpose of comparing the accuracy of different join-size estimation algorithms to have *single-column* input relations. Accordingly, all the input relations that we used consisted of a single integer-valued join attribute J.

We consider the equi-join $R.J = S.J$ for a range of relations R and S. Each of the input relations in our experiments consisted of $N = 100,000$ tuples. The values of $R.J$ were selected according to a uniform distribution over $[0, 32767]$. The values of $S.J$ were selected according to the distribution function $zipf(10000, \theta)$, where θ was varied from 0.2 to 5.0 in increments of 0.2. These experiments address Haas and Swami's "hard" scenario.

The results are displayed in Figure 6. It is apparent that the accuracy of the adaptive sampling algorithm wildly fluctuates. For many values of θ, it grossly underestimates the actual join size, resulting in a very high error (displayed as an error of 200%, but is in most cases an infinite error). The performance of the cross sampling algorithm is better, yet it also grossly underestimates the actual join size for more than half of the values of θ, and only in a few cases does it provide an error smaller than 60%. In contrast, the estimates obtained by using the new bifocal sampling algorithm are always quite accurate, with errors mostly in the range of 0–3% (except for one case of 8%); they are always more accurate than the estimates of both the adaptive sampling algorithm and the cross sampling algorithm.

The analysis in Section 3 shows that our bifocal sampling algorithm is guaranteed to always give good estimates (within a factor close to 2) with high probability, when the join size is $\Omega(n \lg n)$. This was indeed demonstrated in our experiments, with an indication that the actual relative error may be considerably smaller.

Acknowledgments

We thank Albert Greenberg for participating in the research in its early stages. We thank Jeff Naughton for many helpful comments on this work.

References

[HNS94] P. J. Haas, J. F. Naughton, and A. N. Swami. On the relative cost of sampling for join selectivity estimation. In *Proc. 13th ACM Symp. on Principles of Database Systems*, pages 14–24, May 1994.

[HNSS93] P. J. Haas, J. F. Naughton, S. Seshadri, and A. N. Swami. Fixed-precision estimation of join selectivity. In *Proc. 12th ACM Symp. on Principles of Database Systems*, pages 190–201, May 1993.

[HNSS95] P. J. Haas, J. F. Naughton, S. Seshadri, and L. Stokes. Sampling-based estimation of the number of distinct values of an attribute. In *Proc. 21st International Conf. on Very Large Data Bases*, pages 311–322, September 1995.

[HÖD91] W.-C. Hou, G. Özsoyoğlu, and E. Dogdu. Error-constrained COUNT query evaluation in relational databases. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 278–287, May 1991.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *J. American Statistical Association*, 58:13–30, 1963.

