Auto-Tuned Spline Synopses for Database Statistics Management

Arnd Christian König
University of the Saarland
P.O. Box 151150
66041 Saarbrücken, Germany
koenig@cs.uni-sb.de

Gerhard Weikum
University of the Saarland
P.O. Box 151150
66041 Saarbrücken, Germany
weikum@cs.uni-sb.de

ABSTRACT

Data distribution statistics are vital for database systems and other data-mining platforms in order to predict the running time of complex queries for data filtering and extraction. State-of-the-art database systems are inflexible in that they maintain histograms on a fixed set of single attributes, each with a fixed number of buckets regardless of the underlying distribution and precision requirements for selectivity estimation. Despite many proposals for more advanced types of "data synopses", research seems to have ignored the critical tuning issue of deciding on which attribute combinations synopses should be built and how many buckets (or, analogously, transform coefficients, etc.) these should have with a given amount of memory that is available for statistics management overall.

This paper develops a method for the automatic tuning of variable-size spline-based data synopses for multidimensional attribute-value frequency as well as density distributions such that an overall error metric is minimized for a given amount of memory. Our method automatically uses more space for distributions that are harder or more important to capture with good precision. Experimental studies with synthetic and real data demonstrate the viability of the developed auto-tuning method.

1. INTRODUCTION

All database systems and many data-mining platforms keep statistics about their underlying data for a variety of purposes:

First, query optimizers rely on statistics for estimating the selectivity of certain operators. These selectivities matter in the costs of different execution plans for a given query; so accurate estimations are needed for the query optimizer to find a plan with costs not too much above the optimum.

Second, in many situations it is not sufficient to rank different execution plans in the proper order, but it is necessary to predict the absolute sizes of intermediate results and the absolute running times of queries with reasonable accuracy. This is relevant, especially in an OLAP or data-mining environment, to assign proper priorities to long-running queries, ensure that the extracted result data is of proper size that subsequent analysis tools can deal with, avoid queries that would monopolize a server, or even to reconsider whether a long-running data-extraction query is worthwhile at all.

Third, for initial data exploration it may be sufficient for the database system to provide an approximative result for a resource-intensive query, typically for aggregations over huge data sets and ideally with a specified confidence interval. Such approximations could be computed from the data statistics alone rather than scanning all data.

Traditionally, database systems keep statistics only in the simple form of so-called equi-depth [24] histograms with a fixed number of buckets. However, this has widely been recognized as insufficient for reasonably accurate selectivity estimations and other uses of data statistics. In the last decade, a plethora of approximation techniques have been proposed as representations for a database system statistics. These proposals range from advanced forms of histograms (especially V-optimal histograms and including multidimensional variants) [27, 14, 17], sampling [8, 16, 12], and parametric curve-fitting techniques [32, 10] all the way to highly sophisticated methods based on kernel estimators [7], Wavelets and other transforms [21, 20] for range-selectivity estimation, methods based on sampling for approximation of foreign-key joins [3], and other data-reduction techniques [6]. Gibbons et al. have coined the term *data synopsis* as the general concept that abstracts from the variety of all these different representations [13].

All the prior work takes a local viewpoint in that they aim at the best possible synopsis for a single data attribute or multidimensional attribute combination with a given amount of memory (i.e., number of buckets or transform coefficients, etc.). From a global viewpoint, a major problem that has been disregarded so far is how to decide on which attributes or attribute combinations one should build synopses and how much memory should be allocated to each of them with a given total amount of memory that is available for statistics management overall.

1.1 Contribution

The only prior work that has addressed this important tuning problem is [9]. However, this work merely considered the issue of selecting the attributes on which one-dimensional histograms should be kept based on the relevance of such statistics for the decisions of the query optimizer. So a histogram was considered irrelevant if its existence did not

ADVANCES IN DATA MANAGEMENT 2000
Krithi Ramamritham, T.M. Vijayaraman (Editors)
Tata McGraw-Hill Publishing Company Ltd.

make a difference with regard to the optimizer's choice of execution plans. In contrast, our work presented here aims at accurate predictions of the absolute running times, a much more ambitious goal. Also, we substantially generalize the simple yes-or-no decisions of [9] where the same, a priori fixed number of buckets is allocated for each histogram, by allowing a variable number of buckets for the various synopses and tuning these numbers so as to minimize a global error metric.

Our approach is based on the type of linear spline synopses that we studied in [19], but it also would be applicable to standard forms of histograms. Linear Spline synopses reconcile the versatility of histograms with the adaptivity of simple curve fitting limited to linear curves within each bucket (with possible discontinuities at bucket boundaries). The synopsis for a given attribute is constructed from a set of observed (value, frequency) points by a dynamic programming algorithm that determines the optimum partitioning of the data-value domain into a given number of buckets and a least-squares linear fitting within each bucket to minimize the overall approximation error. The algorithm obtains its input points as feedback from the executed queries with a statistical bias on the observed points according to the workload profile, or from an unbiased sampling or full scanning of the actual data.

The current paper generalizes the approach of [19] by allowing different, independent partitionings (i.e., bucket boundaries) for capturing the frequency and the density distribution of an attribute. Furthermore and most importantly, the number of buckets is not a priori fixed for neither of the two distributions and not even for the entire statistics of the attribute. Rather the number of buckets for each attribute and for the two approximations per attribute are automatically determined by a new dynamic programming procedure so as to minimize the overall error. The underlying error metric is a weighted sum of the errors for different classes of estimation problems with weights in proportion to their fractions in the overall workload. Our current method considers estimation errors for range queries, projections and groupings in conjunction with range selections, as well as equijoins over selected value ranges. For the latter class, a special kind of join synopsis is added to the repertoire of synopses.

1.2 Notation

Throughout the paper, we adopt the notation used in [27]. The *domain* $\mathcal{D}$ of an attribute X is the set of all possible values of X, and the value set $\mathcal{V} \subset \mathcal{D}, \mathcal{V} = \{v_1, \ldots, v_n\}$ is the set of values actually present in the underlying relation $\mathcal{R}$. The *spread* s_i of v_i is defined as $s_i = v_{i+1} - v_i$ ($s_n = 0$). The *density* of attribute X in a value range from a through b, $a, b \in \mathcal{D}$, is the number of unique values $v \in \mathcal{V}$ with $a \leq v < b$. The *frequency* f_i of v_i is the number of tuples in $\mathcal{R}$ with value v_i in attribute X. The *cumulative frequency* c_i of v_i is the number of tuples $t \in \mathcal{R}$ with $t.X \leq v_i$. The *data distribution* of X is the set of pairs $\mathcal{T} = \{(v_1, f_1), (v_2, f_2), \ldots, (v_n, f_n)\}$. The *cumulative data distribution* of X is the set of pairs $\mathcal{T}^{\mathcal{C}} = \{(v_1, c_1), \ldots, (v_n, c_n)\}$. The *extended cumulative data distribution* of X, denoted by $\mathcal{T}^{\mathcal{C}^+}$ is $\mathcal{T}^{\mathcal{C}}$ extended over the entire domain $\mathcal{D}$ by assigning zero frequency to every value in $\mathcal{D} - \mathcal{V}$. For the sake of a simple, uniform notation, we extend $\mathcal{T}$ by an artificial final tuple $(v_{n+1}, 0)$ with $v_{n+1} > v_n$. Finally, a *d-dimensional data distribution* in $X_1 \times \cdots \times X_d$ is a set of $d+1$-tuples $\mathcal{T}_d = \{(v_{1,1}, \ldots, v_{d,1}, f_1), \ldots, (v_{1,n}, \ldots, v_{d,n}, f_n)\}$. Its' cumulative extensions are analogous to the one-dimensional case.

The rest of the paper is organized as follows. Section 2 briefly reviews our earlier work on linear spline synopses and fitting procedures [19] as a basis for the generalizations and extensions of the current paper. Section 3 presents the fitting procedure for capturing density distributions with bucket boundaries that are independent of those of the same attribute's frequency synopses. Section 4 introduces the combined error metric for frequency and density approximations and presents the paper's key contribution, the dynamic programming algorithm for tuning the number of buckets of each synopsis. Section 5 extends this framework by a special class of join synopses. Section 6 discusses the CPU costs of our synopsis construction procedures. Section 7 shows how our methods can be carried over to multidimensional distributions. Section 8 presents experimental results that demonstrate the viability of the developed auto-tuning methods for statistics management.

2. APPROXIMATION OF THE ATTRIBUTE-VALUE FREQUENCIES

To approximate a given value-frequency distribution $\mathcal{T}$, we partition the value set $\mathcal{V}$ into m disjoint intervals, called *buckets*, $b_i = [v_{low_i}, v_{high_i})$ in the following manner, where low_i and $high_i$ denote the *subscripts* of the values from $\mathcal{V}$ (i.e., not the actual values) that form the left and right bounds of the (left-side closed and right-side open) value interval covered by the bucket:

$$\forall i \in \{1, 2, \ldots, m-1\}: \quad high_i = low_{i+1},\ low_1 = 1,\ high_m = n+1. \tag{1}$$

Unlike histograms, we approximate the frequency in an interval by a linear function, resulting in a linear spline function [11] over the m buckets. This leads to an improvement in accuracy, depending on the *linear correlation* [29] of the data within a bucket. First, we define $\bar{v}_{[low,high)} := \frac{1}{high-low} \sum_{l=low}^{high-1} v_l$ as the *average attribute value* within $[v_{low}, v_{high})$; analogously, we define the *average frequency* $\bar{f}_{[low,high)} := \frac{1}{high-low} \sum_{l=low}^{high-1} f_l$. The *linear correlation* for bucket b_i is then defined as

$$r_{[low_i,high_i)} := \frac{\sum_{l=low_i}^{high_i-1} (v_l - \bar{v}_{[low_i,high_i)})(f_l - \bar{f}_{[low_i,high_i)})}{\sqrt{\sum_{l=low_i}^{high_i-1} (v_l - \bar{v}_{[low_i,high_i)})^2} \sqrt{\sum_{l=low_i}^{high_i-1} (f_l - \bar{f}_{[low_i,high_i)})^2}} \tag{2}$$

For each interval b_i, $r_{[low_i,high_i)} \in [-1, 1]$. In traditional histograms, the frequency in a bucket b_i is approximated by $\bar{f}_{[low_i,high_i)}$. Using the least-squares fit as an error metric, this results in the overall error

$$f\_err_{[low_i,high_i)} = \sum_{l=low_i}^{high_i-1} (f_l - \bar{f}_{[low_i,high_i)})^2. \tag{3}$$

In a spline-based synopsis, this error becomes :

$$spline\_err_{[low_i,high_i)} = (1 - r_{[low_i,high_i)}{}^2) \cdot f\_err_{[low_i,high_i)}. \quad (4)$$

Summing the error over all buckets in the synopsis, the overall error becomes:

$$ov\_spline\_err = \sum_{i=1}^{m} \big((1 - r_{[low_i,high_i)}{}^2) \cdot f\_err_{[low_i,high_i)}\big). \quad (5)$$

In the following subsections we will develop algorithms for constructing a spline-based synopsis with m buckets for the n observed data-value frequencies in $\mathcal{T}$, aiming to minimize the overall error according to formula 5. This goal involves identifying the $m-1$ most suitable boundaries between buckets and the fitting for the linear approximation within each bucket.

