The Efficiency of Modern-day Histogram-like Techniques for Query Optimization

B. John Oommen[*] **and Luis G. Rueda**[†]

Abstract

One of the most difficult tasks in modern day Database Management Systems (DBMS) is information retrieval. Basically, this task involves a user query, written in a high-level language such as the Structured Query Language (SQL), and some internal operations, which are transparent to the user. The internal operations are carried out through very complex modules that decompose, optimize and execute the different operations. We consider the problem of Query Optimization which consists of the system choosing, among many different Query Evaluation Plans (QEP), the most economical one. Since the number of QEPs increases exponentially as the number of relations involving the query is larger, query optimization is a very complex problem.

Many estimation techniques have been developed in order to approximate the cost of a QEP. Histogram-based techniques are the most used methods in this context. In this paper, we discuss the efficiency of some of these methods: *Equi-width*, *Equi-depth*, the *Rectangular Attribute Cardinality Map (R-ACM)*, and the *Trapezoidal Attribute Cardinality Map (T-ACM)*. These methods are used to estimate the cost of the different QEPs and choose the best one. It has been shown that the errors of the estimates from the R-ACM and T-ACM are significantly less than the corresponding errors obtained from the Equi-width and Equi-depth. This fact has been formally demonstrated using reasonable statistical distributions for the cost of a QEP, the doubly exponential distribution and the normal distribution. For the empirical analysis, we have developed a formal, rigorous prototype model used to analyze these methods on random databases. Our empirical results demonstrate that the R-ACM chooses a superior QEP more than three times as often as the Equi-width and Equi-depth. In the case of the T-ACM the ratio is even greater than four. Indeed, in the most general scenario, we analytically prove that under certain models, the better the accuracy of an estimation technique, the greater the probability of choosing the most efficient QEP.

[*]Senior Member, IEEE. School of Computer Science, Carleton University, 1125 Colonel By Dr., Ottawa, ON, K1S 5B6, Canada. E-mail: `oommen@scs.carleton.ca`. Partially supported by NSERC, the Natural Science and Engineering Research Council of Canada.

[†]School of Computer Science, Carleton University, 1125 Colonel By Dr., Ottawa, ON, K1S 5B6, Canada. E-mail: `lrueda@scs.carleton.ca`. Partially supported by Departamento de Informática, Universidad Nacional de San Juan, Argentina.

1 Introduction

1.1 Overview

Query optimization in relational and object oriented database systems has been studied for many years. When an end user performs a query, many internal operations need to be done to retrieve the information requested. The most important operation between tables is the natural join on a particular attribute. In real databases, a query may consist of joining several tables. When more than two tables have to be joined, intermediate join operations are performed to ultimately obtain the final relation. As a result, the same query can be performed by means of different intermediate (join) operations. A simple sequence of join operations that leads to the same final result is called a *Query Evaluation Plan* (QEP). Each QEP has associated an internal cost, which depends on the number of operations performed in the intermediate joins. The problem of choosing the best QEP is a combinatorially explosive optimization problem. This problem is currently solved by estimating the query result sizes of the intermediate relations and selecting the most efficient access QEP.

Since the analysis of selecting the best QEP must be done in "real" time, it is not possible to inspect the real data in this phase. Consequently, query result sizes are usually estimated using statistical information about the structures and the data maintained in the database catalogue. This information is used to approximate the distribution of the attribute values in a particular relation. Hence the problem of selecting the best QEP depends on how well that distribution is approximated.

In [14], it has been shown that errors in query result size estimates may increase exponentially with the number of joins. Since current databases and the associated queries increase in complexity, numerous efforts have being made to devise more efficient techniques that solve the query optimization problem.

Several techniques have been proposed to estimate query result sizes, including histograms, sampling, and parametric techniques [2, 3, 9, 17, 19, 21, 27, 32]. Among these techniques, histograms are the most commonly used form of statistical information. They (and in particular, the Equi-width and the Equi-depth explained below) are incorporated in most of the commercial database systems, including Oracle, Microsoft SQL Server, Teradata, DB2, etc.

In this paper, we focus on methods that use non-parametric statistics, i.e. histogram-based techniques. First of all, we have developed a formal model for random databases with random attributes and random queries. Using this rigorous model, we have developed a prototype that tests the four prominent models of histograms known in the literature: *Equi-width* [1, 17], *Equi-depth* [21, 27], the *Rectangular Attribute Cardinality Map (R-ACM)* [24, 25, 33], and the *Trapezoidal Attribute Cardinality Map (T-ACM)* [23, 26, 33]. For reasons explained presently, although the current version of the comprehensive package does not include the implementation of the V-optimal scheme (and its enhancements) [14, 15, 28, 29, 30], and the wavelet-based histograms [16, 20], we plan to incorporate the latter strategy too in the next version. Our empirical results demonstrate that the R-ACM chooses a superior QEP more than three times as often as the Equi-width and Equi-depth. In the case of the T-ACM the ratio is even greater than four. Indeed, in the most general scenario, we analytically prove that under certain models, the better the accuracy of an estimation technique, the greater the probability of choosing the most efficient QEP.

1.2 Importance of the Result

In the 1999 IDEAS Conference in Montreal, Canada, the database authority, Prof. J. D. Ullman from Stanford proposed the following question. He queried: "Does a system using a superior histogram method necessarily yield a superior QEP?". He also alluded to the experimental results using the Equi-width [1, 17] and Equi-depth histograms [21, 27] which *seemed to imply* that the answer to the query was *negative.*

The importance of the results of this paper is that we show that the answer to his question is "stochastically positive". In other words, we prove that although a superior histogram method may not always yield a superior QEP, the probability that the superior histogram method yields a superior QEP exceeds the problem that it yields an inferior QEP. This thus justifies and gives a formal rigorous basis for the fact that all current day database systems use histogram methods to determine the QEP.

We also show that if two "almost comparable" histogram methods (like the Equi-width and the Equi-depth) are compared, the probability of the superior one yielding a superior QEP is negligible. This probably answers for Prof. Ullman's implied position. However, if the error of one method is *significantly* less than the error of the second, the probability of obtaining a superior QEP is also significantly greater. This is because of the explicit form of the function involved, and further answers for the experimental results alluded to by Prof. Ullman. The corresponding result for significantly superior histograms also follows from the functional form, and is verified by experimental results involving the Equi-width, the Equi-depth, and the Rectangular Cardinality Attribute Cardinality Map (R-ACM), a recently devised histogram method [24].

2 Query Optimization

The query processing problem is generally divided in three different stages: *query decomposition*, *query optimization*, and *query execution.* In query decomposition, the query, written in SQL, is parsed and validated. This module produces a query in relational algebra as a query *Operator Tree.* The next module, query optimization, takes this operator tree and searches for a plan that establishes the order in which the different algebraic operations are done. For queries involving more than two relations, there are more than one way (or QEP) in which the query can be evaluated and processed.

As discussed above, a QEP defines the order in which the operations are executed. This order is quite important in query optimization, since different QEPs have different costs. The cost is given by the response time and resource consumption in which the system processes the query. In general, the most important aspect to consider is the number of I/O operations, which is the dominant factor in evaluating the cost [33]. Therefore, it is commonly assumed in both the academic literature and in real life systems that the cost of a query is proportional to the total number of all tuples generated in the intermediate relations used to obtain the final answer.

Considering the natural join between n relations, the number of alternatives of executing the query (or different QEPs) grows exponentially with n. The total number of different join orders is given by $\frac{(2(n-1))!}{(n-1)!}$. In real databases the number of relations in a query is normally greater than 10, for which the number of different join orders is 176 billion! The problem of finding the optimal join order has been proven to be NP-hard [12, 4, 31, 18].

Since it is almost impossible to evaluate all the join orders so as to decide on the best one, many techniques have been developed in order to solve this problem. The most important strategy in this area is the evaluation of a query by estimating the number of operations required by a join. These techniques resort to the information stored in the DBMS Catalogue. Whether the approximation of the real cost by estimation is good or not depends on how accurate the estimation process is. The most common techniques proposed and used today are Histogram-based techniques explained below.

3 Histogram Based Techniques

Histogram based techniques are the most used methods among the non-parametric estimates. These techniques approximate the distribution of an attribute value by a pre-computed histogram. Indeed, these are the common schemes used in commercial database systems. Basically, the attribute value set is divided in *intervals* or *buckets*, each of them with an arbitrary number of values. Typically, only the interval bounds and the frequency counters[1] are stored. These data values are later used for approximating the individual counter for each value. The approximation is achieved by interpolation.

3.1 The Equi-Width Histogram

The concept of the Equi-width histogram was used in statistical literature and in statistical pattern recognition for many decades. However, its use in databases was predominantly advocated by Piatetsky-Shapiro and Connel [27]. The basic idea of this technique is to divide the set of attribute values in intervals or buckets that have, *as approximately as possible*, the same width. The value of each attribute value in the bucket is approximated by the height of the bucket. The height is calculated as the summation of the frequencies divided by the number of values in the bucket. The frequency for each attribute value is the number of tuples in the relation that have this attribute value.

3.2 The Equi-Depth Histogram

The structure of the Equi-depth histogram has also been extensively studied both by statisticians and by researches in pattern recognition [7, 10]. However, its use in databases was also primarily motivated by the

[1]The frecuency counter is given by the number of occurrences of all the values in the bucket.

work of Piatetsky-Shapiro and Connel [27]. Basically, this technique consists of dividing the set of attribute values in buckets that have *as approximately as possible* the same population[2]. The frequency of a value is calculated (approximated) by dividing the population in the bucket by the number of attribute values that belong to the bucket.

An easy way of constructing a histogram using this technique, consists of the following. Starting from the leftmost value in the direction of the rightmost value, accumulate the frequency counters until the population of the current bucket is greater than or equal to the whole population divided by the number of buckets. After achieving this, generate a new bucket starting with the remaining frequency of the previous bucket and proceeding along the rest of the attribute values.

3.3 The Rectangular Attribute Cardinality Map

The Rectangular Attribute Cardinality Map (R-ACM) histogram of a given attribute, in its simplest form, is a one-dimensional integer array that stores the count of the tuples of a relation corresponding to that attribute, and for some subdivisions for the range of values assumed by that attribute [33, 24]. The R-ACM is, in fact, a modified form of the histogram. But unlike the two major forms of histograms, namely, the Equi-width histogram (in which all the sector widths are equal) and the Equi-depth histogram (in which where the number of tuples in each histogram bucket is equal) the R-ACM has a variable sector width, and has varying number of tuples in each sector. The sector widths or subdivisions of the R-ACM are generated according to a rule that aims at minimizing the estimation error within each subdivision.

The R-ACM can be either one-dimensional or multi-dimensional depending on the number of attributes being mapped. In this paper, we shall only deal with the one-dimensional case of ordinal numeric values. As in the case of traditional histograms, non-numeric attributes are dealt with by using mapping functions.

A formal definition of the R-ACM and its properties can be found in [33, 25]. Also found in [33, 25] are various expressions for the join estimation error, average-case and worst-case errors for both equality-select and range-select operations using the R-ACM. Experimentally, the R-ACM has been found to approximate the actual histograms with much smaller errors than the Equi-width and the Equi-depth. These significantly superior results obtained from running the different methods in simulated databases and real life databases like the NBA, the US Census, and benchmark systems like the TPC-D, can be found in [33, 25].

The mathematical foundation for the R-ACM is the previous work due to Oommen and his student Nguyen, in which they proposed a general strategy for obtaining *Moment-Preserving Piecewise Linear Approximations* of functions [22]. This technique approximates a function by preserving a finite number of geometric moments which are related to the transform domain parameters of a function. If this method is applied directly to yield a time/spatial domain approximation it ensures that the approximated function is close to the actual function in both the time (space) domain and the transform domain in a well defined way. Indeed, the R-ACM can be viewed as a specific perspective of this method since it preserves (or minimizes

[2]The population is the number of tuples in which the attribute has any of the values in the range.

the variation from) the *first* geometric moment of the actual histogram. It thus ensures that the actual histograms are well approximated in both the time and transform domains.