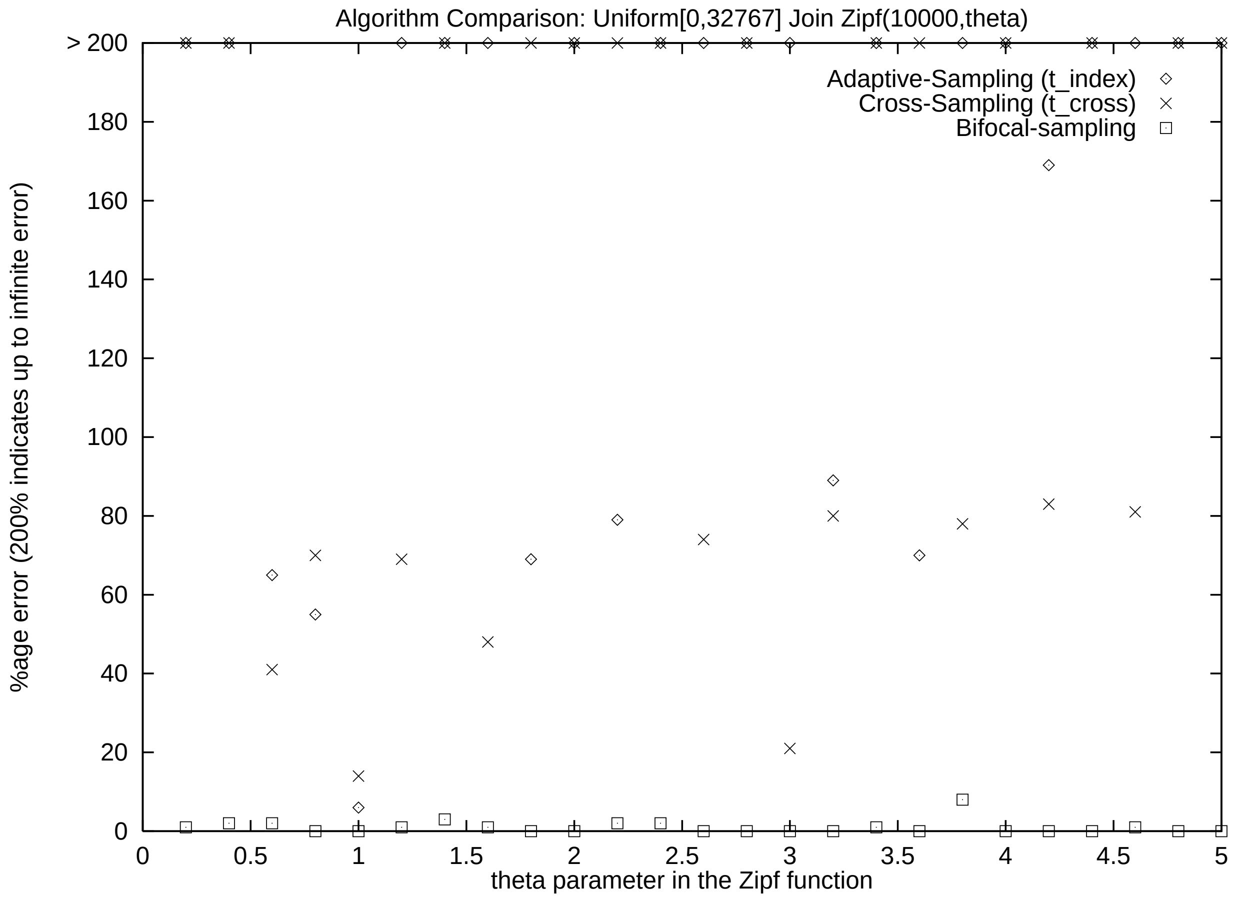

Figure 6: Comparison of our bifocal-sampling algorithm with the adaptive-sampling algorithm and the cross-sampling algorithm. We show the relative errors of the three algorithms for the join of two data sets, one with a uniform distribution over $[0, 32767]$, and the other with the Zipf distribution with parameter θ, for various values for θ between 0.2 and 5.0. Each data set has 100,000 items. For the adaptive-sampling algorithm, there are 17 settings of θ with over 200% error; 14 of these are infinite error. For the cross-sampling algorithm, there are 14 settings of θ with over 200% error; 12 of these are infinite error.

[HÖT88] W.-C. Hou, G. Özsoyoğlu, and B. K. Taneja. Statistical estimators for relational algebra expressions. In *Proc. 7th ACM Symp. on Principles of Database Systems*, pages 276–287, March 1988.

[HÖT89] W.-C. Hou, G. Özsoyoğlu, and B. K. Taneja. Processing aggregate relational queries with hard time constraints. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 68–77, June 1989.

[HS92] P. J. Haas and A. N. Swami. Sequential sampling procedures for query size estimation. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 1–11, June 1992.

[LN89] R. J. Lipton and J. F. Naughton. Estimating the size of generalized transitive closures. In *Proc. 15th International Conf. on Very Large Data Bases*, pages 165–172, August 1989.

[LN90] R. J. Lipton and J. F. Naughton. Query size estimation by adaptive sampling. In *Proc. 9th ACM Symp. on Principles of Database Systems*, pages 40–46, April 1990.

[LN95] R. J. Lipton and J. F. Naughton. Query size estimation by adaptive sampling. *J. Computer and System Sciences*, 51(1):18–25, 1995.

[LNS90] R. J. Lipton, J. F. Naughton, and D. A. Schneider. Practical selectivity estimation through adaptive sampling. In *Proc. ACM SIGMOD International Conf. on Management of Data*, pages 1–12, May 1990.

[LNSS93] R. J. Lipton, J. F. Naughton, D. A. Schneider, and S. Seshadri. Efficient sampling strategies for relational database operations. *Theoretical Computer Science*, 116(1-2):195–226, 1993.

[LS92] Y. Ling and W. Sun. A supplement to sampling-based methods for query size estimation in a database system. *SIGMOD Record*, 21(4):12–15, 1992.