2.1 Fitting the Frequency Function within a Bucket

For the derivation of this basic building block suppose that the boundaries of a bucket are already fixed. For each bucket $b_i = [v_{low_i}, v_{high_i})$ we need to calculate the linear approximation $frq_i(x) = a_1 \cdot x + a_0$ of the attribute frequency that minimizes the squared error

$$spline\_err_{[low_i,high_i)} = \sum_{l=low_i}^{high_i-1} (frq_i(v_l) - f_l)^2. \quad (6)$$

This definition of the error over a bucket is equivalent to definition 4 [29]; however, to evaluate formula 4, the coefficients a_1 and a_0, which are the unknowns of the fitting, do not have to be known. This is the basis for the greedy algorithms introduced in Section 2.3.

Using definition 6, finding frq_i becomes a problem of *linear least squares* fitting [29]; i.e., we have to find coefficients a_0, a_1 s.t. $\frac{\partial spline\_err_{[low_i,high_i)}}{\partial a_0} = \frac{\partial spline\_err_{[low_i,high_i)}}{\partial a_1} = 0$. This problem can be solved using Singular Value Decomposition, for details see [19].

2.2 Optimal Partitioning of $\mathcal{V}$

We are now interested in a partitioning such that the overall error (formula 5) is minimized. When arbitrary partitionings and continuous splines of arbitrary degree are considered, this is known as the *optimal knot placement problem* [11], which – due to its complexity – is generally solved only approximatively by heuristic search algorithms. In our case, however, only linear splines are used and only members of $\mathcal{V}$ are candidates for bucket boundaries. Since the value for each $high_i$ is either low_{i+1} or v_{n+1} (see definition 1), we only need to determine the optimal lower bucket boundaries to compute:

$$f\_opt\_err := \min_{\substack{(low_2,\ldots,low_m)\in\mathcal{V}^{m-1} \\ low_1 \le low2 \le \ldots \le low_m}} \sum_{l=1}^{m} (1 - r_{[low_l,high_l)}{}^2) \cdot f\_err_{[low_l,high_l)} \quad (7)$$

Because the resulting spline function is allowed to be discontinuous over the chosen intervals $b_1, \ldots, b_m$, fitting the data in a bucket can be addressed separately for each bucket b_i. The main improvement in efficiency does, however, result from the fact that the following *principle of optimality* (also known as the Bellman principle) holds for our partitioning problem:

THEOREM 1. *If for $l \ge 2$: $(low_l, low_{l+1}, \ldots, low_m) \in \mathcal{V}^{m-l+1}$ is an optimal partitioning of $[v_{low_{l-1}}, v_{high_m})$ using $m-l+2$ buckets, then $(low_{l+1}, low_{l+2}, \ldots, low_m) \in \mathcal{V}^{m-l}$ is an optimal partitioning of $[v_{low_l}, v_{high_m})$ using $m-l+1$ buckets (the proof can be found in [19]).*

Because of this property, the problem of finding an optimal partitioning becomes a *dynamic programming problem*. This allows us to formulate a redefinition of formula 7: Define

$$\begin{aligned} f\_opt\_err_{low,\bar{m}} &:= \text{optimal overall error for fitting } [v_{low}, v_n) \text{ by } \bar{m} \text{ buckets.} \\ err_{[low,high)} &:= \text{approximation error } spline\_err_i \text{ for buckets } b_i = [v_{low}, v_{high}). \end{aligned}$$

Trivially, $f\_opt\_err_{i,1} = err_{[i,n)}$. Then the overall error produced by the optimal partitioning for buckets 1 through m is

$$f\_opt\_err_{1,m} = \min_{l \in \{1,2,\ldots,n\}} err_{[1,l)} + opt\_err_{l,m-1}. \quad (8)$$

By keeping track of the partitioning, this equation can be used to compute an optimal partitioning in $O(m \cdot n^2)$ time, using $O(n^2)$ space. We refer to this algorithm as *OPTIMAL*.

2.3 Greedy Partitioning

Even if a spline-based synopsis were to be recomputed only occasionally, the cost for computing an optimal partitioning could be unacceptable when n is large. Therefore, we have also developed two greedy methods of partitioning of $\mathcal{V}$, which result in a partitioning that is close to optimal while being much more efficient.

The first technique starts out with a large number (e.g., n) of trivial buckets (e.g., each interval between two adjacent observed data values leads to one bucket), and then gradually merges the pair of adjacent buckets, that causes the smallest increase in overall error. This is repeated, until we arrive at the desired number of m buckets. We will refer to this algorithm as *GREEDY-MERGE*.

The second greedy partitioning-algorithm takes the opposite approach: Initially, all tuples are grouped in one bucket. Now, we will compute the split that leads to the greatest reduction in the overall error (formula 5) and execute it, resulting in an additional bucket. This is repeated, until (after $m-1$ splits) m buckets remain. This algorithm is called *GREEDY-SPLIT*. Both heuristics require $O(n)$ space and compute a close to optimal partitioning in $O(n \log_2 n)$ operations (*GREEDY-MERGE*) or $O(m \cdot n \log_2 n)$ operations (*GREEDY-SPLIT*).

3. APPROXIMATION OF VALUE DENSITY

Accurate approximation of the density distribution is of critical importance for estimation of all queries involving grouping/aggregation. Therefore, we approximate value densities in $\mathcal{V}$ independently from value frequencies, such that: (1) the approximation captures the same number of values as the $\mathcal{V}$ (although it should require less values to represent $\mathcal{V}$) and (2) the squared deviation between the actual attribute

values $v_l \in \mathcal{V}$ and their approximated counterparts $\hat{v}_l$ is minimized. Analogously to the approach for frequency distributions, we partition $\mathcal{V}$ into m' disjoint intervals which we refer to as *density buckets* $db_i = [v_{dlow_i}, v_{dhigh_i})$, with $dlow_i$ and $dhigh_i$ denoting the subscripts of the values from $\mathcal{V}$ that constitute the bucket boundaries. Analogously to formula 1, for all $i \in \{1, 2, \ldots, m'-1\}$ we have: $dhigh_i = dlow_{i+1}$ and $dlow_1 = 1$, $dhigh_{m'} = n+1$. Note that the number of buckets, m', can be chosen independently from the number of buckets, m, for the frequency distribution of the same attribute.

Using the squared deviation of attribute values, the error within bucket db_i is

$$density\_err_{[dlow_i, dhigh_i)} = \sum_{l=dlow_i}^{dhigh_i} \left(v_l - \hat{v}_l\right)^2. \qquad (9)$$

In order to estimate the value distribution within a bucket, we assume that the values are spread evenly over the bucket's width. This means, that the j-th value in a density bucket $db = [dlow, dhigh)$ containing D values is approximated as $\hat{v} = v_{dlow} + (j-1) \cdot \frac{v_{dhigh} - v_{dlow}}{D}$. Thus, denoting the number of values in bucket db_i by D_i, the error of the bucket becomes:

$$density\_err_{[dlow_i, dhigh_i)} =$$
$$\sum_{l=0}^{D_i-1} \left(v_{dlow_i+l} - \left(v_{dlow_i} + \left(l \cdot \frac{v_{dhigh_i} - v_{dlow_i}}{D_i}\right)\right)\right)^2. \qquad (10)$$

To minimize this error we treat the interval's right-end side as a control variable, denoted v_opt_i, whose value should be chosen optimally. Note that the best value of v_opt_i may be different from both the highest attribute value that falls into the bucket and the next higher occurring value, and it may even be smaller than the highest value within the bucket if the density distribution is skewed towards values closer to the interval's lower boundary. The formula for the resulting bucket error then is obtained from equation 10 by simply replacing v_{dhigh_i} with v_{opt_i}:

$$density\_err_{[dlow_i, dhigh_i)} =$$
$$\sum_{l=0}^{D_i-1} \left(v_{dlow_i+l} - \left(v_{dlow_i} + \left(l \cdot \frac{\mathbf{v\_opt_i} - v_{dlow_i}}{D_i}\right)\right)\right)^2. \qquad (11)$$

The parameter v_{opt_i} should be chosen such that the bucket's error is minimized, i.e. $\frac{\partial density\_err_{[dlow_i, dhigh_i)}}{\partial v_{opt_i}} = 0$. Computing the derivative and solving this equation for v_{opt_i} yields:

$$v_{opt_i} = \frac{v_{dlow_i} \cdot \left({D_i}^2 - 1\right) - 6\left(\sum_{l=0}^{D_i-1} l \cdot v_{dlow_i+l}\right)}{2 \cdot {D_i}^2 - 3 \cdot D_i + 1}.$$

The optimal density error for a bucket can now be computed by substituting v_{opt_i} into equation 11. The overall optimal error for the entire attribute-value density synopsis then is the sum of the errors over all buckets $1, \ldots, m'$:

$$d\_opt\_err_{1,m'} = \sum_{i=1}^{m'} density\_err_{[dlow_i, dhigh_i)}. \qquad (12)$$

Finding an optimal partitioning of $\mathcal{V}$ and computing the optimal control parameters v_{opt_i} is mathematically equivalent to the partitioning and per-bucket parameter fitting problem for frequency distributions that we already solved in the previous section. Thus we can apply the same dynamic programming algorithm or, alternatively, one of the greedy heuristics to compute an optimal density synopsis with a given number of buckets. We refer to the resulting error as $d\_opt\_err_{1,m'}$.