3.4 The Trapezoidal Attribute Cardinality Map

The Trapezoidal Attribute Cardinality Map (T-ACM) histogram of a given attribute is a one-dimensional integer array that stores the count of the tuples of a relation corresponding to that attribute, and for some equal subdivisions for the range of values assumed by that attribute. The T-ACM is, in fact, a modified form of the histogram. But unlike the Rectangular Attribute Cardinality Map (R-ACM) discussed above and the two major forms of histograms, namely Equi-width histogram [17] and the Equi-depth histogram [27], where the histogram buckets are rectangular cells, the sectors in the T-ACM are trapezoidal cells. The Trapezoidal Attribute Cardinality Map (T-ACM) is a non-parametric histogram-like estimation technique based on the trapezoidal-rule of numerical integration, which generalizes the R-ACM from a "step" representation to a "linear" representation. In this paper we consider the one-dimensional T-ACM whose formal definition and properties can be found in [33, 26]. As in the R-ACM, the T-ACM has also been found to achieve much smaller errors than the traditional histogram methods (the Equi-width and the Equi-depth) on well-known standards such as the NBA, the US Census, and TPC-D benchmarks [33, 26].

3.5 Our Contribution

In this paper, we analytically show that a system using a superior histogram method yields a superior QEP. We provide a "stochastically positive" answer to the question regarding the relation between the efficiency and optimality of histogram methods. In other words, we prove that although a superior histogram method may not always yield a superior QEP, the probability that the superior histogram method yields a superior QEP exceeds the problem that it yields an inferior QEP. We have analytically shown this fact by using doubly exponentially distributed random variables. For the case of normally distributed random variables, we have used a numerical analysis, due to the impossibility of integrating the normal distribution density function. The empirical results obtained from the experiments that we have conducted on randomly generated databases corroborate the superiority of the R-ACM and the T-ACM over the traditional histogram methods, the Equi-width and the Equi-depth.

3.6 Comparative Open Work

Apart from the references mentioned above, a few significant references (which also objectively survey the field) and which present relatively new contributions deserve prominent mention. The first in this list is the paper due to Poosala *et. al.* [29], which specifically addresses the issue of designing histograms suitable for range predicates. In particular, the histogram design using the equi-sum, V-optimal, and spline-based methods are discussed. Of these, V-optimal-based histogram methods are of particular interest, because they group contiguous sets of frequencies into buckets so as to minimize the variance of the overall frequency approximation. Philosophically speaking, this resembles the principle of the R-ACM described presently, which attempts to minimize the variation around the mean in any given bucket, and consequently has the property of minimizing the variance inside the bucket. While, in the R-ACM, this is done along the attribute values, all the tuples within a bucket have almost identical *frequency* values. Thus, the R-ACM can be perceived to be analogous to clustering the attribute values by their frequencies, and yet maintaining the sequential nature of the attribute values. The similarities and differences between the actual histograms generated by these two families of methods are still being investigated. The paper [29] catalogues different families of V-optimal histograms based on the sort parameters and the source parameters, and contains a complete taxonomy of all resulting histograms.

Poosala and Ionnadis continued this work in 1997 [28], where the assumption of independence of the individual attributes within a relation was relaxed. While this work was directed towards certain specific histograms (and in particular, some of the histograms mentioned above), we believe that the exact same techniques can be used to develop multidimensional ACMs. Indeed, we believe that the Hilbert numbering concept and the single value decomposition specified in [28] can be applied to the new histograms that we have introduced, for higher dimensions. This is also currently being investigated.

Jagadish *et. al.* have studied the problem of yielding optimal histograms which guarantee certain quality specifications [15]. These specifications can be either space-bounded or error-bounded. A formal scheme to design space-bounded V-optimal histograms (and a corresponding approximation algorithm for the same) have also been presented in [15]. Identical developments for the error bounded V-optimal histograms have also been included. As a matter of comparison with our work, we believe that these identical philosophies can be used to obtain space bounded and error bounded histograms for the R-ACM and the T-ACM. Because the R-ACM aims to minimize the variation around the first moment of the moment-based approximation, it is our conjecture that it is, in principle, an error bounded quality approximation. The space bounded approximation for the R-ACM is *probably* the approximation obtained by controlling and optimizing the value of "τ". All these issues lead to numerous open problems.

A completely new strategy for approximating density functions in database query optimization involves wavelet-based approximations. Such approximations have been used in signal processing, image processing, and, in general, in function approximation for almost two decades [16, 11, 20]. The application of these techniques to databases naturally leads to the approximation of the density function using the m-most significant wavelet coefficients. Informally speaking, due to the results in [22], we believe that these wavelet-based histograms are closely related to the R-ACM and the T-ACM because, the latter also optimize the linear approximations by maintaining the most significant moment coefficients, as opposed to wavelet coefficients. The issue of comparing our ACM technologies with such approximations is still open, and warrants investigation.

From the above discussion, it is clear that a lot of work has yet to be done to compare the ACMs with other histogram methods reported in the academic and commercial literature. A benchmarking exercise which compares the various schemes in terms of accuracy and query-evaluation plan efficiency, has to be undertaken before a conclusive ranking of the schemes can be specific. Such a tedious study for the TPC-family of benchmarks is currently underway. However, we believe the results presented here are extremely valuable, because they demonstrate how the ACMs compare with the two histogram methods that are used in *all the commercial database products.*

4 Efficiency vs. Accuracy of Various Histogram Methods

It has been shown that the R-ACM and the T-ACM are more accurate than the Equi-width and the Equi-depth histograms, with regard to the approximation to the real underlying attribute-value histograms. In this section, we analyze the efficiency of these methods from another point of view. Consider the equi-join operation of several relations. We shall now quantify the quality of a method by measuring the number of times this method achieves the right decision, i.e. it chooses the optimal QEP.

For the analysis done below, we work with two models for the error function: the doubly exponential distribution and the normal distribution. In the former, the probability of obtaining a value that deviates from the mean (or true value) falls exponentially as a function of the deviation. The exponential distribution is more typical in reliability analysis and in failure models, and in this particular domain, the question is one of evaluating how reliable the quality of a solution is if only an estimate of its performance is available. More importantly, it is used as an approximation to the Gaussian distribution for reasons which will be clarified momentarily. The Gaussian model is much more difficult to analyze, since there is no closed-form algebraic expression for integrating the probability density function. However, a formal computational proof is included, which confirms our hypothesis.

4.1 Analysis Using Exponential Distributions

A random variable, X, is said to be *d-exp*[3] distributed with parameter λ if the density function is given by:

$$f_X(x) = \frac{1}{2}\lambda e^{-\lambda|x-M|} \qquad -\infty < x < \infty \ , \tag{1}$$

where $E(X) = M$.

If X is a d-exp random variable, it can be shown that:

$$\text{Var}[X] = \frac{2}{\lambda^2} \ . \tag{2}$$

[3] *d-exp* is a short notation for "doubly exponential".

Although we only consider the analysis for any two histogram schemes, referred to by M_1 and M_2 (which could be the R-ACM and the Equi-width schemes), the result we claim can be extended to the T-ACM and the Equi-depth as well.

Consider two query-size estimation methods, M_1 and M_2. The probability of choosing a cost value of a particular QEP by M_1 and that of choosing a cost value by M_2 are represented by two independent random variables. Clearly, this assumption of independence is valid because there is no reason why the value obtained by one estimation strategy should affect the value obtained by the second.

Without loss of generality, if the mean cost of the optimal QEP is μ, by shifting the origin by μ, we can work with the assumption that the cost of the best QEP is 0, which is the mean of these two random variables. The cost of the second best QEP is given by another two random variables (one for M_1 and the other one for M_2) whose mean, $c > 0$, is the same for both variables. An example will help to clarify this.

Example 1. Suppose that M_1 chooses the optimal cost value with probability represented by the random variable $X_1^{(opt)}$ whose mean is 0 and $\lambda_1 = 0.4$. This method also chooses another sub-optimal cost value according to $X_1^{(subopt)}$ whose mean is 8 and $\lambda_1 = 0.4$.

M_2 is another method that chooses the optimal cost value with probability given by $X_2^{(opt)}$ whose parameters are $M = 0$ and $\lambda_2 = 0.2$. Another sub-optimal cost value is chose with probability given by $X_2^{(subopt)}$ whose parameters are $M = 8$ and $\lambda_2 = 0.2$.

Since $\frac{2}{\lambda_1^2} < \frac{2}{\lambda_2^2}$, with some insight, we can see that the probability that M_1 chooses a sub-optimal cost value is smaller than that of M_2 choosing the sub-optimal cost value. This scenario is depicted in Figure 1. $\square$

The result described above is formalized in the following theorem.

Theorem 1. Suppose that:

- M_1 and M_2 are two query result size estimation methods.
- X_1 and X_2 are two doubly exponential random variables that represent the cost values of the *optimal* QEP obtained by M_1 and M_2 respectively.
- X_1' and X_2' are another two random variables representing the cost value of a *non-optimal* QEP obtained by M_1 and M_2 respectively.
- $0 = E[X_1] = E[X_2] \leq E[X_1'] = E[X_2'] = c$.

Let p_1 and p_2 be the probabilities that M_1 and M_2 respectively make the wrong decision. Then,

$$\text{if } \mathrm{Var}[X_1] = \mathrm{Var}[X_1'] = \frac{2}{\lambda_1^2} \leq \frac{2}{\lambda_2^2} = \mathrm{Var}[X_2] = \mathrm{Var}[X_2'], \quad p_1 \leq p_2 \ .$$

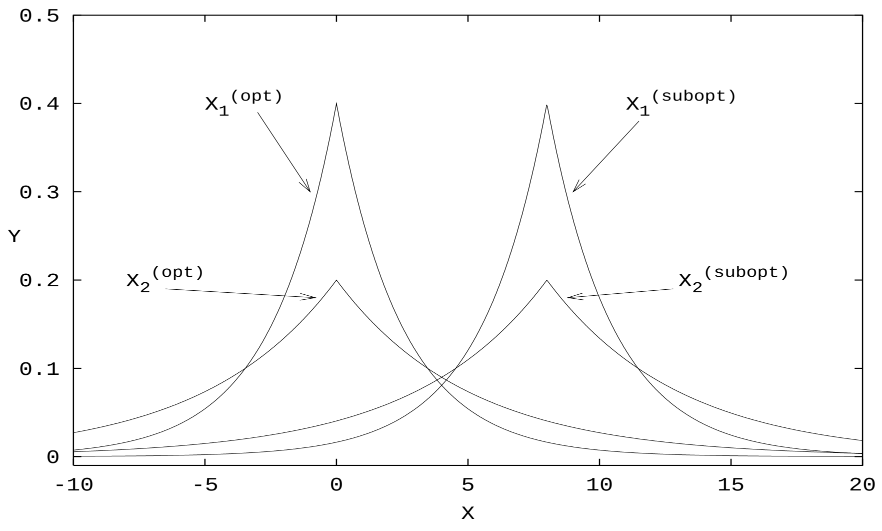


Figure 1: An example of d-exp distributions for the random variables $X_1^{(opt)}$, $X_2^{(opt)}$, $X_1^{(subopt)}$ and $X_2^{(subopt)}$, whose parameters are $\lambda_1 = 0.4$ and $\lambda_2 = 0.2$.

Proof. Consider a particular value x. The probability that the value x leads to a wrong decision made by M_1, is given by:

$$\begin{aligned} I_{11} &= \int_{-\infty}^{x} \tfrac{1}{2}\lambda_1 e^{\lambda_1(u-c)}\,du & \text{if } x < c \text{ , and} \\ I_{12} &= \int_{-\infty}^{c} \tfrac{1}{2}\lambda_1 e^{\lambda_1(u-c)}\,du + \int_{c}^{x} \tfrac{1}{2}\lambda_1 e^{-\lambda_1(u-c)}\,du & \text{if } x > c \text{ .} \end{aligned} \tag{3}$$

Solving the integrals, (3) results in:

$$\begin{aligned} I_{11} &= \tfrac{1}{2}e^{\lambda_1(x-c)} - \lim_{u\to-\infty} \tfrac{1}{2}e^{-\lambda_1(u-c)} &= \tfrac{1}{2}e^{-\lambda_1(x-c)} \text{ , and} \\ I_{12} &= \lim_{u\to-\infty} \tfrac{1}{2}e^{-\lambda_1(-u+c)} + \tfrac{1}{2} - \tfrac{1}{2}e^{-\lambda_1(x-c)} + \tfrac{1}{2} &= 1 - \tfrac{1}{2}e^{-\lambda_1(x-c)} \text{ .} \end{aligned} \tag{4}$$

The probability that M_1 makes the wrong decision for *all* the values of x is the following function of λ_1 and c:

$$p_1 = I(\lambda_1, c) = \int_{-\infty}^{0} I_{11}\frac{1}{2}\lambda_1 e^{\lambda_1 x}\,dx + \int_{0}^{c} I_{11}\frac{1}{2}\lambda_1 e^{-\lambda_1 x}\,dx + \int_{c}^{\infty} I_{12}\frac{1}{2}\lambda_1 e^{-\lambda_1 x}\,dx \text{ .} \tag{5}$$

which, after applying the distributive law and substituting the values of I_{11} and I_{12}, can be written as:

$$\int_{-\infty}^{0} \frac{\lambda_1}{4} e^{2\lambda_1 x - \lambda_1 c}\, dx - \int_{0}^{c} \frac{\lambda_1}{4} e^{-\lambda_1 c}\, dx + \int_{c}^{\infty} \frac{\lambda_1}{2} e^{-\lambda_1 x} - \frac{\lambda_1}{4} e^{-2\lambda_1 x + \lambda_1 c}\, dx \ . \tag{6}$$

After solving the integrals, (6) is transformed into:

$$\frac{1}{8} e^{-\lambda_1 c} + \frac{1}{4} \lambda_1 c e^{-\lambda_1 c} + \frac{3}{8} e^{-\lambda_1 c} = \frac{1}{2} e^{-\lambda_1 c} + \frac{1}{4} \lambda_1 c e^{-\lambda_1 c} \ . \tag{7}$$

Similarly, we do the same analysis for p_2, which is a function of λ_2 and c:

$$p_2 = I(\lambda_2, c) = \frac{1}{2} e^{-\lambda_2 c} + \frac{1}{4} \lambda_2 c e^{-\lambda_2 c} \ . \tag{8}$$

We have to prove that:

$$p_1 = \frac{1}{2} e^{-\lambda_1 c} + \frac{1}{4} \lambda_1 c e^{-\lambda_1 c} \le \frac{1}{2} e^{-\lambda_2 c} + \frac{1}{4} \lambda_2 c e^{-\lambda_2 c} = p_2 \ . \tag{9}$$

Multiplying both sides by 2, and substituting $\lambda_1 c$ for α_1 and $\lambda_2 c$ for α_2, (9) can be written as follows:

$$e^{-\alpha_1} + \frac{1}{2} \alpha_1 e^{-\alpha_1} \le e^{-\alpha_2} + \frac{1}{2} \alpha_2 e^{-\alpha_2} \ . \tag{10}$$

Substituting α_2 for $k\alpha_1$, $\alpha_1 \ge 0$ and $0 < k \le 1$, (10) results in:

$$q_1 = e^{-\alpha_1} + \frac{1}{2} \alpha_1 e^{-\alpha_1} \le e^{-k\alpha_1} + \frac{1}{2} k\alpha_1 e^{-k\alpha_1} = q_2 \, . \tag{11}$$

We now prove that $q_1 - q_2 \le 0$. After applying natural logarithm to both sides of (11) and some algebraic manipulation, $q_1 - q_2 \le 0$ implies:

$$F(\alpha_1, k) = k\alpha_1 - \alpha_1 + \ln(1 + \frac{1}{2}\alpha_1) - \ln(1 + \frac{1}{2} k\alpha_1) \le 0 \, . \tag{12}$$

To prove that $F(\alpha_1, k) \le 0$, we use the fact that $\ln x \le x - 1$. Hence, we have:

$$\begin{aligned} F(\alpha_1, k) &= \alpha_1(k-1) + \ln\left(\frac{1+\frac{1}{2}\alpha_1}{1+\frac{1}{2}k\alpha_1}\right) && (13)\\ &\leq \alpha_1(k-1) + \frac{1+\frac{1}{2}\alpha_1}{1+\frac{1}{2}k\alpha_1} - 1 && (14)\\ &= \alpha_1(k-1) + \frac{\alpha_1 - k\alpha_1}{2+k\alpha_1} && (15)\\ &= \frac{k\alpha_1 + k^2\alpha_1^2 - \alpha_1 - k\alpha_1^2}{2+k\alpha_1} && (16)\\ &= \frac{\alpha_1(k-1)(k\alpha_1+1)}{2+k\alpha_1} \leq 0\,, && (17) \end{aligned}$$

because:

(i) $0 < k \leq 1$ and $\alpha_1 \geq 0 \;\Rightarrow\; \alpha_1(k-1) \leq 0$ and $k\alpha_1 + 1 > 0$. Hence $\alpha_1(k-1)(k\alpha_1+1) \leq 0$, and

(ii) $0 < k \leq 1$ and $\alpha_1 \geq 0 \;\Rightarrow\; 0 < k\alpha_1 \leq \alpha_1 \;\Rightarrow\; k\alpha_1 + 2 > 2 > 0$.

Hence the theorem. □

The above theorem can be viewed as a "sufficiency result". In other words, we have shown that $q_1 - q_2 \leq 0$ or that $p_1 \leq p_2$. We now show a "necessity result" stated as a uniqueness result. This result states that the function $p_1 \leq p_2$ has its equality ONLY at the boundary condition where the two distributions are exactly identical.

To prove the necessity result, we consider $q_2 - q_1$ which, derived from (11), can be written, as a function of α_1 and k, as:

$$G(\alpha_1, k) = e^{-k\alpha_1} + \frac{1}{2}k\alpha_1 e^{-k\alpha_1} - e^{-\alpha_1} - \frac{1}{2}\alpha_1 e^{-\alpha_1}\,. \tag{18}$$

By examining its partial derivatives we shall show that there are two solutions for equality. Furthermore, when $\alpha_1 \geq 0$ and $0 < k \leq 1$, we shall see that for a given k, there is only one solution, namely $\alpha_1 = 0$ and k, $0 < k \leq 1$, proving the uniqueness.

Theorem 2. Suppose that $\alpha_1 \geq 0$, $0 < k \leq 1$. Let $G(\alpha_1, k)$ be:

$$G(\alpha_1, k) = e^{-k\alpha_1} + \frac{1}{2}k\alpha_1 e^{-k\alpha_1} - e^{-\alpha_1} - \frac{1}{2}\alpha_1 e^{-\alpha_1}\,. \tag{19}$$

Then $G(\alpha_1, k) \geq 0$, and there are exactly two solutions for $G(\alpha_1, k) = 0$, being: $\{\alpha_1 = -1, k = 1\}$ and $\{\alpha_1 = 0, k\}$.

Proof. We must prove that, as defined in the theorem statement, $G(\alpha_1, k) \geq 0$.

We shall prove that this is satisfied by determining the local minima for $G(.,.)$, where $\alpha_1 \geq 0$ and $0 < k \leq 1$. We first find the partial derivatives of (19) with respect to α_1 and k:

$$\frac{\partial G}{\partial \alpha_1} = -\tfrac{1}{2}ke^{-k\alpha_1} - \tfrac{1}{2}k^2\alpha_1 e^{-k\alpha_1} + \tfrac{1}{2}e^{-\alpha_1} + \tfrac{1}{2}\alpha_1 e^{-\alpha_1} = 0\,, \text{ and} \tag{20}$$

$$\frac{\partial G}{\partial k} = -\tfrac{1}{2}\alpha_1 e^{-k\alpha_1} - \tfrac{1}{2}k\alpha_1^2 e^{-k\alpha_1} = 0\,. \tag{21}$$

We now solve (20) and (21) for α_1 and k. Equation (21) can be written as follows:

$$-\frac{1}{2}\alpha_1 e^{-k\alpha_1} = \frac{1}{2}k\alpha_1^2 e^{-k\alpha_1}\,, \tag{22}$$

which, after canceling some terms results in $k\alpha_1^2 + \alpha_1 = 0$. Solving this equation for α_1, we have: $\alpha_1 = -\frac{1}{k}$ and $\alpha_1 = 0$. Substituting $\alpha_1 = -\frac{1}{k}$ in (20) and canceling some terms, we obtain:

$$\frac{1}{2}e^{-\alpha_1} + \frac{1}{2}\alpha_1 e^{-\alpha_1} = 0\,, \tag{23}$$

which results in the solution to be $\alpha_1 = -1$, and consequently, $k = 1$.

The second root, $\alpha_1 = 0$, indicates that the minimum is achieved for any value of k.

We have thus found two solutions for (20) and (21), $\{\alpha_1 = 0, k\}$ and $\{\alpha_1 = -1, k = 1\}$. Since $\alpha_1 \geq 0$, it means that α_1 can have at least a value of 0, and hence the local minima is in $\{\alpha_1 = 0, k\}$. Substituting these two values in G, we see that $G(\alpha_1, k) = 0$, which is the minimum. Therefore, $G(\alpha_1, k) \geq 0$ for $\alpha_1 \geq 0$ and $0 < k \leq 1$.

Hence the theorem. □

To get a physical perspective of these results, let us analyze the geometric relation of the function G and the histograms estimation methods. G is a positive function in the region $\alpha_1 \geq 0$, $0 < k \leq 1$. When $\alpha_1 \to 0$, $G \to 0$. This means that for small values of α_1, G is also small. Since $\alpha_1 = \lambda_1 c$, the value of α_1 depends on λ_1 and c. When c is small, G is very close to its minimum, 0, and hence both probabilities, p_1 and p_2, are very close. This behavior can be observed in Figure 2.

In terms of histogram methods and QEPs, when c is small, the optimal and the sub-optimal QEP are very close. Since histogram methods such as Equi-width and Equi-depth produce a larger error than the R-ACM and the T-ACM, the former are less likely to find the optimal QEP than the latter.

On the other hand, G is very small when λ_1 is close to 0. This means that $\text{Var}[X_1]$ is very large. Since $\text{Var}[X_1] \leq \text{Var}[X_2]$, $\text{Var}[X_2]$ is also very large, and both are close each other (In Figure 1, we would observe almost flat curves for both distributions). Random variables for histogram methods such as Equi-width

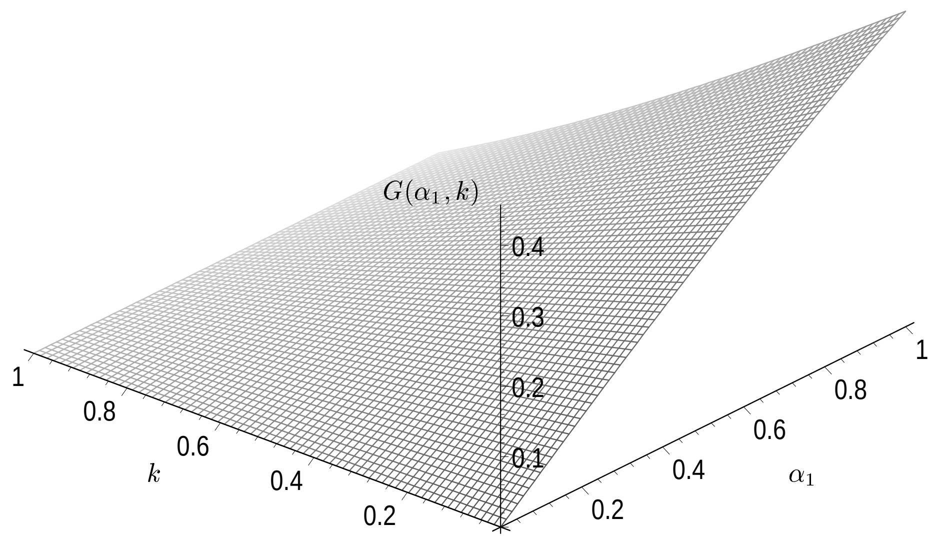


Figure 2: Function $G(\alpha_1, k)$ plotted in the ranges $0 \le \alpha_1 \le 1$ and $0 \le k \le 1$.

and Equi-depth yield similar error estimation distributions with large and similar variances. Hence, the probabilities p_1 and p_2 are quite close, and consequently, similar results are expected for these estimation methods. As a consequence of this, the results of Theorems 1 and 2 are not too meaningful in the absence of the new histogram methods, the R-ACM and the T-ACM, which effectively imply random variables with smaller variances and with underlying random variables quite different from those implied by the Equi-width and the Equi-depth methods.