4. MEMORY RECONCILIATION

4.1 Reconciling Frequency and Density Synopses for one Attribute

So far we have shown how to minimize the error for each aspect separately, but assumed that the number of buckets, m for frequency and m' for density synopses, is fixed and a priori given for each synopsis. To reconcile both synopses for an attribute, we face the problem of how to divide the available memory space between them, i.e., how to choose the values of m and m' under the constraint that their sum should not exceed a given constant M. Intuitively, we are interested in allocating the memory in such a way that the aspect that is more difficult to approximate is given a higher number of buckets. This means minimizing the combined error of both approximations. However, to make sure that we do not compare apples versus oranges (i.e., value frequencies versus actual data values), we have to take into account that the two approximations refer to domains with possibly radically different scales. Therefore, we first normalize the error metrics for the two classes of synopses and define a relative error for each aspect. As the normalizing factor, we use the highest possible approximation error for each domain, i.e. the maximal difference between the actual value f_i (or v_i) and it's approximation $\hat{f}_i$ (or $\hat{v}_i$ respectively), when the respective domain is approximated using one bucket only. Intuitively, these factors represent the "difficulty" of approximating each domain. We refer to these factors as f_domain and v_domain; the relative error for a single value v_i then becomes $\left(\frac{v_l - \hat{v}_l}{v\_domain}\right)^2 = \frac{(v_l - \hat{v}_l)^2}{(v\_domain)^2}$. The relative error for a single frequency is $\left(\frac{f_j - frq(v_j)}{f\_domain}\right)^2 = \frac{(f_j - frq(v_j))^2}{(f\_domain)^2}$, respectively.

In addition to this normalization it is desirable to consider the possibly different relative importance of the two synopses. With a query workload that is dominated by range or exact-match queries, frequency estimations are obviously the key point. On the other hand, with a workload that requires many estimations of the result size of projection and grouping queries, the density synopses become more important. This consideration is taken into account by attaching a relative weight to each synopsis, i.e., constants w_f and w_d, which are derived from online statistics about the workload and the resulting estimation problems (as discussed in Section 3). Then the combined error for both synopses is the weighted sum of their relative errors.

To divide the available memory, we exploit the fact that computing the optimal partitioning of either approximation type for j buckets by the dynamic programming algorithms of Sections 4 and 5 also generates the optimal partitionings for $1, \ldots, j-1$ buckets at no additional computational cost. The same holds for the GREEDY-SPLIT heuristics; and for GREEDY-MERGE the additional output can be generated at very small extra cost (with j additional merge opera-

tions). Thus, with a given amount of memory for a total of M buckets, we compute the optimal partitionings and resulting relative errors for each approximation with up to $M-1$ buckets, and then divide the available memory in such a way that the sum of the combined relative error terms

$$f\_d\_err = w_f \frac{f\_opt\_err_{1,m}}{(f\_domain)^2} + w_d \frac{d\_opt\_err_{1,m'}}{(v\_domain)^2} \quad (13)$$

is minimized under the constraint that $m \geq 1$, $m' \geq 1$ and $m + m' = M$. This is a trivial combinatorial problem, solvable in $O(M)$ steps.

4.2 Reconciling Synopses for Multiple Attributes

The problem of dividing up the memory between the synopses for multiple attributes, either within the same relation or across multiple relation, is an extension of Section 4.1. Assume that we want to approximate the frequency and density distributions of k different attributes using a total of M buckets. In the following we use the same formalism as before and add superscripts in the range from 1 through k to indicate which attribute a term belongs to (e.g., $low_i^{(j)}$ is the left border of the i-th bucket in the frequency distribution of the j-th attribute). Having k attributes means that there are $2k$ approximations, each of which could be stored in 1 to $M-2k+1$ buckets, assuming that each approximation is given at least one bucket. As before, we compute the partitioning for each approximation for up to $M-2k+1$ buckets. In order to be able to compare the relative errors from different distributions, we again perform the normalization via the factors f_domain or v_domain. We also incorporate attribute- or synopsis-specific weights to reflect the different levels of importance of the various estimation problems. Denoting the weights of the jth attribute's frequency and density synopses by $w_f^{(j)}$ and $w_d^{(j)}$ and the corresponding numbers of buckets by $m^{(j)}$ and $m'^{(j)}$, and enumerating the 2k synopses from 1 through 2k such that the synopses with numbers 2j-1 and 2j refer to the frequency and density of attribute j, we have the following error for the l-th synopsis with m_l buckets 1 through m_l:

$$syn\_err_{l,m_l} =$$

$$\begin{cases} w_f^{(j)} \dfrac{f\_opt\_err^{(j)}_{1,m_j}}{(f\_domain^{(j)})^2} & \text{for } l = 2j-1 \text{ for some } j, 1 \leq j \leq k \\ w_d^{(j)} \dfrac{d\_opt\_err^{(j)}_{1,m'_j}}{(v\_domain^{(j)})^2} & \text{for } l = 2j \text{ for some } j, 1 \leq j \leq k \end{cases}$$

Now, dividing the available memory between all $2k$ approximations means finding the memory allocation $(m_1, \ldots, m_{2k})$, with $\sum_{i=1}^{2k} m_i = M$ such that the overall error

$$ov\_syn\_err := \sum_{l=0}^{2k} syn\_err_{l,m_l} \text{is minimized.} \quad (14)$$

Fortunately, this problem can also be expressed as a *dynamic programming problem*. Define

$$opt\_syn\_err_{i,\bar{m}} := \text{optimal overall error for fitting approximations 1 through } i, \text{ by } \bar{m} \text{ buckets.}$$

Now, $opt\_syn\_err_{2k,M} =$

$$\min_{\mu=1,\ldots M-k+1} \{syn\_err_{1,\mu} + opt\_syn\_err_{2k-1,M-\mu}\},$$

i.e., that the Bellman principle holds, allowing us to compute the optimal memory allocation using $O(M^2)$ space and $O(M^2 \cdot k)$ operations. While a good approximative solution can again be computed using greedy heuristics, it is generally the case that $M \ll n$, and therefore the time for computing the optimal memory partitioning insignificant (see Section 6).

With a simpler version of this technique (i.e. without separate synopses for value frequency and density) it is possible to compute an optimal memory allocation for multiple traditional histograms. Again, it is necessary to compute the errors resulting from each possible size of each histogram beforehand; in case of *V-Optimal* histograms with a different *sort parameter* than attribute frequency (see [17]), this information is already provided by the partitioning routines.

4.3 Using the Synopses for Query Estimation

In the following we briefly describe how to use the resulting auto-tuned synopses for estimating the result sizes of various types of basic queries or even for approximative query answering. We focus on the following query types: projections or, equivalently as far as result-size estimation is concerned, grouping with aggregation, range queries on a single attribute, and joins, with projections and joins possibly restricted to a specified value range (i.e., in conjunction with a range filter).

Projections (and grouping queries): In order to estimate the number of unique values resulting from a projection over the interval $[a, b)$, we differentiate three cases:

- If $[a, b)$ corresponds to the boundaries of a density bucket db_i (i.e., $a = v_{dlow_i}$ and $b = v_{dlow_{i+1}}$), the resulting value distribution is (using the equi-spread assumption within the bucket)

$$\hat{\mathcal{V}} := \left\{ v_{dlow_i} + \left(l \cdot \frac{v_{opt_i} - v_{low_i}}{D_i} \right) \middle| l = 0, \ldots, D_i - 1 \right\}.$$

- If $[a, b)$ is completely contained in density bucket db_i (i.e., $a \geq v_{dlow_i}$ and $b < v_{dlow_{i+1}}$), the resulting value distribution is computed analogously, the only difference being that we consider only values between a and b:

$$\hat{\mathcal{V}} := \left\{ v_{dlow_i} + \left(l \cdot \frac{v_{opt_i} - v_{low_i}}{D_i} \right) \middle| \right.$$
$$\left. l = \left\lceil \frac{(a - dlow_i) \cdot D_i}{dlow_{i+1} - dlow_i} + 1 \right\rceil, \ldots, \left\lfloor \frac{(b - dlow_i) \cdot D_i}{dlow_{i+1} - dlow_i} + 1 \right\rfloor \right\}.$$

- Finally, an approximate projection query for $[a, b)$ that spans more than one density bucket can easily be estimated by clipping $[a, b)$ against the buckets and building the union of the estimations for the intervals $[a, v_{low_j}), [v_{low_j}, v_{low_{j+1}}), \ldots [v_{low_{j+l}}, b)$. Thus, the estimation problem is reduced to the previous formulas.

The result size estimation for this query type then is simply the cardinality of $\hat{\mathcal{V}}$. Whenever approximating values over discrete value domains (i.e., integer domains), we can improve upon the estimation by rounding to the next discrete value.

Range Selections: For this purpose, we define $frequency(x)$ to denote the function described by the spline function over $\mathcal{V}$, i.e.: $frequency(x) = frq_i(x)$ for $low_i \leq x < high_i$. To

estimate the query result for a range query with range $[a, b)$, we compute the approximative density distribution $\hat{\mathcal{V}}$ for $[a, b)$ using the above technique for projection queries, and then compute the frequency distribution of the range query result as $\hat{\mathcal{F}} = \{frequency(\hat{v}) \mid \hat{v} \in \hat{\mathcal{V}}\}$.

When only the number of tuples in $[a, b)$, i.e., the query result size, is to be estimated, the computation can be sped up by summing up $m + m' + 1$ terms (for details see [19]).

Joins: When joining two approximated data distributions we first compute the approximative density distributions $\hat{\mathcal{V}}_1$ and $\hat{\mathcal{V}}_2$ and perform the join between them, resulting in the approximate density distribution $\hat{\mathcal{V}}_{join}$. Then we compute the join result's frequency distribution as $\hat{\mathcal{F}}_{join} = \{frequency_1(\hat{v}) \cdot frequency_2(\hat{v}) \mid \hat{v} \in \hat{\mathcal{V}}_{join}\}$, with $frequency_1$ and $frequency_2$ being the frequency functions of the two join attributes. Note that this procedure is, in principle, general enough to capture not only equijoins (for which the join of the density distributions degenerates into a simple intersection) but also other join predicates such as band, temporal, or even spatial joins. Again, the estimation for the join result size is simply the cardinality of the approximative join result.

4.4 Matching Frequency and Density Synopses

While the above techniques allow us to minimize the weighted sum of the approximation error for attribute value density and frequency, this is not always sufficient to guarantee a low estimation error for the queries described in the previous subsection.

In order to illustrate the problem, we examine a value-frequency pair (v_l, f_l) from a data distribution $\mathcal{T}$ and its approximation $(\hat{v}_l, \hat{f}_l)$. Since we model frequencies as a linear spline functions $frq_i, i = 1, \ldots, m$ over $\mathcal{V}$, we can write $(\hat{v}_l, \hat{f}_l)$ as $(\hat{v}_l, frq_k(\hat{v}_l))$ for some $k \in \{1, \ldots, m\}$. In our approximation of attribute-value frequencies, we minimize the error of formula 6 and thus fit the frequencies at the original attribute values $v_1, \ldots, v_n$, rather than their approximations $\hat{v}_1, \ldots, \hat{v}_n$. In most cases (when v_l and $\hat{v}_l$ belong to the same frequency bucket k, with $frq_k(x) = \alpha_k \cdot x + \beta_k$) this still results in accurate approximation, since the approximation error

$$\begin{aligned} \text{for } f_l &:= |f_l - frq_k(\hat{v}_l)| \\ &= |f_l - frq_k(v_l + (\hat{v}_l - v_l))| \\ &= |\underbrace{f_l - frq_k(v_l)}_{\text{minimized by frequency approx.}} + \alpha_k \cdot \underbrace{(\hat{v}_l - v_l)}_{\text{minimized by density approx.}}| \end{aligned}$$

The problem becomes apparent either for large α_k (resulting in a large second term in the above formula) or when v_l and $\hat{v}_l$ belong to different frequency buckets. In the latter case (demonstrated in Figure 1), the frequency of v_l ist estimated as $frq_k(v_l)$, whereas the frequency of $\hat{v}_l$ by $frq_t(\hat{v}_l)$, $t \neq k$. Now, frq_k is chosen to fit all points in $[low_k, high_k)$, among them (v_l, f_l), optimally, thereby reducing the error $|f_l - frq_k(v_l)|$. The function frq_t is fitted to a different set of points, not including (v_l, f_l), and therefore generally a poor estimator for f_l.

To avoid this problem, we use $\hat{\mathcal{T}} = \{(\hat{v}_1, f_1), \ldots, (\hat{v}_n, f_n)\}$ for approximation of the frequency domain, thereby minimizing the approximation error for the attribute values which will later be used in estimation. However, $\hat{\mathcal{T}}$ depends on the number of buckets m' used for approximation of the attribute value density, which in turn depends on the error $f\_opt\_err_{1,m}$ (see formula 8) made when fitting $\hat{\mathcal{T}}$. A straightforward solution would be to compute the resulting $\hat{\mathcal{T}}$ for all possible $M - 2k + 1$ values of m', fit them, and then determine the optimal combination of density and frequency error. This, however, would increase the computational complexity of our approach by the factor of M. Therefore, we use a simpler approach. First, we compute $f\_opt\_err_{1,m}$ for fitting $\mathcal{T}$ and use it to determine the corresponding numbers of buckets, m, m', for frequency and density approximation. Then we compute $\hat{\mathcal{T}}$ for m', again compute the optimal approximation of the value frequency, and use the resulting spline as the final synopsis. Thus we only compute the optimal partitioning of $\mathcal{V}$ for value frequency approximation twice, instead of M times. Furthermore, since the $f\_opt\_err_{1,m}$ for fitting $\mathcal{T}$ and for fitting $\hat{\mathcal{T}}$ are generally quite similar, almost all times both approaches result in the same values for m and m' and therefore in identical approximations.

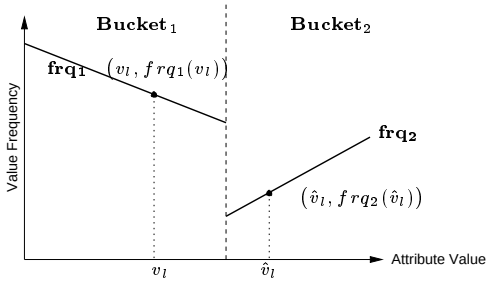

Figure 1: The need for matching approximations

5. APPROXIMATION OF JOIN RESULTS

The kinds of synopses developed so far also allow us to estimate the result of an equijoin between two tables with given frequency and density synopses for the two join attributes. However, the join over two approximated data distributions is often very different from an approximate representation (e.g., a sample) of a join of the underlying complete distributions [8, 3]. Therefore we extend our repertoire of data synopses and adopt the idea of [3] to capture important join-result distributions in a special class of join synopses. Because the main difficulty in estimating join-result sizes lies in the accurate approximation of the density distribution (whereas the frequency distribution can be computed with decent accuracy from the frequency synopses of the base relations), we introduce special *join-density* buckets that capture the density (i.e., existing unique values) of the data values that are common to both join attributes and are thus present in the join result.

5.1 Join Synopses and their Error Metric

First of all we have to define an error metric for a join synopsis, i.e., the error between a join result J and its approximation $\hat{J}$. To this end we have to take into account both the deviation in the individual tuple frequencies and the deviation in the attribute density distribution of the join-attribute values that are present in the join result (i.e., are not dangling values), and also the fact that the approximated and the actual join result may contain a different number of

unique attribute values. A priori it is unclear how to match the the values in the approximated join with the values that occur in the actual result. Simply matching the i-th value in each distribution is not satisfactory, even if J and $\hat{J}$ contained the same number of values. To illustrate this point, consider the case where $J = \{(v_1, f_1), \ldots, (v_{n-1}, f_{n-1})\}$ and $\hat{J} = \{(v_2, f_2), \ldots, (v_n, f_n)\}$, which would incur a large error if tuples were simply matched pairwise in the above orders, although the join result and its approximation match perfectly except for one value. Because of this, we match individual tuples by their join-attribute-value distances to compute the error between the approximation and the actual result. For tractability we do this in a greedy fashion:

1. First, match all tuples of equal attribute values, i.e. all $(v_a, f_a) \in J, (\hat{v}_b, \hat{f}_b) \in \hat{J}$ where $v_a = \hat{v}_b$.

2. Then, match the tuples $(v_a, f_a) \in J, (\hat{v}_b, \hat{f}_b) \in \hat{J}$ where the distance between v_a and $\hat{v}_b$ is minimal. This is repeated until only tuples from either J of $\hat{J}$ remain.

3. Finally $||J| - |\hat{J}||$ unmatched tuples remain. Each is matched with an artificial value with frequency 0, located in the center of the active domain of the join attribute.

Using a merge-based approach, it is possible to compute this matching efficiently: Initially, both J and $\hat{J}$ are merged into a single list sorted by the attribute value of each tuple; in this process, all exact matches are eliminated (step 1). For each matching in step 2, we only consider the up to $2 \cdot \min\{|J|, |\hat{J}|\}$ distances between tuples adjacent in the resulting list, and coming from different sets; these are stored in a *priority queue*. Every time a pair of tuples is matched, these are eliminated from the list, and at most one new distance needs to be computed. Therefore the computational complexity of the three steps is (with $p = \min\{|J|, |\hat{J}|\}$ and $l = \max\{|J|, |\hat{J}|\}$): $O(p)$ (steps 1&3) and $O(l \log_2 l)$ (step 2). All steps have to be executed once for each different allocation of join-buckets, so if M denotes the overall number of buckets and k the number of different attributes in the approximation, the matching has to be carried out $M - 2 \cdot k$ times, resulting in the overall complexity of $O((M-k) \cdot (p + l \log_2 l))$.

Once we have this matching, we compute the overall error analogously to the error for frequency and density distributions, namely, by computing the relative difference in value and frequency for each matching pair of tuples. The normalizing factors *f_join_domain* and *v_join_domain* are computed as before (see Section 4.1), on the basis of one bucket being used for approximation of the respective domain.

$$join\_error :=$$

$$\sum_{\text{all matched tuples } (a,b)} \left(\frac{v_a - \hat{v}_b}{v\_join\_domain}\right)^2 + \left(\frac{f_a - \hat{f}_b}{f\_join\_domain}\right)^2$$

$$+ \sum_{\text{all unmatched tuples } q \text{ in } J} \left(\frac{v_q - (v_{|J|} - v_0)/2}{v\_domain}\right)^2 + \left(\frac{f_q - 0}{f\_join\_domain}\right)^2$$

$$+ \sum_{\text{all unmatched tuples } \hat{q} \text{ in } \hat{J}} \left(\frac{\hat{v}_{\hat{q}} - (v_{|J|} - v_0)/2}{v\_domain}\right)^2 + \left(\frac{\hat{f}_{\hat{a}} - 0}{f\_join\_domain}\right)^2$$

5.2 Integrating Join Synopses into the Auto-Tuning Method

Integrating dedicated join synopses for selected equijoins into our framework again means solving the problem of how to divide memory between all types of approximations. In the following, we explain how to determine whether a join synopsis for a particular equijoin $J^{(c,d)}$ between the relations c and d on their common attribute J is worthwhile to be added to the overall statistics and how many buckets this join synopsis should have. As before, we assume that there is memory for a total of M buckets. Further, let $\mathcal{V}_{join}$ denote the density distribution of $J^{(c,d)}$.

We assume that we have already solved the problem of dividing up the memory for the density and frequency synopses for all k attributes (see Section 4.2). So we also know the optimal partitionings of each attribute and corresponding errors for all numbers of buckets (overall) less or equal to M. Now we have to compute the error resulting from computing the join using no additional join synopsis. For this purpose, we compute the approximate join $\hat{J}^{(c,d)}$ using the techniques of Section 4.3 and then compute the resulting error, which we refer to as *standard_syn_join_err*.