4.2 Analysis Considering Normal Distributions

For the analysis done in this section, we consider that we are given two histogram methods, M_1 and M_2, for which the probabilities of choosing optimal or suboptimal QEPs are represented by two normally distributed random variables, X_1 and X_2, whose means are μ_1 and μ_2, and whose variances are σ_1^2 and σ_2^2 respectively.

Although the model using normal distributions is more realistic in real life problems, the analysis becomes impossible because there is no closed-form algebraic expression for integrating the normal probability density function. Alternatively, we have used numerical integration and we have obtained rather representative values for which the implication between efficiency and optimality is again corroborated.

Without loss of generality, if the mean cost value of the optimal QEP is μ_1, by shifting the origin by μ_1, we again assume that the cost of the best QEP is 0, which is the mean of these two random variables. The cost of the second best QEP is given by another two random variables (one for M_1 and the other one for M_2) whose mean, $\mu_2 > 0$, is the same for both variables. We also assume that, by scaling both distributions[4], the variance of M_1 choosing the optimal QEP is 1. An example will help to clarify this.

[4]This can be done by multiplying σ_1^2 and σ_2^2 by σ_1^{-2}, and μ_1 and μ_2 by σ_1^{-1}. This is a particular case of the simultaneous diagonalization between d-dimensional normal random vectors for which $d = 1$ [10].

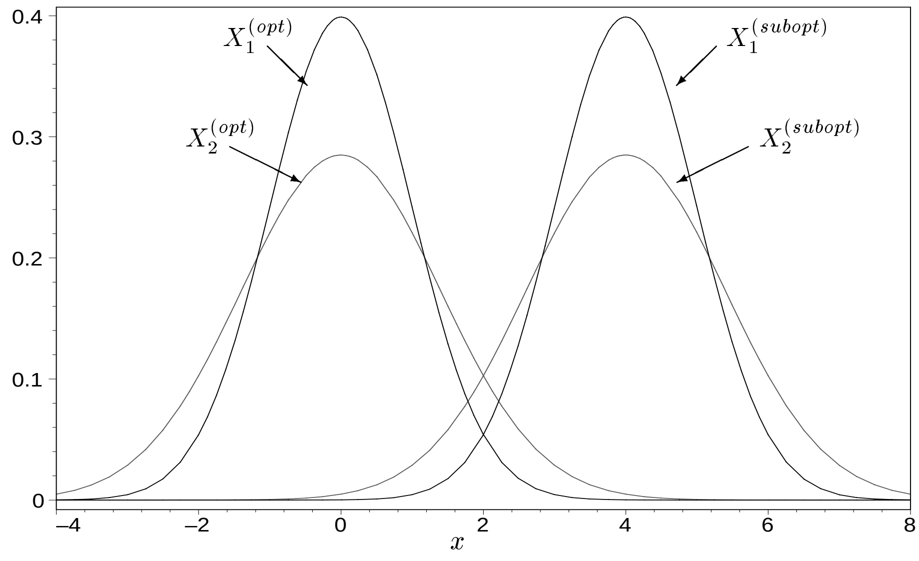


Figure 3: An example showing the probability density function of four normal random variables whose parameters are $\sigma_1 = 1$, $\sigma_2 = 1.4$, $\mu_1 = 0$, and $\mu_2 = 4$.

Example 2. Suppose that M_1 chooses the optimal QEP with probability represented by the normal random variable $X_1^{(opt)}$ whose mean is 0 and standard deviation is $\sigma_1 = 1$. This histogram method also estimates the cost of another sub-optimal QEP according to $X_1^{(subopt)}$ whose mean is 4 and $\sigma_1 = 1$.

M_2 is another histogram method that estimates the cost of the optimal QEP with probability given by $X_2^{(opt)}$ whose parameters are $\mu = 0$ and $\sigma_2 = 1.4$. Another sub-optimal cost value is obtained with probability given by $X_2^{(subopt)}$ whose parameters are $\mu = 4$ and $\sigma_2 = 1.4$.

Observe that $\sigma_1 < \sigma_2$, and hence we are expecting that the probability of M_1 making a wrong decision is smaller than that of H_2. The probability density functions for these four random variables are depicted in Figure 3. Note that, as in the doubly exponential distribution, given a particular value of x, if its probability under $X_1^{(opt)}$ is high, then the area for which M_1 makes the wrong decision (i.e. its cumulative probability under $X_1^{(subopt)}$) is small. Since these two quantities are multiplied and integrated, the final value is smaller than that of M_2, as σ_2 is significantly higher than $\sigma_1 = 1$. This is what we formally show below. $\square$

Result 1. [5]

Suppose that:

- M_1 and M_2 are two query result size estimation methods.
- X_1 and X_2 are two normally distributed random variables that represent the cost values of the *optimal* QEP obtained by M_1 and M_2 respectively.

[5]We cannot claim this result as a theorem, since the formal analytic proof is impossible. This is because there is no closed-form expression for integrating the Gaussian probability density function. However, the computational proof that we present renders this to be more than a conjecture.

- X_1' and X_2' are another two normally distributed random variables representing the cost values of a *non-optimal* QEP obtained by M_1 and M_2 respectively.
- $0 = \mathrm{E}[X_1] = \mathrm{E}[X_2] \leq \mathrm{E}[X_1'] = \mathrm{E}[X_2'] = \mu$.

Let p_1 and p_2 be the probabilities that M_1 and M_2 respectively make the wrong decision. Then,

$$\text{if } \mathrm{Var}[X_1] = \mathrm{Var}[X_1'] = \sigma_1^2 \leq \sigma_2^2 = \mathrm{Var}[X_2] = \mathrm{Var}[X_2'], \quad p_1 \leq p_2 \ .$$

Computational Proof. To achieve this proof, we proceed by doing the same analysis that we did for the doubly exponential distributions. If we consider a particular value x, the probability that x leads to a wrong decision made by M_1, is given by:

$$I_1 = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(u-\mu)^2}{2\sigma_1^2}}\, du \,. \tag{24}$$

The probability that H_1 makes the wrong decision for *all* the values of x is obtained by integrating the function resulting from multiplying every value of I_1 for each x and the probability density function of $X_1^{(opt)}$, which results in:

$$p_1 = \int_{-\infty}^{\infty} I_1 \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{x^2}{2\sigma_1^2}}\, dx \ . \tag{25}$$

Similarly, p_2 can also be expressed as follows:

$$p_2 = \int_{-\infty}^{\infty} I_2 \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{x^2}{2\sigma_2^2}}\, dx \ \ , \tag{26}$$

where I_2 is obtained in the same way as in (24) for the distribution with variance σ_2^2.

Since there is no closed-form algebraic expression for integrating the normal probability density function, no analytical solution for proving that $p_1 \leq p_2$ can be formalized.

Alternatively, we have invoked a computational analysis by calculating these integral for various representative values of σ_1 and σ_2 by using the trapezoidal rule. The values of $G = \frac{p_2}{p_1} \geq 1$ (i.e. for $1 \leq \sigma_1 \leq 10$ and $1 \leq \sigma_2 \leq 10$, where $\sigma_1 \leq \sigma_2$) are depicted in Table 1 in the form of a *lower-diagonal* matrix. All the values of the *upper-diagonal* matrix (not shown here) are less than unity. Note that by making the value of $\sigma_1 = 1$, the analysis reduces to the first and second columns of this table. For example, if $\sigma_1 = 1$ and $\sigma_2 = 2$, $\frac{p_2}{p_1} \approx 33.6276$. For more neighboring values of σ_1 and σ_2, e.g. $\sigma_1 = 9$ and $\sigma_2 = 10$ ($\sigma_1 = 1$ and $\sigma_2 \approx 1.2345$

$\sigma_1 \rightarrow$ $\sigma_2 \downarrow$	1.00	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00	10.00
1.00	1.0000									
2.00	33.6276	1.0000								
3.00	73.9210	2.1982	1.0000							
4.00	102.5081	3.0483	1.3867	1.0000						
5.00	122.1988	3.6339	1.6531	1.1921	1.0000					
6.00	136.2472	4.0516	1.8431	1.3291	1.1150	1.0000				
7.00	146.6138	4.3599	1.9834	1.4303	1.1998	1.0761	1.0000			
8.00	154.7078	4.6006	2.0929	1.5092	1.2660	1.1355	1.0552	1.0000		
9.00	161.0448	4.7891	2.1786	1.5710	1.3179	1.1820	1.0984	1.0410	1.0000	
10.00	166.1716	4.9415	2.2480	1.6211	1.3598	1.2196	1.1334	1.0741	1.0318	1.0000

Table 1: Ratio between the probability of making the wrong decision for two normally distributed random variables whose standard deviations are σ_1 and σ_2.

after scaling), $\frac{p_2}{p_1} \approx 1.0318$, which is very close to unity. The ratio for $\sigma_1 = 1$ and $\sigma_2 = 10$ is much bigger, i.e. more than one hundred times. $\square$

In order to get a better perspective of the computational analysis, we study the behavior of the function $G = \frac{p_2}{p_1}$. Using the values of G given in Table 1, we have plotted this function in the three-dimensional space as $G(\sigma_1, \alpha_1)$, where $\alpha_1 = k\sigma_1$, $1 \leq k \leq 10$. The plot is depicted in Figure 4. In order to enhance the visualization of G, we have approximated it by using the regression utilities of the symbolic mathematical software package Maple V [6]. When $k = 1$, the surface lies on the $z = 0$ plane, in the form of a straight line $x = y$ (labelled "$k = 1$ or $\sigma_1 = \sigma_2$" in the figure). This is the place in which G reaches its minimum, when both histograms have identical variances. When k is larger (i.e. $k = 10$), the function G becomes much larger (up to 166 in in Table 1).

The analysis above shows the importance of the variance in deciding on a histogram. When k is small, it implies that the optimal and sub-optimal QEP are very close. Therefore, histogram methods like Equi-width and Equi-depth are less likely to find the optimal QEP, since they produce larger errors than histogram approximation methods such as the R-ACM and the T-ACM. The latter produce very small errors, and hence, when comparing any of them with the Equi-width or Equi-depth, we will have a much larger value of k. This will be reflected in our empirical results presented in the next section.

5 The Join Ordering Model

In this section, we describe the formal join ordering model that we have developed to test the efficiency of the methods (or for that matter, any query optimization strategy) discussed above. Since the estimation of the accuracy of these methods is done under *synthetic* data distributions, it requires a formal relational database model in which the relations are populated randomly according to some statistical distributions,

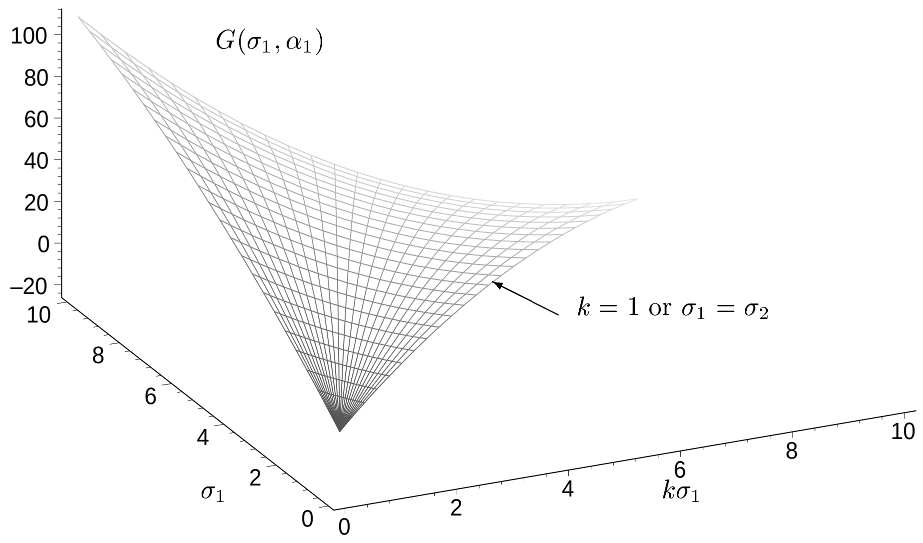


Figure 4: Function $G(\sigma_1, k\sigma_1)$ plotted in the ranges $1 \leq \sigma_1 \leq 10$ and $1 \leq k\sigma_1 \leq 10$, where $\sigma_2 = k\sigma_1$.

and the queries also obey user-defined query distributions.