Furthermore, for $l = 1, \ldots, M - 2k$, we compute the join approximation resulting from using l buckets for an additional join synopsis as follows. For each l, we compute the approximate $\mathcal{V}_{join}$ using l buckets, and the resulting approximate density distribution $\hat{\mathcal{V}}_l$. Then we compute the resulting frequency distribution using the frequency approximation of the base relations when only $M - l$ buckets are used overall. For each resulting join approximation we compute its error, which we refer to as $join\_syn\_join\_err_l$. Also, we define $join\_syn\_join\_err_0 := standard\_syn\_join\_err$.

Since we essentially use the same error metrics for join-result and base-data distributions, we can directly compare the error terms. Then, using the relative weight $w_j^{(c,d)}$ to denote the importance of the particular join and its result-size estimation, the problem of partitioning the available memory again means finding l such that the overall error

$$final\_syn\_err :=$$

$$opt\_syn\_err_{2k,M-l} + w_j^{(c,d)} \cdot join\_syn\_join\_err_l \text{ is minimal.}$$

For considering multiple joins, we extend the optimization analogously to Section 4.2; the problem of partitioning the available memory M for $2k$ density and frequency distributions and h possible joins is again a dynamic programming problem, solvable in time $O(M^2 \cdot (k + h))$.

6. RUNNING TIMES

To assess the CPU costs of our algorithms, we measured the running times of approximating the frequency and density distributions as well as a self-join result for a single attribute and different sizes of n and M (and randomly chosen v_i and f_i values) for execution on a single processor of a *SUN UltraSPARC 4000/5000* (168 MHz), shown in Table 1. We measured the CPU time used for partitioning of the frequency (*F-part*) and density domain (*D-part*) for all partitioning methods (*Optimal, Greedy-Merge and Greedy-Split*). Computing the join synopsis also entails computing the join-error (*J-error*) for all possible bucket combinations (see Section 5.1). Finally, *M-part* gives the time used for rec-

Method	Step	$n =$	500		1000		4000	
		$M =$	10	50	10	50	10	50
Optimal	F-part.		0.38	1.99	1.98	10.27	51.16	264.63
	V-part.		0.38	1.00	1.97	5.93	41.35	133.43
G-Merge	F-part.		0.012	0.018	0.023	0.033	0.097	0.141
	V-part.		0.008	0.009	0.018	0.020	0.084	0.086
G-Split	F-part.		0.005	0.017	0.012	0.046	0.068	0.268
	V-part.		0.006	0.023	0.016	0.058	0.083	0.271
All	J-error		0.060	0.681	0.256	1.45	1.21	6.41
	M-part.		0.000031	0.00325	0.000031	0.00325	0.000031	0.00325

Table 1: Running times in seconds

onciling the sizes of all attributes (see Section 4.1 and 5.2).

7. MULTIDIMENSIONAL DATASETS

The result size of complex queries involving multiple correlated attributes depends on these attributes' *joint data distribution*. While most commercial systems assume individual attributes to be uncorrelated and maintain approximations on individual attributes only, this has been shown to result in extremely inaccurate result size estimations for multidimensional exact-match or range queries as well as join queries [26]. In response, a number of techniques for approximating distributions over multiple attributes have been proposed, which can be separated into two categories: *multidimensional histogram techniques* and *transform-based techniques*.

The latter class approximates multi-dimensional frequency distribution only, representing either $\mathcal{T}_d$ or $\mathcal{T}_d^{\mathcal{C}^+}$ via a discrete Wavelet [21] or cosine decomposition [20]. While this leads to accurate estimations of range-selectivity queries, these techniques do not approximate the attribute-value distribution at all and are therefore unsuitable for (complex) queries involving projections/aggregations or joins.

Multidimensional histogram techniques (which were introduced in [26] and refined by [15, 4, 2]) show similar problems. While accurate for range-selectivity estimation, histograms incur significant errors when approximating projection and join queries. This is because of the way the attribute-value domain is represented within buckets: multidimensional histograms assume values within a bucket to be spread *evenly in every dimension*, meaning that if the number of unique attribute values for each attribute $l \in \{1, \ldots, d\}$ is denoted by $values_l$, then all $\prod_{l=1}^{d} values_l$ combinations of attribute values are assumed to be present; the cumulative frequency in each bucket is then distributed evenly over all combinations. This generally results in tremendous overestimation of the number of distinct attribute values (and correspondingly underestimation of their frequencies). While range selectivity queries may still be estimated with decent accuracy, all queries dependent on accurate approximation of the attribute-value domain (i.e. almost all queries involving joins or projections/aggregations) are not captured well at all. This effect becomes worse with rising dimensionality, as data becomes more sparse.

A further problem with the use of multi-dimensional histograms is the complexity of computing an optimal partitioning of the attribute-value domain for most suitable error-functions. In case of *V-optimal* Histograms, which offer the best performance for one-dimensional distributions and optimize the error defined by formula 3, the problem of computing the optimal partitioning using arbitrary buckets becomes NP-hard even for $d = 2$ [23].

7.1 Combining Splines Synopses and Space-Filling Curves

A straightforward extension of our spline techniques to multiple dimensions is not a viable alternative as it would encounter the same problems associated with histogram-based techniques. Rather, we use the following approach. First, an injective mapping $\theta : \mathbb{R}^d \mapsto \mathbb{R}$ is applied to the attribute-value domain of the base data, reducing the d-dimensional distribution to a one-dimensional one, which is then approximated by the techniques introduced previously. We store the resulting approximation; when it is used to estimate a multi-dimensional query, the approximated data is mapped back to $\mathbb{R}^d$ via the inverse mapping $\psi := \theta^{-1}$. For ψ, we use space-filling curves [30], which are continuous, surjective mappings $\psi : [0,1] \mapsto [0,1]^d$. Originally devised in topology, they have been used for a number of applications, among them heuristic solutions to the *Traveling Salesman Problem* [25] , partitioning problems in VLSI design [5] and multidimensional indexing [18].

The key for this approach is that distances in $\mathbb{R}$ are preserved by ψ (i.e., if a point x and its approximation $\hat{x}$ are close in $\mathbb{R}$, $\psi(x)$ and $\psi(\hat{x})$ are close in $\mathbb{R}^d$) , so that minimizing the error in $\mathbb{R}$ leads to a low error in approximation of the original data domain as well. This approach also leverages the advantages of our spline techniques (efficiency, accurate representation of both attribute-value and attribute-frequency domains, fast computation). Figure 2 shows the interaction between the spline synopses and space-filling curves.

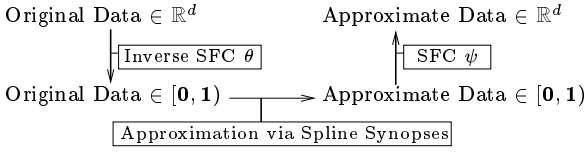

Figure 2: Combining Space-Filling Curves (SFC) and Spline Synopses

There are a number of space-filling curves, most of which require low computational overhead and some of which can be extended to arbitrary dimensions d. For our implementation, we have chosen a variation on a curve construction due to Sierpiński [31]. This particular curve is constructed by successively partitioning $[0,1]^d$ as shown in Figure 3 and marking the *endpoints* of each sub-partitioning with the corresponding position along the curve in $[0,1)$ (shown for

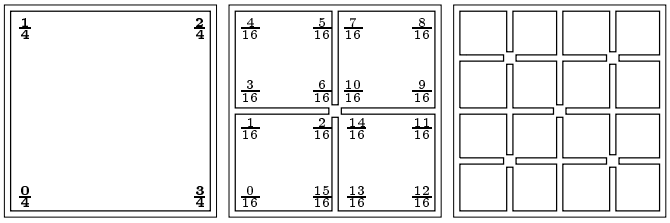

Figure 3: The Sierpiński curve for $d = 2$ and 2, 4, and 6 iterations of the recursive construction

the first two partitionings). A Sierpiński curve computed by t iterations of the curve construction projects n tuples of the d-dimensional distribution onto the set $\left\{\frac{p}{2^t} \mid p = 0, \ldots, 2^t - 1\right\}$.

This curve can be extended to arbitrary dimensions, leads to ψ and θ functions that can be evaluated efficiently (the resulting algorithm can be found in [5]) and furthermore has three salient properties setting it apart from other space-filling curves:

- It contains no "jumps", i.e., for arbitrary $x, x' \in \mathbb{R}$, the ranges $[x, x+\delta]$, $[x', x'+\delta]$ correspond to pieces of the curve of identical length.
- It is symmetric (unlike the *Hilbert or Peano curve*), i.e. invariant to rotations by multiples of 90 degrees in the direction of each dimensional axis. This means that the approximation quality is independent of which dimension we map each approximated attribute to.
- The resulting projection ψ preserves distances very well. For example, in 2-dimensional space the following holds: if we denote the distance of two points $a, a' \in [0,1]^2$ by the Euclidean distance $D[a, a']$ and of two points $x, x' \in [0,1]$ along the curve by $\Delta[x, x'] := min\{|x - x'|, 1 - |x - x'|\}$ (this is the natural metric since the curve is a closed loop with $\psi(0) = \psi(1)$), then

$$\forall x, x' \in [0,1) : D[\psi(x), \psi(x')] \leq 2\sqrt{\Delta[x, x']}.[25]$$

The number of iterations t of the recursive partitioning depends on the data itself. In discrete value domains we need to make sure that t is large enough so that the curve passes over every point in the underlying data. Since the Sierpiński space-filling curve partitions all dimensions symmetrically, we only have to make sure that the dimension $\tilde{d}$ with the biggest overall value spread (i.e. $total\_spread_{\tilde{d}} := \max_{i=1,\ldots,n} v_{\tilde{d},i} - \min_{i=1,\ldots,n} v_{\tilde{d},i}$ is maximal) is captured correctly. The number of distinct values in one dimension captured by a Spierpiński curve constructed by t partitionings is $values(t) := 2^{\frac{t}{d}-1} + 1$. Therefore we choose the number of partitionings t so that the number of distinct points in each dimension is larger than $total\_spread_{\tilde{d}}$: $\tilde{t} := \left\lceil (\frac{\ln(total\_spread_{\tilde{d}}-1)}{\ln 2} + 1) \cdot d \right\rceil$. In continuous value domains, t depends upon the desired accuracy.

Finally, before applying θ, we have to map all tuples to $[0,1]^d$; this means subtracting the minimal value $min_p := \min_{i=1,\ldots,n} v_{p,i}$ for each dimension p from all values in that dimension and dividing the result by $values(\tilde{t}) - 1$: $v_{p,j} \mapsto \frac{v_{p,j} - min_p}{values(\tilde{t})-1}$, for $p = 1, \ldots, d$, $j = 1, \ldots, n$. Mapping the resulting points to $[0,1)$ requires $O(d \cdot t)$ operations per point; so the overal complexity is $O(n \cdot d \cdot t + n \log_2 n)$ (the tuples have to be sorted after mapping) with t, d being small constants.

Reconstruction of the original values requires mapping the approximated values via ψ onto $[0,1]^d$ and reversing the normalization described in the previous paragraph. d-dimensional ranges specified for projections or range selections have to be mapped to the corresponding ranges in $[0,1]$, which involves computing all intersections of the Sierpiński curve with the surface of the specified range and deciding which of the corresponding intervals are included in the query range (see [22]). Fortunately, since synopses typically reside in main memory, even brute-force approaches that explicitly map all tuples in the approximation back to the original domain and then discard those not satisfying the selection criteria are sufficiently fast.

8. EXPERIMENTAL EVALUATION

8.1 Experimental Setup

The Techniques: We compare the spline synopses with the following existing histogram techniques: *equi-width* and *equi-depth* [24], *MaxDiff(V,A)* [28], and *V-optimal(V,F)* [17] histograms for one-dimensional data and multidimensional *MaxDiff(V,A)* histograms constructed via MHIST-2 (which performed best in [26]) for multidimensional data. The storage requironments are 3 values (number of distinct attribute values, average frequency and largest attribute value) for a one-dimensional histogram bucket and $3d+1$ values for a d-dimensional one (overall frequency, and for each dimension: low and high values and number distinct values in that dimension) .

For workloads consisting solely of range selections, we also examine the accuracy of *discrete transform techniques* based either on the Wavelet transform [21], using Haar Wavelets, and the discrete cosine transform (DCT) [20] (because both techniques do not approximate the attribute-value density, they are not suitable for projection and join approximation). To determine which Wavelet coefficients to keep for the first technique, we used the first thresholding method introduced in [21], keeping the largest M coefficients for the experiments shown here (the other three thresholding methods show only insignificant changes in the approximation accuracy for the data sets used here). In case of the discrete cosine transform, we select coefficients by reciprocal sampling, which was shown to perform best amoung all filtering methods presented in [20]. For both techniques we store 2 values for each kept coefficient, storing the coefficient's value and its index/position (in case of multidimensional data, the multidimensional index is coded as a single value).

These are compared with Spline Synopses computed using the *OPTIMAL, GREEDY-MERGE* and *GREEDY-SPLIT* partitioning techniques. The storage overhead is 3 values per bucket for the frequency approximation (lowest value, coefficients of the frequency-function) or density approximation (lowest value, number of unique values, *v_opt*) for one-dimensional datasets. In case of multidimensional data,

$d+1$ additional values must be stored to facilitate the mapping to and from $[0,1]$: $values(\tilde{t})$ and min_p for $p = 1, \ldots, d$ (see Section 7.1). We do not tune the spline synopses for each estimation task, but use constant relative importance weights $w_f = 1, w_d = 1, w_j^{(1,1)} = 1$ throughout all experiments.

Datasets: Out of our experiments carried out with one-dimensional distributions, we present a subset for three representative datasets: two real-life datasets provided by the KDD Data Archive of the University of California [1] and a synthetic one serving as a stress-test for the different spline techniques. The two real-life data sets are single attributes of the *Forest CoverType* data set (581012 tuples, 54 attributes, 75.2MB data size) of which we use (1) the attribute *elevation* (1978 unique values, medium distortion of frequency and density distribution) and (2) the attribute *horizontal_distance_to_hydrology* (189 unique values, small distortion of frequency and density distribution). Finally, we use a synthetic data set with randomly chosen frequencies (uniform distribution) and regular density distribution (1000 unique values). Because there is no correlation between adjacent frequency-values, this results in buckets with very small linear correlation r (see formula 2) and constitutes a "worst-case" scenario for the frequency approximation by linear splines. The datasets are visualized in Figure 8.1.

Out of our experiments on multidimensional distributions we present the results for two data sets: (1) 1970 PUMS Census data, of which we use the attributes *migplac5* and *chborn* (661 unique attribute-value pairs, 70817 tuples), again provided by the KDD Archive [1] and (2) a synthetic stress-test similar to the one used before, with 400 unique value pairs, 10000 tuples, and randomly chosen density distribution and attribute-value frequencies.

Query Workload: For the single attribute data sets we first compute the approximation for the given size, which is then used for three different estimation tasks: **(a)** estimating the size of a self-join, **(b)** estimating range-selection queries of the type $\{X < b \mid b \in \mathcal{V}\}$ and **(c)** estimating the number of unique values for ranges $\{X < b \mid b \in \mathcal{V}\}$.

For multi-dimensional data sets we computed **(a)** the estimation error for range and **(b)** projection queries, both for ranges $\{(X_1, \ldots, X_d) < (b_1, \ldots, b_d) \mid (b_1, \ldots, b_d) \in \mathcal{V}_d\}$.

In each case we measure the mean squared error

$$\textbf{MSE} := \frac{1}{n} \sum_{i=1,\ldots,n} (exact\_size_i - approx\_size_i)^2,$$

with n being the number of queries.

8.2 Experiments on One-dimensional data

The results of the first two experiments on one-dimensional data are shown in Figures 5 – 7. For projection and self-join queries, the *optimal* spline techniques consistently outperform all other techniques, followed closely by the *greedy* spline variants (the sole exception being *V-Optimal(V,F)* histograms, which outperform spline synopses constructed via GREEDY-MERGE for self-join estimation on the first dataset). Among histogram techniques, the *V-Optimal(V,F)* histograms perform best, for which the partitioning algorithm is of the same asymptotic complexity as the *OPTIMAL* spline partitioning. For range-selections, the spline techniques are again are more accurate than histograms and the estimation by Wavelets; however, when 40 or more values are kept, the DCT technique performs slightly better.

When comparing spline synopses to the DCT/Wavelet techniques, we have to consider that these techniques are geared specifically to range-selectivity estimation and can not be used to estimate queries depending on the attribute-value density of a dataset.

The results for estimating the "worst-case" data are shown in Figure 8 (we omitted the results for projection estimation since all techniques capture the density domain accurately). Again, the *optimal* and *greedy-split* spline techniques outperform all competitors other than the DCT technique; since no linear correlation can be exploited, the DCT technique exhibits major gains in this particular experiment.

8.3 Experiments on Multi-dimensional Data

The results of the two experiments on multi-dimensional data are shown in Figures 9 and 10. For range-selectivity estimation the histogram-based techniques consistently outperformed the multidimensional *MaxDiff(V,A)* histograms. Wavelets turned out to be superior to splines for range queries for one experiment and the DCT based technique was the winner in the other experiment. However, the winning technique in one experiment performed very badly in the other one.

The performance of both transform-based techniques depended very much on the sparseness of the data. With approximating data via Haar Wavelets, as used in the experiment, very sparse data results in a large number of coefficients of value 0, which can therefore be dropped without increasing the estimation error. Consequently, the Wavelet technique results in very accurate approximation for sparse data. The opposite is true in dense domains, however; here all other techniques perform better. The two datasets are examples for this behavior. While the Census data contains 661 unique value pairs in a domain of $715 \cdot 13 = 9295$ value pairs, the synthetic data contains 400 value pairs in a domain of $40 \cdot 40 = 1600$ value pairs.

Regarding projection estimation, multi-dimensional histograms were consistently outperformed by the spline techniques by very large margins. Further experiments (not shown here for lack of space) confirmed this observation and indicated a trend that this effect increases with dimensionality.

8.4 Experiments on Memory Reconciliation

These experiments serve to examine how our approach for dividing the available memory between multiple synopses can improve estimation accuracy. We used the three one-dimensional relations with the data distributions shown in Section 8.1, with a total memory size M available for all synopses all together. We performed three experiments: (a) range-queries on all three relations, (b) projections on all three relations, and (c) self-joins on all three relations. The operations on the various relations were equally weighted within each of the three experiments.

We compare the overall estimation accuracy for all three data sets for different ways of dividing up the available memory: (i) the synopses reconciliation technique introduced in Section 4.2, which we refer to as *Adaptive Memory Reconciliation*, (ii)giving the same amount of memory to every

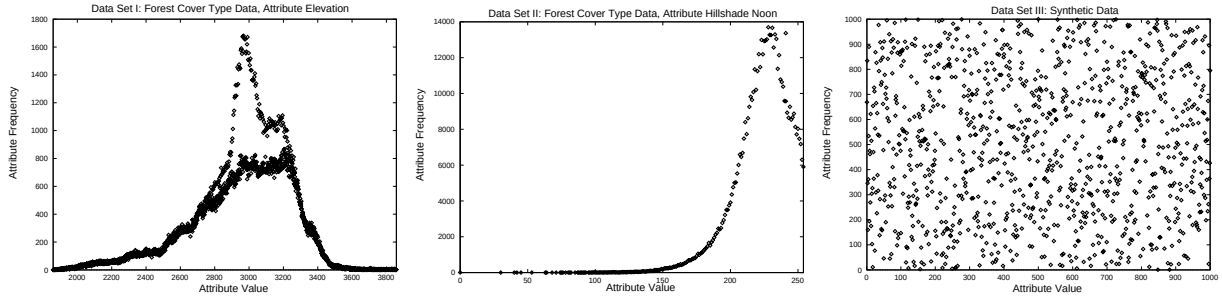

Figure 4: The one-dimensional datasets used in the experiments

synopsis, referred to as *Equi-Size*, (iii)assigning the memory relative in proportion to the sizes of the underlying data distributions, referred to as *Data-Size*, and (iv) assigning the memory relative in proportion to the number of different attribute values in the underlying data distributions, referred to as *Data-Values*. To examine the overall accuracy for all three approximations, we measure the error as a fraction of the actual query result size ($relative\_error := \frac{|exact\_size - approx\_size|}{exact\_size}$) and then compute the average error the three relations. Figure 11 shows the error for range queries weighted with equal weights over all three relations. Especially for small memory sizes M, the adaptive memory reconciliation technique leads to an impressive reduction of the overall error. The results were similar for projection and self-join estimation (which are omitted). In order to show that adaptive memory reconciliation is not limited to spline synopses, we repeated the experiment for *V-optimal(V,A)* histograms, using the synopses sizes obtained previously. Again the reconciliation technique clearly outperforms all others (Figure 11). We repeated these experiments for a number of different combinations of data sets and workloads, resulting in similar observations.