The database has n relations, called $R_1, R_2, \ldots, R_n$. Each relation has n attributes, called A_{ij}, such that A_{ij} is the j^{th} attribute of the relation R_i. It will be presently clear that the attribute A_{ij} symbolically represents the joining attribute for R_i and R_j, and the attribute A_{ii} represents the data inherent to relation R_i alone.

Among the various *join* operations available in relational database systems the model incorporates the equi-join. Clearly, this can be extended easily to other join operations.

Definition 1. The equi-join operation, denoted $R_1 \bowtie_{X=Y} R_2$, combines all the tuples in the relations R_1 and R_2 whenever the value of the attribute X from relation R_1 is equal to the value of the attribute Y from relation R_2.

We represent a query as an ordered sequence of integers as a permutation of the integers $[1..n]$. Each number in the sequence represents the relation number, and the query itself is performed as per the following definition.

Definition 2. A query is represented by the string $Q = i_1 i_2 \ldots i_n$, such that $1 \leq i_j \leq n$ and $1 \leq i_k \leq n$, for all i_j, i_k satisfying $j, k = 1, \ldots, n$, $i_j \neq i_k$. The query join resulting from the string $i_1, i_2, \ldots, i_n$ is the following:

$$J = R_{i_1} \bowtie_{A_{i_1 i_2} = A_{i_2 i_1}} R_{i_2} \bowtie_{A_{i_2 i_3} = A_{i_3 i_2}} \ldots \bowtie_{A_{i_{n-1} i_n} = A_{i_n i_{n-1}}} R_{i_n} \ . \tag{27}$$

This string is called *query string*. For the sake of simplicity, the query join given in (27) is referred below to as $R_{i_1} \bowtie R_{i_2} \bowtie \ldots \bowtie R_{i_n}$ or simply as J.

Example 3. Suppose that the user requests a query represented by the following query string: $Q_1 = 1\,2\,3\,4$. This means that R_1 is joined with R_2 on the attributes A_{12} and A_{21}, respectively, generating the relation R_{12}. R_{12} and R_3 are joined on the attributes A_{23} and A_{32} respectively, generating R_{123}. Consequently, R_{123} and R_4 are joined on the attributes A_{34} and A_{43} respectively.

Consider another query string, $Q_2 = 1\,3\,2\,4$. Unlike in the query string Q_1, in this query, R_1 is joined with R_3 on the attributes A_{13} and A_{31} respectively, generating R_{13}. R_{13} and R_2 are joined on the attributes A_{32} and A_{23}, respectively, which generates R_{132}. Finally, R_{132} and R_4 are joined over the attributes A_{24} and A_{42}, respectively.

Note that we do not consider the order in which the join operations are done. The query join given by the string Q_1 may be solved by first joining R_3 and R_4 generating the relation R_{34}, then joining R_2 and R_{34} which generates R_{234}, and finally joining R_1 and R_{234}. Observe that different orderings on the sequence of join operations represent different QEPs.

Since the equi-join operation is associative [33], the join given in (27) can be done in many different ways, and the resulting relation is the same in all the cases. As discussed in Section 2, given a query string with n relations, the total number of different join orders is given by $\frac{(2(n-1))!}{(n-1)!}$. It turns out that performing *all* the different join orders is a very complex problem. We have devised a method that analyzes all the different join orders given a query. To describe this, we introduce the following definition:

Definition 3. Given a query string, $Q = i_1 i_2 \ldots i_n$, a *join order* or *Query Evaluation Plan (QEP)* is given by another string $S = j_1 j_2 \ldots j_{n-1}$, where $1 \leq j_k \leq n$ and $1 \leq j_l \leq n$, for all j_k and j_l satisfying $k, l = 1, \ldots, n$, $j_k \neq j_l$, such that the query join $J = R_{i_1} \bowtie R_{i_2} \bowtie \ldots \bowtie R_{i_n}$ is performed according to the procedure described in Algorithm **Join_Order_Selection** given below.

For the sake of simplicity the resulting join operation $(R_i \bowtie_X R_j)$ is referred to as $R_i R_j$, or simply R_{ij}.

Algorithm 1 : Join_Order_Selection

```
Input: A query string, $J = R_{i_1} \bowtie R_{i_2} \bowtie \ldots \bowtie R_{i_n}$.
       A QEP string, $S = j_1 j_2 \ldots j_{n-1}$.
Output: The resulting relation $R_{i_1 i_2 \ldots i_n}$.
Method:
 while $S$ is not empty do
   Select the smallest index from $S$, say $j_k$.
   $R_{j_k j_{k+1}} \leftarrow R_{j_k} \bowtie R_{j_{k+1}}$
   Remove $j_k$ from $S$.
   Replace $R_{j_k}$ by $R_{j_k j_{k+1}}$ in $J$.
   Remove $R_{j_{k+1}}$ from $J$.
 endwhile
 end Algorithm Join_Order_Selection
```

Example 4. Consider the following join query: $J = R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4$. Suppose that the QEP to perform all the join operations is $S = 2\,1\,3$. A trace of the different join evaluations is given in Figure 5. The

$$\begin{tikzcd}
\begin{matrix} R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4 \\ 2 \qquad \mathbf{1} \qquad 3 \end{matrix} \arrow[d] \\
\begin{matrix} R_1 \bowtie R_{23} \bowtie R_4 \\ \mathbf{2} \qquad 3 \end{matrix} \arrow[d] \\
\begin{matrix} R_{123} \bowtie R_4 \\ \mathbf{3} \end{matrix} \arrow[d] \\
R_{1234}
\end{tikzcd}$$

Figure 5: Evaluation of the query evaluation plan $S = 2\,1\,3$ for the join query $J = R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4$, given in Example 4.

smallest index found in the first step is 1 which is in the second position. Thus the second join in the query, namely $R_2 \bowtie R_3$ is first executed, and hence the join $R_2 \bowtie R_3$ is performed, resulting in R_{23}. In the next step, the smallest index in S is 2 at position 1. Hence, the join $R_1 \bowtie R_{23}$ is performed and R_{123} is obtained. Finally, the only index left in S is 3 which is at position 3. The third join to be done, join $R_{123} \bowtie R_4$, is performed obtaining the final relation R_{1234}.

We have described how to perform a query evaluation plan given a string query. Note that the QEP is also represented by a string, and for n relations, the length of the join order string is $n - 1$. Since all the indexes in the string are different, there are $(n - 1)!$ different strings representing the $(n - 1)!$ QEPs. All we need now is to consider how to perform the join for the $(n - 1)!$ different join order strings, calculate the number of operations for each of them, and choose the string that leads to the minimum cost. This effectively models our join ordering selection mechanism.

As discussed in Section 1, this is an NP-hard problem, in which the number of different join orders increases with the size of the query. In order to avoid time-consuming computations, different methods have been devised, such as pruning and estimation. We use the estimation methods discussed above to *approximate* the size of the query.

6 Join Estimation

The aim of the query optimization problem is to minimize two crucial quantities: the *response time*, which means how quickly the query is executed, and the *resource consumption*, i.e. the computer resources (memory, CPU, etc.), used to execute the query. We first deal with some aspects in the calculation of the cost of a query.

The cost of a query, in terms of the response time, is given by the *query result sizes*, the *query result distributions*, the *disk access costs*, and other non-relevant factors. Of these, only the number of read/write disk operations are the dominant factors [33]. We actually consider the cost of a query in terms of join cardinalities.

Given two relations, R_1 and R_2, to be joined by a common attribute, X, we calculate the cost of an equi-join query, $R_1 \bowtie_X R_2$, as the size of the resulting relation. That is:

$$\mathcal{C} = |(R_1 \bowtie_X R_2)| \ . \tag{28}$$

Our problem now is one of calculating the cost of an equi-join that includes more than two relations. Using Algorithm **Join_Order_Selection**, we can calculate the total number of operations for performing the join $J = R_{i_1} \bowtie R_{i_2} \bowtie \ldots \bowtie R_{i_n}$ subject to the join order $S = j_1 j_2 \ldots j_{n-1}$, as follows:

$$\sum_{k=1}^{n-1} |R_k| \ , \tag{29}$$

where R_k is the intermediate relation generated in the k^{th} step of Algorithm **Join_Order_Selection**, i.e. $R_k = R_{j_k} R_{j_{k+1}}$.

We have discussed how to calculate the number of operations of the real query given a QEP. The main goal of query optimization is to discern, *in advance*, what is the optimal (or close-to optimal) QEP for a given query. We make use of the estimation methods discussed earlier to approximate the cost of the different QEPs and choose the best one.

We use the procedure described in Algorithm **Join_Order_Selection** to calculate the estimated number of operations required to perform the join $J = R_{i_1} \bowtie R_{i_2} \bowtie \ldots \bowtie R_{i_n}$ subject to the join order $S = j_1 j_2 \ldots j_{n-1}$. This can be estimated by adding the estimated sizes of the intermediate relations as if they were created in every step of Algorithm **Join_Order_Selection**.

To estimate the total number of operations needed to perform the join $J = R_{i_1} \bowtie R_{i_2} \bowtie \ldots \bowtie R_{i_n}$ subject to the join order $S = j_1 j_2 \ldots j_{n-1}$, we add all the costs of *all* the intermediate relations as follows:

$$\mathcal{C}_n = \sum_{k=1}^{n-1} \mathcal{C}_k \ , \tag{30}$$

where $\mathcal{C}_k$ is the estimated cost at step k.

Since, at each step, two relations of J are merged into a new one, the relations in J can be of two types, referred to as primitive or derivative. In all the possible scenarios encountered we shall show that the result of Theorem 3 (stated below) can be recursively invoked so as to calculate the cost at every step and obtain

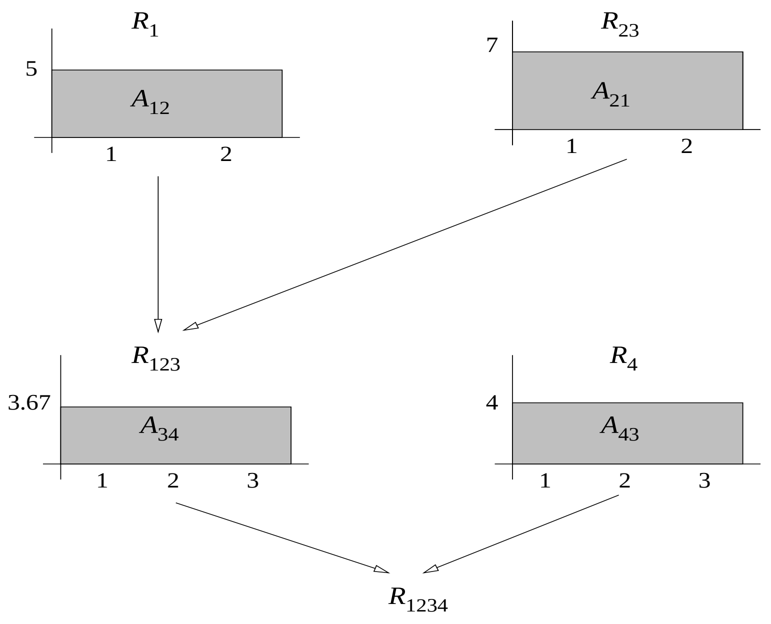


Figure 6: Cost estimation for the join $J = R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4$, subject to the order string $S = 2\,1\,3$, given in Example 5.

the total cost given by (30). An example will help to clarify this.

Example 5. Consider the join query given in Example 4. Suppose that the sizes of the relations are $|R_1| = 10$, $|R_2| = 14$, $|R_3| = 11$, and $|R_4| = 12$, and that each attribute value has only one sector.

The first join to be estimated is $R_2 \bowtie_{A_{23}=A_{32}} R_3$. Supposing that A_{23} and A_{32} have 3 attribute values and a histogram with one sector, the cost is estimated as $\mathcal{C}_1 = |\widehat{R}_{23}| = \frac{(14)(11)}{(14)(11)}(4.67)(3.67) = 17.14$.

The second join, $R_1 \bowtie_{A_{12}=A_{21}} R_{23}$, is estimated on the attributes A_{12} and A_{21}, each having two attribute values and a histogram with one sector, as depicted in Figure 6. The cost of the join is estimated as follows: $\mathcal{C}_2 = |\widehat{R}_{123}| = \frac{(10)(17.14)}{(10)(14)}(5)(7) = 42.85$. As we observe, $|\widehat{R}_{23}|$, the estimated size of R_{23}, is recursively used in this estimation.

The last join to be estimated is $R_{123} \bowtie_{A_{34}=A_{43}} R_4$. Suppose that A_{34} and A_{43}, whose histograms are depicted in Figure 6, have three attribute values. The estimated cost of the join results in: $\mathcal{C}_3 = |\widehat{R}_{1234}| = \frac{(42.85)(12)}{(11)(12)}(3.67)(4) = 57.19$.

We now use (30) to estimate the cost of all the joins: $\mathcal{C}_4 = \mathcal{C}_1 + \mathcal{C}_2 + \mathcal{C}_3 = 17.14 + 42.85 + 57.19 = 117.18$.

Theorem 3. Consider the join $R_a \bowtie_{A_{jk}=A_{kj}} R_b$, in which R_a and R_b can be either primitive or derivative relations, where a is a string of integers whose *last* symbol is j, and b is a string of integers whose *first* symbol is k.

If A_{jk} and A_{kj} are independent of R_a and R_b respectively, the size of $R_a \bowtie R_b$, $|\widehat{R}_{ab}|$, is estimated as:

$$|\widehat{R}_{ab}| = \frac{|\widehat{R}_a||\widehat{R}_b|}{|R_j||R_k|}\sum_{i=1}^{m}\widehat{x}_i\widehat{y}_i \ , \tag{31}$$

where $\widehat{x}_i$ and $\widehat{y}_i$ are the estimated frequencies for the i^{th} value of A_{jk} and A_{kj} respectively.

Proof. First of all, observe that the join is between R_a and R_b. Since a is a string of integers with j as the last symbol, and b is a string of integers with k as the first symbol, by the model of computation the join is done on the attributes A_{jk} and A_{kj} respectively.

To prove the theorem, we consider three mutually exclusive and collectively exhaustive cases:

Case 1: R_a and R_b are primitive.

The proof of this case is straightforward, since $a = j$ and $b = k$, which implies that $R_a = R_j$ and $R_b = R_k$. Hence:

$$|\widehat{R}_{ab}| = \frac{|\widehat{R}_a||\widehat{R}_b|}{|R_j||R_k|}\sum_{i=1}^{m}\widehat{x}_i\widehat{y}_i = \frac{|R_j||R_k|}{|R_j||R_k|}\sum_{i=1}^{m}\widehat{x}_i\widehat{y}_i = \sum_{i=1}^{m}\widehat{x}_i\widehat{y}_i \ . \tag{32}$$

Case 2: Without loss of generality, suppose that R_a is a primitive relation and R_b is a derivative relation[6]. Thus, $R_a = R_j$ and $R_b = R_{ki_1i_2\ldots}$

The estimated size of R_{ab} can be calculated as:

$$|\widehat{R}_{ab}| = \sum_{i=1}^{m}\widehat{\widehat{x}}_i\widehat{\widehat{y}}_i \ , \tag{33}$$

where $\widehat{\widehat{x}}_i$ and $\widehat{\widehat{y}}_i$ are the estimates for the i^{th} value of A_{jk} and A_{kj} in R_a and R_b respectively.

Since R_a is primitive, $R_a = R_j$. Hence:

$$\widehat{\widehat{x}}_i = \frac{|\widehat{R}_a|}{|R_j|}\widehat{x}_i \ , \tag{34}$$

where $\widehat{x}_i$ is the estimated frequency for the i^{th} value of A_{jk}.

On the other hand, the probability that v_i is a value of R_b is given by:

$$p(v_i|R_b) = \frac{\widehat{\widehat{y}}_i}{|\widehat{R}_b|} \ . \tag{35}$$

[6]The case in which R_a is derivative and R_b is primitive follows by simmetry.

We also know that,

$$p(v_i|R_k) = \frac{\widehat{y}_i}{|R_k|} \ , \tag{36}$$

where $\widehat{y}_i$ is the estimate for the i^{th} value of A_{kj} in R_k.

By invoking the assumption that A_{kj} is independent of R_b, and using (35) and (36), we have:

$$\widehat{\widehat{y}}_i = p(v_i|R_b)|\widehat{R}_b| = p(v_i|R_k)|\widehat{R}_b| = \frac{|\widehat{R}_b|}{|R_k|}\widehat{y}_i \ . \tag{37}$$

Using (33), (34), and (37), the estimated size of R_{ab} yields:

$$|\widehat{R}_{ab}| = \frac{|\widehat{R}_a||\widehat{R}_b|}{|R_j||R_k|}\sum_{i=1}^{m}\widehat{x}_i\widehat{y}_i \ . \tag{38}$$

Case 3: Suppose that R_a and R_b are derivative relations. Thus, $R_a = R_{i_1 i_2 \ldots j}$ and $R_b = R_{k i_1 i_2 \ldots}$.

The probability that u_i is a value of R_a is given by:

$$p(u_i|R_a) = \frac{\widehat{\widehat{x}}_i}{|\widehat{R}_a|} \ . \tag{39}$$

Again, invoking the assumption that A_{jk} is independent of R_a, we have:

$$\widehat{\widehat{x}}_i = p(u_i|R_a)|\widehat{R}_a| = p(u_i|R_j)|\widehat{R}_a| = \frac{|\widehat{R}_a|}{|R_j|}\widehat{x}_i \ . \tag{40}$$

since $p(u_i|R_j)$ is the probability that u_i is a value for the attribute A_{jk} in R_j.

Using (33), (37), and (40), the estimated size of R_{ab} is calculated as:

$$|\widehat{R}_{ab}| = \frac{|\widehat{R}_a||\widehat{R}_b|}{|R_j||R_k|}\sum_{i=1}^{m}\widehat{x}_i\widehat{y}_i \ . \tag{41}$$

Hence the theorem. □

In our model (and in all database query systems worldwide), we have assumed independence between attributes and relations. We briefly consider what happens if there is attribute dependency.

Suppose, for example, that our database has three relations R_1, R_2, and R_3 with three attributes each. To estimate the cost of the join $R_1 \bowtie R_2$ we need the frequencies for A_{12} and A_{21}. Since both are primitive relations, the frequencies can be obtained directly from the unconditional probabilities.

The problem arises when we are to do the estimation of $R_{12} \bowtie R_3$. To achieve this, we need the frequencies for each value, v_i, of A_{23} in R_{12}. These frequencies are not exactly available for R_{12} - they are only available for R_2. To obtain $p(v_i|R_{12})$ we would need to store $p(v_i|u_j)$, $i, j = 1, \ldots, m$, where u_j is the j^{th} value of the attribute A_{21}. This is because the new attribute was produced as a result of $R_1 \bowtie R_2$ being performed on the old attribute values. In general, if all the attributes have m possible values, for n relations with n attributes we would need to store $(n-1)n^2m^2$ conditional probabilities. As this number grows with the number of relations and the number of attributes, it is practically *impossible* to consider dependency.

Note that the estimated frequency of a particular attribute value, $\widehat{x}_i$, is approximated by using any of the methods discussed in Section 3. Therefore, the *accuracy* of estimation of the total number of operations to perform a join query depends on the method used to construct the histograms. This property is discussed below by showing, analytically and empirically, the impact of the error estimation in the selection of the best QEP.

7 Database Generation

In this section, we discuss the statistical distributions used in the generation of the databases. We have carried out two sets of experiments for the methods discussed above. In one set of experiments, the databases were generated using the uniform distribution modified under a multi-fractal decomposition. These databases were used to compare the R-ACM with the Equi-width and the Equi-depth methods. In the other set of experiments, we have tested the T-ACM with the Equi-width and the Equi-depth with distributions which are more suitable for linear modeling of attribute values.

Note that we have not used the current database benchmarks (the TPC-H) to test the efficiency of the various QEPs selected by the methods. This is because, to test a single query, we need to find the optimal QEP, which implies that for a single query, we must perform *all* the *exponential* number of possible QEPs. There seems to be no way to achieve this (since the problem is NP-hard) other than by a "brute force" method. This "brute force" searching becomes practically *impossible* even for relatively small databases with a smaller number of tuples and relations. The test is even more infeasible for scalable databases with scalable queries as in the TPC family of benchmarks.

Although we have presented two new histogram methods, the R-ACM and the T-ACM, the question that has to be answered in a practical setting, is that of determining which histogram is applicable for a particular database. While this problem is, generally speaking, open, there is one issue that is reasonably clear. If the density of the attributes is reasonably flat, we could use both the R-ACM and the T-ACM. On the other hand, if the density is significantly sloped, it is meaningless to use the R-ACM. In such a case, it is obvious that the R-ACM would be a poor approximation and it would be futile to even attempt such a modeling.

In this case, the better approximation should be the T-ACM, where the slope of the trapezoid is made to approximate the slope of the density function. This also answers for why we have tested the ACMs with the respective distributions.

7.1 Database Generation for R-ACM Evaluation

The databases used to test the accuracy of R-ACM are generated using the uniform distribution modified by the multi-fractal decomposition. In order to populate the database we used a probability list for each attribute value as follows.

Definition 4. Consider an attribute A, whose set of values is $\mathcal{V} = \{v_1, \dots, v_m\}$. The probability set of A is given by $P = \{p_1, \dots, p_m\}$, such that v_i is randomly generated with probability p_i.

Using this probability set, the values for each attribute in each table are randomly generated in such a way that

$$E(x_i) = Np_i \ , \tag{42}$$

where $E(x_i)$ is the expected number of times that v_i is in the relation, and N is the number of tuples in the relation.

In order to generate a database, we first ensure that the distribution of the probability set is uniform. We do this by assigning the probability values by setting the value p_i to $\frac{1}{m}$, for $i = 1, \dots, m$. Hence the expected number of occurrences of a value in a table is given by $E(x_i) = \frac{N}{m}$.

The next step of the database generation consists of transforming the distribution of P to a non-uniform multi-fractal distribution. In order to achieve this, we used the multi-fractal decomposition described in [9]. This method receives as input a decomposition probability $p \neq 0.5$ as a parameter. The basic idea of multi-fractal decomposition is to multiply the first half of the probability set by p, and the second half by $1 - p$. The same procedure is recursively applied to each half.

In our model, we have further augmented the multi-fractal decomposition by a sliding index, k. The output list of the previous recursive scaling, $\{p_1, p_2 \dots, p_m\}$, is transformed into a new normalized probability list, $\{p_k, p_{k+1}, \dots, p_m, p_1, p_2, \dots, p_{k-1}\}$, where k is a random integer such that $1 \leq k \leq m$.

The procedure that implements this method is described in Algorithm **Multi_Fractal_Decomposition**. The output of the algorithm, P, must be normalized[7].

7.2 Database Generation for T-ACM Evaluation

The databases used in the experiments for testing the T-ACM and comparing it with the traditional methods, used a particular probability distribution that we called the *noisy* TD, which is based an underlying exact TD

[7] P is normalized by being transformed such that $\sum_{i=1}^{m} p_i = 1$.

distribution. We work with this distribution because comparing the schemes with the uniform/multi-fractal distribution is meaningless if the model of linearity is not valid in the first place.