8.5 Discussion of our results

In summary our experiments underlined the viability and salient properties of the advocated memory adaptive spline synopses. More specifically, the following insights were experimentally substantiated:

- Spline synopses consistently outperformed the best histogram techniques, for both one-dimensional and multi-dimensional distributions.
- For range queries, spline synopses were competitive with the best specialized[1] techniques, namely, Wavelet or DCT based techniques. Note that among the latitude techniques there was no consistent winner: both performed excellent in some experiments and badly in others.
- Like histograms and unlike transform-based techniques, spline synopses are extremely versatile in that they can cope with the full spectrum of query types. Also they are robust in that they yield decent estimates even for query types and data that are hard to capture, while performing excellent in most, well-behaved, cases.
- The proposed method for adaptive memory reconciliation underlined its practical viability by clearly outperforming various simple heuristics for dividing memory among multiple synopses. This method is orthogonal to the actual approximation technique, which was demonstrated by applying it to both spline synopses and histograms.

[1]Preliminary experiments indicate that specializing and optimizing spline synopses solely for range selectivity estimation results errors comparable to the best transform-based techniques. This can be done by interpreting f_l as weights to the approximation of the $density\_err$. We then minimize the function $density\_err_{[dlow_i, dhigh_i)} = \sum_{l=dlow_i}^{dhigh_i} f_l \cdot (v_l - \hat{v}_l)^2$, thereby ensuring that frequent attribute values are fitted better than infrequent ones.

9. CONCLUSION

This paper has aimed to improve the flexibility of a database system's statistics management by incorporating frequency, density, and equijoin synopses with a variable number of buckets into a common framework. The dynamic programming algorithms developed in this paper allow us to choose the number of buckets for each of these synopses such that the overall error for the estimation problems posed by a query optimizer or query scheduler becomes minimal. The presented experiments clearly demonstrate the viability of the developed approach towards auto-tuned statistics management. Our long-term vision is to unify the plethora of data approximation techniques that have been proposed for the purpose of selectivity estimation and approximative query answering. The practical incentive for this is to enhance database systems and data-mining platforms with corresponding auto-tuning capabilities, or at least provide administrators with explicit guidance on how to reconcile the various approximation techniques and their resource requirements.

10. REFERENCES

[1] The UCI KDD Archive (http://kdd.ics.uci.edu).

[2] A. Aboulnaga and H. F. Naugton. Accurate Estimation of the Cost of Spatical Selections. In *Proceedings of the IEEE Conference on Data Engineering*, pages 123–134, 2000.

[3] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. Join Synopses for Approximate Query Answering. In *Proceedings of the ACM SIGMOD Conference*, pages 275–286. ACM Press, 1999.

[4] S. Acharya, V. Poosala, and S. Ramaswamy. Selectivity Estimation in Spatial Databases. In *Proceedings ACM SIGMOD Conference*, pages 13–24. ACM Press, 1999.

[5] C. J. Alpert and A. B. Kahng. Multi-Way Partitioning Via Geometric Embeddings, Orderings and Dynamic Programming. *IEEE Transactions on CAD*, 14(11):1342–1358, 1995.

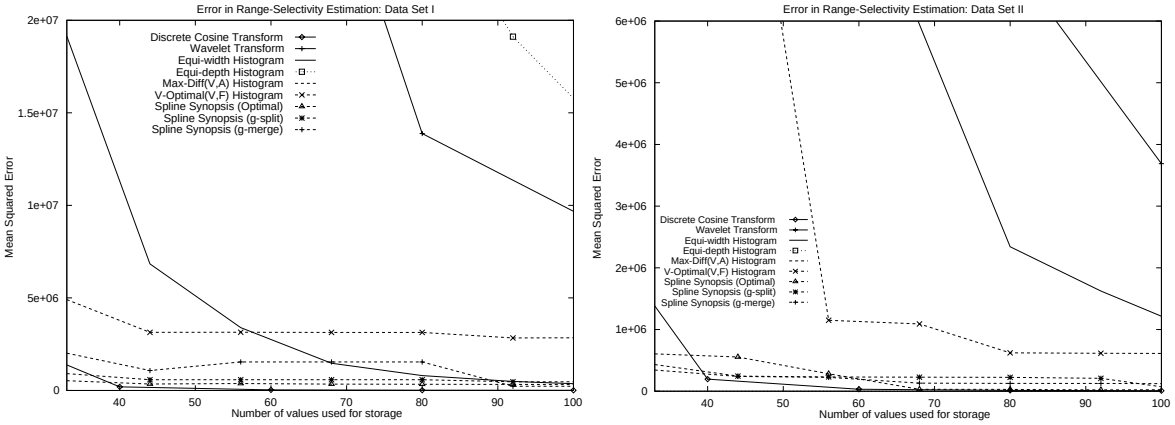

Figure 5: Accuracy of different techniques for range queries on real-life data

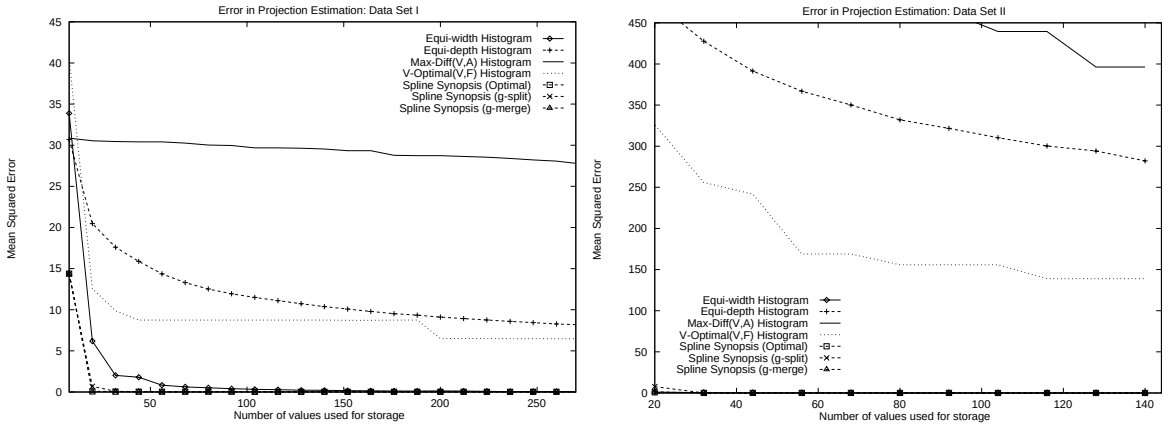

Figure 6: Accuracy of different techniques for projection queries on real-life data

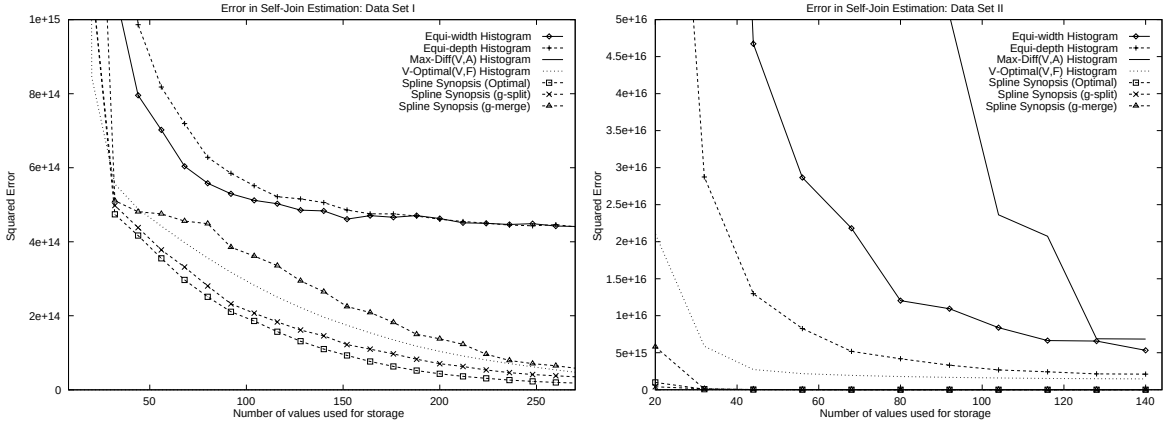

Figure 7: Accuracy of different techniques self-join estimation on real-life data