In the exact TD distribution, the probability set of a given attribute, P, is divided in k sub-sets, $P = \{P_1, \ldots, P_k\}$. Each sub-set has the same range values as the corresponding range in the Equi-width histogram. The probability sub-set P_1 is generated by choosing two random numbers, a and b; $p_j \in P_1$ is calculated as $p_j = (v_j - v_1)\frac{b-a}{|P_j|}$, where $|P_j|$ is the number of values of P_j. The next probability sub-sets are generated by setting a for the new range as the 'b' of the previous range, and choosing a new random number b for the subsequent range. Once all the probability sub-sets are generated, P is normalized. A typical probability distribution TD is depicted in Figure 7. For a given attribute, generated with this distribution, the shape of its T-ACM histogram will be similar to that of the graphic drawn in Figure 7.

Algorithm 2 : Multi_Fractal_Decomposition

```
Input : The Probability Set, $P$. The decomposition probability, $p$.
Output : The transformed probability set, $P$, translated by a sliding random value, $k$.
Method:
  procedure Multi_Fractal(var $P$ : ProbabilitySet; left, right : integer)
     middle ← (left + right) div 2
    for i:=left to middle do
      $p_i \leftarrow p_i p$
    endfor
    for i:=middle+1 to right do
      $p_i \leftarrow p_i(1-p)$
    endfor
    if middle - left $\geq 1$ then
      Multi_Fractal($P$, left, middle)
    endif
    if right - middle + 1 $\geq 1$ then
      Multi_Fractal($P$, middle+1, right)
    endif
  endprocedure
  begin
     Multi_Fractal($P$, 1, $m$)
     return $\{p_1, p_2, \ldots, p_m\} \leftarrow \{p_k, p_{k+1}, \ldots, p_m, p_1, p_2, \ldots, p_{k-1}\}$
  end Algorithm Multi_Fractal_Decomposition
```

The database is generated in such a way that all the pairs of attributes to be joined have the same distribution. In other words, the attribute A_{ij} has the same distribution as the attribute A_{ji}, for $i, j = 1, \ldots, n$, $i \neq j$.

The random values for each attribute for the noisy TD are generated using the exact TD distribution with some noise κ. The expected number of occurrences of v_i in the relation is $N(1 - o_i)p_i$, where o_i is a randomly generated real number such that $-\kappa \leq o_i \leq \kappa$, $\sum_{i=1}^{m} o_i = 0$, and N is the number of tuples in the relation.

Example 6. Suppose that $p_i = .08$, $\kappa = .2$, and $N = 100$. Suppose also that o_i is generated randomly

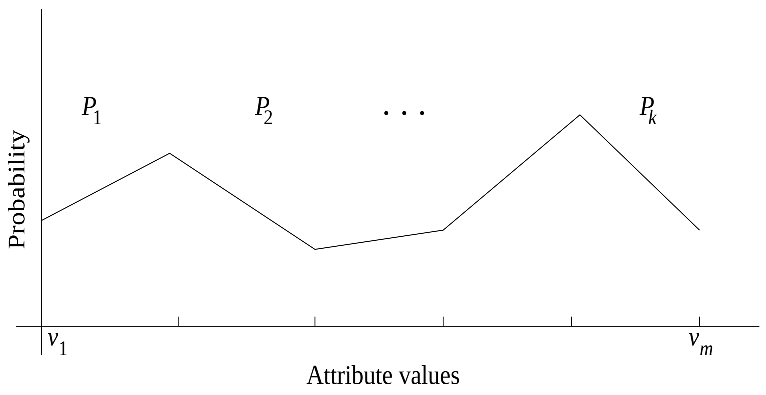


Figure 7: A typical probability set distribution generated by TD.

resulting in $o_i = .17$. The expected number of tuples for the attribute value v_i in the relation is $.08(1 - .17)100 = 6.64$.

One of the problems encountered in the implementation is the following. We expect that v_i occurs $N(1 - o_i)p_i$ times in the relation. According to the *law of large numbers* the actual number of times that v_i occurs in the relation is equal to $N(1 - o_i)p_i$ as $N \to \infty$. If we want these numbers to be arbitrarily close each other, we need N to be large. But there is a practical reason for not choosing N quite large. In order to test the methods, we need to perform an equi-join of n relations, which grows exponentially as the number of tuples is larger.

In other words, if the number of tuples in the relations is small, the histograms are not so good and the estimations are poor. However, if the number of tuples is large, the estimates are good, but the equi-joins can not be achieved due to the sizes of the joined relations. This problem is aggravated by the fact that we do not perform any query pruning. That is, we process *all* the QEPs and find which is the one that minimizes the number of operations. Thus, even though the underlying model is exact, learning the model using the histogram is far from trivial. In spite of this, the results we have obtained are quite amazing.

8 Simulation Results

In this section, we discuss the empirical results obtained from some simulations with randomly generated databases. We have conducted two independent experiments, one of them is to compare the efficiency of R-ACM with the traditional methods, and the other one is related to the T-ACM.

In each experiment, we performed four independent runs. In each run, 100 random databases were generated. Each database was composed of six relations, each of them having six attributes. Each relation was populated with 100 tuples.

Simulation	**R-ACM**	**Equi-width**
1	26	12
2	24	15
3	35	11
4	29	15
Total	**114**	**53**

Table 2: Simulation results for R-ACM and Equi-width, after optimizing a query on 400 randomly generated databases.

Simulation	**R-ACM**	**Equi-depth**
1	35	12
2	42	13
3	46	8
4	46	8
Total	**169**	**41**

Table 3: Results of 400 runs for query optimization used to test the efficiency of R-ACM and Equi-depth.

8.1 Simulation for R-ACM and Traditional Methods

In order to compare the efficiency of the R-ACM with that of the Equi-width and the Equi-depth, we used 50 values per attribute. We set the number of bins for the Equi-width to be 22 and the number of bins of the Equi-depth to be as close to 22 as possible. The number of bins for the R-ACM is the smallest integer, s, such that $s \geq 11$ and τ is an integer; s is automatically calculated by the prototype by performing a sequential search over all possible values of τ. In order to be impartial with the evaluation, we set the number of bins for the R-ACM to be *approximately half* of that of the Equi-width and the Equi-depth, because the former needs twice as much storage as that of the latter.

The simulation results obtained from 400 independent runs, used to compare the efficiency of the R-ACM and the Equi-width, are given in Table 2. As in the subsequent tables, the column labeled "R-ACM" is the number of times that R-ACM is better than Equi-depth. The column labeled "Equi-width" indicates the number of times that Equi-width is superior to R-ACM. As we can observe from the last row, the R-ACM chooses the superior solution more than *three* times as often as the Equi-width. Table 3 shows the results obtained after comparing the R-ACM with the Equi-depth. Again, we observe the superiority of the R-ACM over the Equi-depth – the R-ACM chooses a superior solution almost *four* times as often.

8.2 Simulation for T-ACM and Traditional Methods

In this section we discuss the results obtained from the simulations performed to compare the efficiency of the T-ACM, the Equi-width, and the Equi-depth. In our experiments, we have used the noisy TD distribution to generate random databases with a noise level of 20%. Hence, the samples were generated with $\kappa = .2$, and $-.2 \leq o_i \leq .2$, for $i = 1, \ldots, m$. We set the number of attribute values to be 25, and the number of buckets

Simulation	**T-ACM**	**Equi-width**
1	26	3
2	25	7
3	24	8
4	18	5
Total	**93**	**23**

Table 4: Simulation results of 400 runs used to test the efficiency of T-ACM and Equi-width.

Simulation	**T-ACM**	**Equi-depth**
1	23	3
2	34	9
3	33	8
4	24	6
Total	**114**	**26**

Table 5: Experimental results obtained by performing query optimization on 400 randomly generated databases used to test the efficiency of T-ACM and Equi-depth.

to be 4.

The experimental results obtained after performing query optimization on 400 random databases are shown in Table 4. We observe the superiority of the T-ACM over the Equi-width – it chooses a superior solution more than *four* times as often as the Equi-width. In Table 5, we show the results of another set of runs used to compare the T-ACM and the Equi-depth. Again, the T-ACM chooses a superior solution more than *four* times as often as the Equi-depth. The power of T-ACM is obvious.

9 User Interface

In order to facilitate the use of the prototype, we have built a sophisticated user interface using the state-of-art Java programming language in a Visual Development Environment [13, 5, 8]. The main screen of the prototype is depicted in Figure 8. The user can choose among different options accessible by "double-clicking".

The option *Initialize Parameters* is used to set the the following data:

- Number of tuples: This specifies the number of tuples per relation.
- Number of Intervals: This parameter is the number of buckets that the histogram contains. It can be specified for each attribute individually or for all the attributes simultaneously.
- Range Values: This parameter assigns values that an attribute can take. It is the same for all the attributes.
- Miscellaneous: In this window (see Figure 9), the user can specify additional parameters, such as:

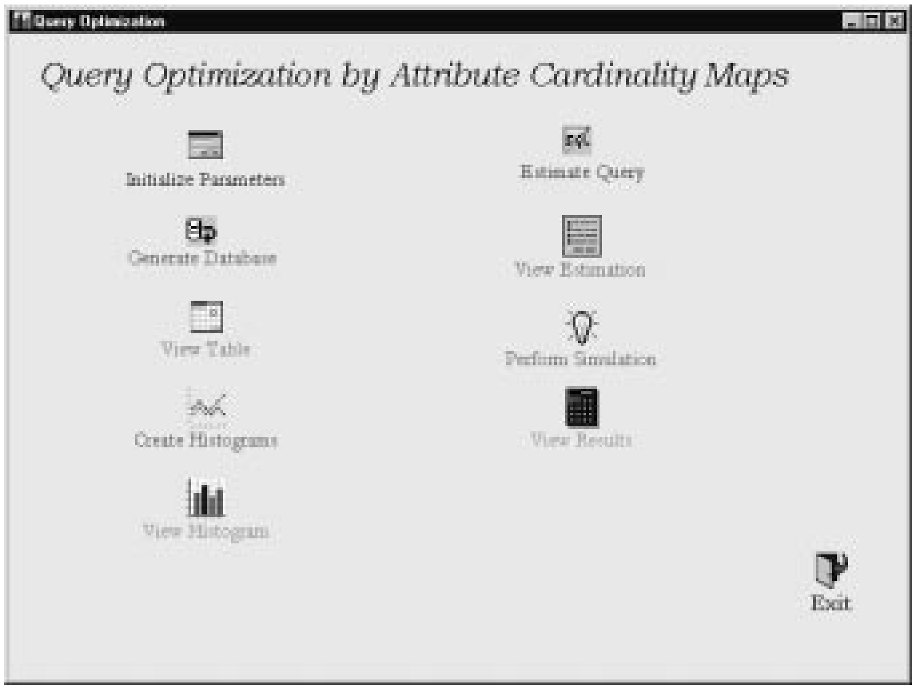


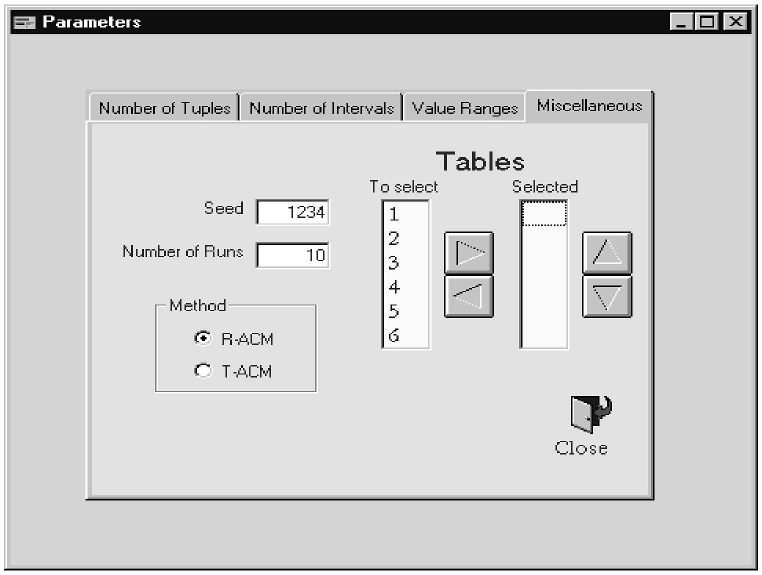
Figure 8: Main Screen of the Prototype.