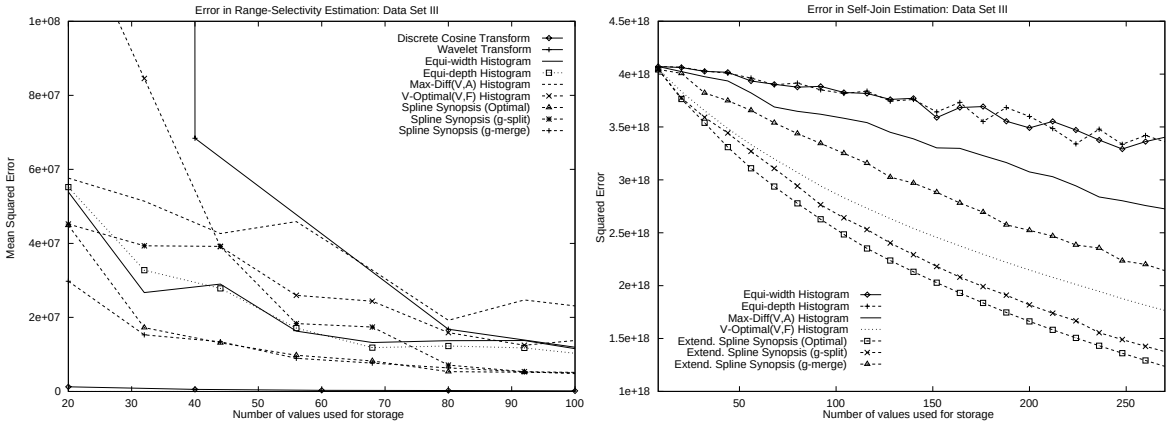

Figure 8: Accuracy of different techniques on "worst-case" data

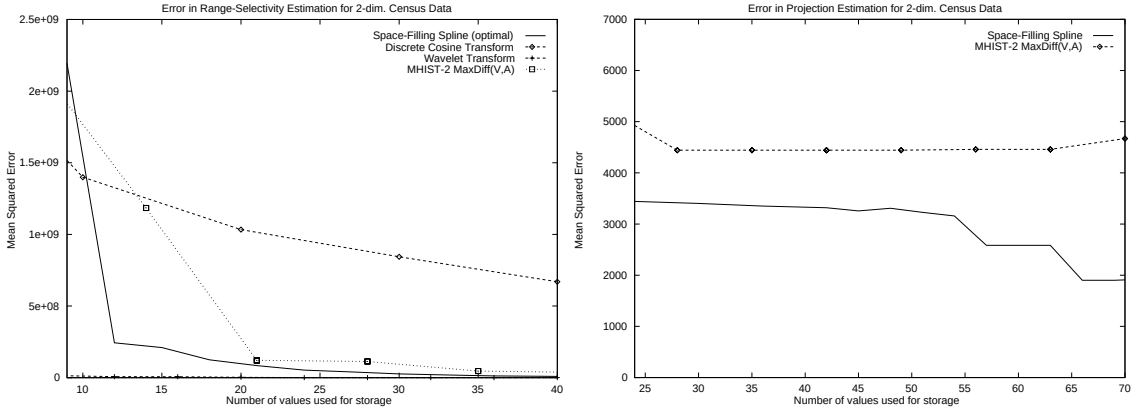

Figure 9: Accuracy of different techniques for multidimensional CENSUS data

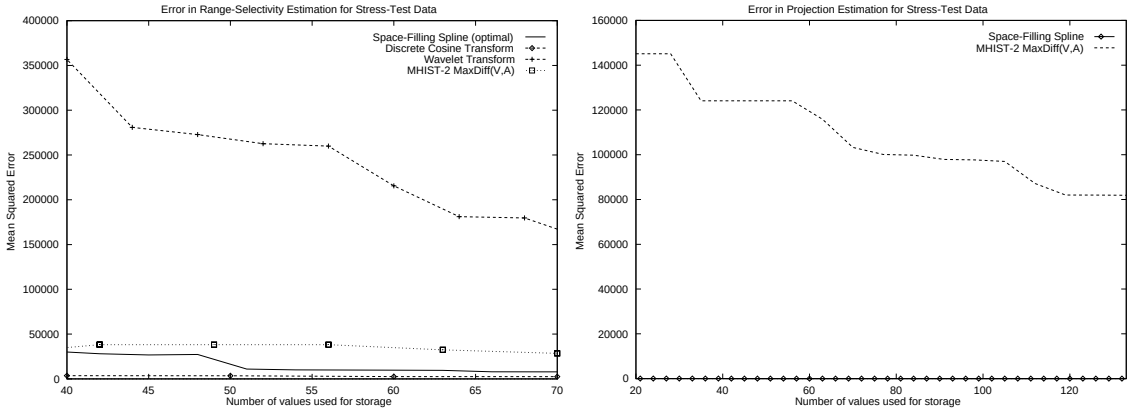

Figure 10: Accuracy of different techniques for multidimensional synthetic data

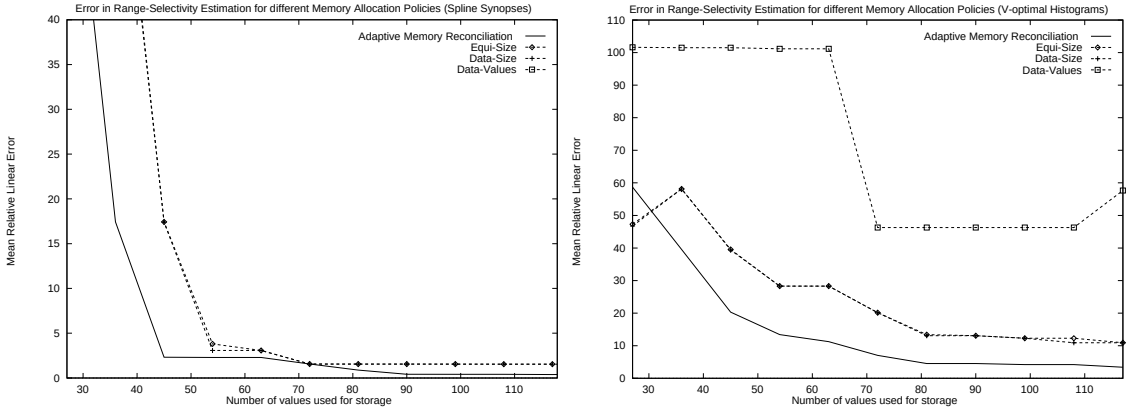

Figure 11: The relative error for different memory distribution techniques

[6] D. Barbará, W. DuMochel, C. Faloutsos, P. J. Haas, J. M. Hellerstein, Y. Ioannidis, H. Jagadish, T. Johnson, R. Ng, V. Poosala, K. A. Ross, and K. C. Sevcik. The New Jersey Data Reduction Report. *IEEE D.E. Bulletin*, 1997.

[7] B. Blohsfeld, D. Korus, and B. Seeger. A Comparison of Selectivity Estimators for Range Queries on Metric Attributes. In *Proceedings of the ACM SIGMOD Conference*, pages 239–250, 1999.

[8] S. Chaudhuri, R. Motwani, and V. R. Narasayya. On Random Sampling over Joins. In *Proceedings of the ACM SIGMOD Conference*, pages 263–274, 1999.

[9] S. Chaudhuri and V. R. Narasayya. Automating Statistics management for Query Optimizers. In *Proceedings of the IEEE Conference on Data Engineering*, pages 339–348, 2000.

[10] C. M. Chen and N. Roussoploulos. Adaptive Selectivity Estimation Using Query Feedback. In *Proceedings of the ACM SIGMOD Conference*, pages 161–172, May 1994.

[11] C. de Boor. *A practical guide to splines*. Springer-Verlag, 1978.

[12] P. B. Gibbons and Y. Matias. New Sampling-Based Summary Statistics for Improving Approximate Query Answers. In *Proceedings of the ACM SIGMOD Conference*, 1998.

[13] P. B. Gibbons and Y. Matias. Synopsis Data Structures for Massive Data Sets. In *Symposium on Discrete Algorithms*, 1999.

[14] P. B. Gibbons, Y. Matias, and V. Poosala. Fast Incremental Maintenance of Approximate Histograms. In *Proceedings of the 23rd International Conference on Very Large Databases*, 1997.

[15] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi. Approximating Multi-Dimensional Aggregate Range Queries Over Real Attributes. In *Proceedings of the ACM SIGMOD Conference*, 2000.

[16] P. J. Haas. Selectivity and Cost Estimation for Joins Based on Random Sampling. *Journal of Computer and System Sciences*, pages 550–569, 1996.

[17] H. V. Jagadish, N. Koudas, S. Mutukrishnan, V. Poosala, K. Sevcik, and T. Suel. Optimal Histograms with Quality Guarantees. In *Proceedings 24th International Conference on Very Large Databases*, pages 275–286, 1998.

[18] I. Kamel and C. Faloutsos. Hilbert R-Tree: An Improved R-Tree using Fractals. In *Proceedings of the 20th International Conference on Very Large Databases*, pages 500–509, 1994.

[19] A. König and G. Weikum. Combining Histograms and Parametric Curve Fitting for Feedback-Driven Query Result-size Estimation. In *25th International Conference on Very Large Databases*, 1999.

[20] J.-H. Lee, D.-H. Kim, and C.-W. Chung. Multi-dimensional Selectivity Estimation Using Compressed Histogram Information. In *Proceedings of the ACM SIGMOD Conference*, pages 205–214, 1999.

[21] Y. Matias, J. S. Vitter, and M. Wang. Wavelet-Based Histograms for Selectivity Estimation. In *Proceedings of the ACM SIGMOD Conference*, pages 448–459. ACM Press, 1998.

[22] B. Moon, H. Jagadish, C. Faloutsos, and J. H. Saltz. Analysis of the Clustering Properties of the Hilbert Space-Filling Curve. *To appear in IEEE Transactions on Knowledge and Data Engineering*.

[23] S. Muthukrishnan, V. Poosala, and T. Suel. On Rectangular Partitionings in Two Dimensions. In *Proceedings of International Conference on Database Theory*, pages 236–256, 1999.

[24] G. Piatetsky-Shapiro and C. Connel. Accurate Estimation of the Number of Tuples Satisfying a Condition. In *Proceedings of the ACM SIGMOD Conference, Boston, Massachusetts*, pages 256–276, 1984.

[25] L. K. Platzman and J. J. Bartholdi, III. Spacefilling Curves and the Planar Travelling Salesman Problem. *Journal of the ACM*, pages 719–737, Oct 1989.

[26] V. Pooosala and Y. E. Ioannidis. Selectivity Estimation Without the Attribute Value Independence Assumption. In *Proceedings of the ACM SIGMOD Conference*, Athens, Greece, 1997.

[27] V. Poosala. *Histogram-based Estimation Techniques in Database Systems*. PhD thesis, University of Wisconsin-Madison, 1997.

[28] V. Poosala, Y. Ioannidis, P. Haas, and E. Shekita. Improved Histograms for Selectivity Estimation or Range Predicates. In *Proceedings of the ACM SIGMOD Conference*, pages 294–305. ACM Press, 1996.

[29] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Receipes in C*. Cambridge University Press, 1996.

[30] H. Sagan. *Space-Filling Curves*. Springer-Verlag, 1994.

[31] W. Sierpiński. Sur une nouvelle courbe contine qui remplit toute une air plane. *Bull. Acad. Sci. de Cracovie (Sci. math. en nat., Serie A)*, pages 462–478, 1912.

[32] W. Sun, Y. Ling, N. Rishe, and Y. Deng. An instant and accurate Size Estimation Method for Joins and Selections in an Retrival-Intensive Environment. In *Proceedings of the ACM SIGMOD Conference*, pages 79–88, 1993.