- Seed: The seed is used to generate random data. The user can repeat the *same* experiment many times by introducing the same seed.
- Number of Runs: This is the number of times that a database is generated and the selected query is optimized.
- Method: The user can choose between the R-ACM and the T-ACM. The method chosen along with the Equi-width and the Equi-depth are then used to optimize the selected query.
- Query: There are two lists specifying the Tables involved in the joins. The tables chosen completely determine the query. The user can select some or all of them to define a query, because the tables selected and their order completely define the desired query.

Once the parameters are set, there are two possible ways by which the user can evaluate the efficiency of the selected method.

One of them can be done by choosing the icon *Generate Database*. The user can then visualize the data contained in each relation by selecting *View Table*. The next step consists of generating the histograms. This also provides another option, *View Histogram*, which allows the user to observe the Equi-width histogram, Equi-depth histogram and the selected method[8] histogram, by selecting any attribute of the database. A sample histogram for the Equi-width, the Equi-depth, and the R-ACM is depicted in Figure 10. Note the

[8]The system visualizes the histogram for the method that is selected in Parameters -> Miscellaneous, either for the R-ACM or the T-ACM.


Parameters

Number of Tuples | Number of Intervals | Value Ranges | Miscellaneous

Seed: 1234

Number of Runs: 10

Method: [✓ R-ACM] T-ACM

Tables

To select: 1, 2, 3, 4, 5, 6

Selected:

Figure 9: Parameters Screen: Miscellaneous. The list of tables selected completely specifies the query to be processed.

estimation accuracy obtained by the R-ACM compared to that of the Equi-width and the Equi-depth. The option *Estimate Query*, generates all the possible QEPs for the user-specified query, and calculates the estimated number of operations for each method as well as the number of operations required to perform the actual join. Finally, the user can visualize the results by selecting *View Estimation*.

The other way of evaluating the efficiency of the selected method is achieved by choosing *Perform Simulation*. The system generates as many databases as the number of runs specifies (which was specified as a parameter). The system also chooses the best QEP according to each method, and compares it with the best QEP obtained by performing the actual join for all the possible QEPs. By selecting the option *View Results*, the user can observe the number of times in which the R-ACM (or the T-ACM) is superior/inferior to the Equi-width and the Equi-depth schemes.

10 Conclusions

In this paper, we have discussed the efficiency of the traditional histogram-based estimation methods for Database Query Optimization, namely the Equi-width and the Equi-depth, as well as some new methods proposed in [24, 33] and [23, 33], the R-ACM and the T-ACM respectively. We have presented a prototype that was developed to benchmark histogram-based estimation methods.

We have shown analytically (using a reasonable model of accuracy, namely the doubly exponential distribution for errors) that as the accuracy of an estimation method increases, the possibilities of it leading to a superior QEP also increases. We have shown that histogram methods that produce errors with similar vari-

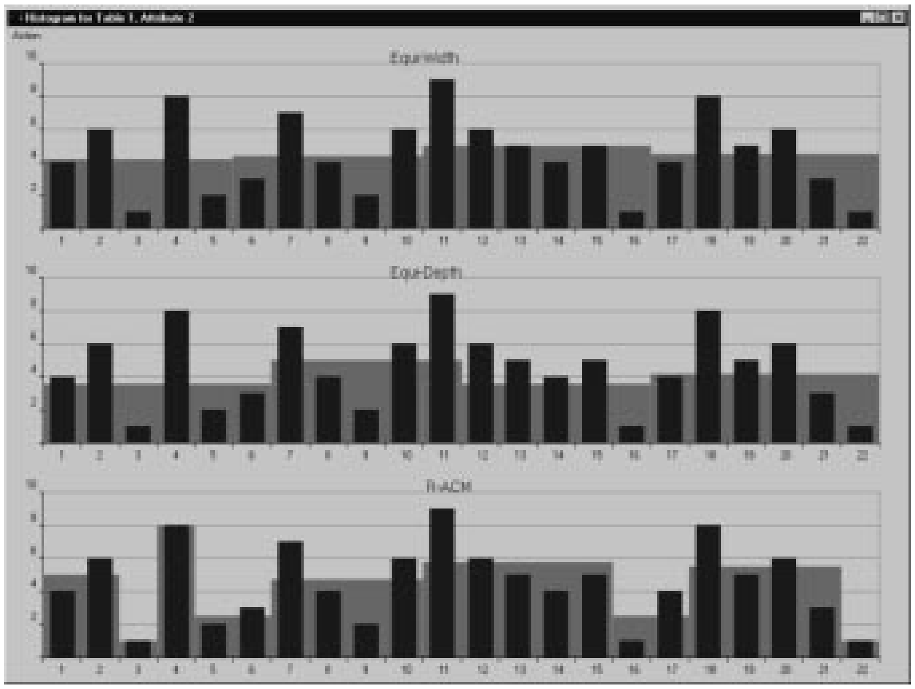


Figure 10: A Sample Histogram for Equi-width, Equi-depth, and R-ACM.

ances (such as the Equi-width *vs.* the Equi-depth, or the recently introduced the R-ACM *vs.* the T-ACM), the expected results are also similar. We have also shown that the R-ACM and the T-ACM, which produce error with significantly smaller variances than the traditional methods, yield better QEPs in a substantially larger number of times.

Due to the constraints involved in deriving a closed-form expression for integrating the normal probability density function, we have presented a computational analysis of the accuracy/optimality problem for the Gaussian distribution. Our analysis has also validated the result that histogram methods producing smaller errors lead more often to optimal QEPs.

We have also presented some experimental results obtained on randomly generated databases. The empirical results show that the R-ACM chooses superior QEPs in more than *twice* as many times as the Equi-width and in more than *four* times often than the Equi-depth. This ratio is more than *four* times when comparing the T-ACM with the Equi-width and the Equi-depth.

Although the current version of the comprehensive package does not include the implementation of the V-optimal scheme (and its enhancements) [14, 15, 28, 29, 30], and the wavelet-based histograms [16, 20], we plan to incorporate them in the next version.

The power of the new estimation methods, the R-ACM and the T-ACM, over the traditional methods, Equi-width and Equi-depth, has been clearly demonstrated both analytically and empirically.

References

[1] S. Christodoulakis. Estimating selectivities in data bases. In *Technical Report CSRG-136*, Computer Science Dept, University of Toronto, 1981.

[2] S. Christodoulakis. Estimating block transfers and join sizes. In *ACM SIGMOD 83, Proceedings of the Annual Meeting*, pages 40–54, San Jose, CA, 1983.

[3] S. Christodoulakis. Estimating record selectivities. In *Information Systems*, volume 8, 1983.

[4] S. Cluet and G. Moerkotte. On the complexity of generating optimal left-deep processing trees with cartesian products. In *Proceedings of the International Conference on Databases Theory*, pages 54–67, Prague, 1995.

[5] Microsoft Corporation. *Microsoft Visual J++ 6.0 - Programmer's Guide.* Microsoft Press, 1998.

[6] E. Deeba and A. Gunawardena. *Interactive Linear Algebra with MAPLE V.* Springer, 1997.

[7] R. Duda and P. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, Inc., 1973.

[8] S. Dunn. *Visual J++ 6.0 - Developer's Workshop.* Microsoft Press, 1998.

[9] C. Faloutsos, Y. Matias, and A. Silberschatz. Modeling skewed distributions using multifractals and the 80-20 law. In *Technical Report*, Dept. of Computer Science, University of Maryland, 1996.

[10] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1990.

[11] J. Goswami and A. Chan. *Fundamentals of Wavelets: Theory, Algorithms, and Applications.* Wiley Series-Microwave and Optical Engineering, 1999.

[12] T. Ibaraki and T. Kameda. On the optimal nesting order for computing n-relational joins. In *ACM Transactions on Database Systems*, volume 9, pages 482–502, September 1984.

[13] K. Ingalls and D. Jinguji. *Learn Microsoft Visual J++ 6.0 Now.* Microsoft Press, 1998.

[14] Y. Ioannidis and S. Christodoulakis. On the propagation of errors in the size of join results. In *Proceedings of the ACM-SIGMOD Conference*, pages 268–277, 1991.

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

[16] K. Chakrabarti and M. Garofalakis and R. Rastogi and K. Shim. Approximate Query Processing Using Wavelets. In *26th International Conference on Very Large Databases*, pages 111–122, Cairo, Egypt, September 2000.

[17] R. P. Kooi. *The optimization of queries in relational databases.* PhD thesis, Case Western Reserve University, 1980.

[18] R. Krishnamurthy, H. Boral, and C. Zaniolo. Optimization of nonrecursive queries. In *Proceedings of the 12th International Conference on Very Large Databases*, pages 128–137, Kyoto, 1986.

[19] M.V. Mannino, P. Chu, and T. Sager. Statistical profile estimation in database systems. In *ACM Computing Surveys*, volume 20, pages 192–221, 1988.

[20] Y. Matias, J. Vitter, and M. Wang. Wavelet-Bassed Histograms for Selectivity Estimation. In *ACM SIGMOD Conference on Management of Data*, pages 448–459, 1998.

[21] M. Muralikrishna and D. Dewitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. In *Proceedings of ACM-SIGMOD Conference*, pages 28–36, 1988.

[22] T. Nguyen and B. J. Oommen. Moment-Preserving Piecewise Linear Approximations of Signals and Images. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 19, pages 84–91, 1997.

[23] B. J. Oommen and M. Thiyagarajah. The Trapezoidal Attribute Cardinality Map: A New Histogram-like Technique for Query Optimization. *Submitted for Publication.*

[24] B. J. Oommen and M. Thiyagarajah. The Rectangular Attribute Cardinality Map: A New Histogram-like Technique for Query Optimization. In *International Database Engineering and Applications Symposium, IDEAS'99*, pages 3–15, Montreal, Canada, August 1999.

[25] B. J. Oommen and M. Thiyagarajah. The Rectangular Attribute Cardinality Map: A New Histogram-like Technique for Query Optimization. Technical Report TR-99-01, School of Computer Science, Carleton University, Ottawa, Canada, January 1999.

[26] B. J. Oommen and M. Thiyagarajah. The Trapezoidal Attribute Cardinality Map: A New Histogram-like Technique for Query Optimization. Technical Report TR-99-04, School of Computer Science, Carleton University, Ottawa, Canada, February 1999.

[27] G. Piatetsky-Shapiro and C. Connell. Accurate estimation of the number of tuples satisfying a condition. In *Proceedings of ACM-SIGMOD Conference*, pages 256–276, 1984.

[28] V. Poosala and Y. Ioannidis. Selectivity Estimation Without the Attribute Value Independence Assumption. In *23rd. International Conference on Very Large Databases*, pages 486–495, Athens, Greece, August 1997.

[29] V. Poosala, Y. Ioannidis, P. Haas, and E. Shekita. Improved Histograms for Selectivity Estimation of Range Queries. In *ACM SIGMOD Conference on Management of Data*, pages 294–305, 1996.

[30] W. Poosala. *Histogram Based Estimation Techniques in Databases.* PhD thesis, University of Wisconsin - Madison, 1997.

[31] W. Scheufele and G. Moerkotte. On the complexity of generating optimal plans with cross products. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 238–248, Tucson, Arizona, 1997.

[32] P. Selinger, M.M. Astrahanand D.D. Chamberlin, R.A. Lorie, and T.G. Price. Access Path Selection in a Relational Database Management System. In *Proceedings of ACM-SIGMOD Conference*, pages 23–34, 1979.

[33] M. Thiyagarajah. *Attribute Cardinality Maps: New Query Result Size Estimation Techniques for Database Systems.* PhD thesis, Carleton Univeristy, Canada, 1999.