Independence is Good: Dependency-Based Histogram Synopses for High-Dimensional Data

Amol Deshpande*
Univ. of California, Berkeley
amol@cs.berkeley.edu

Minos Garofalakis
Bell Laboratories
minos@bell-labs.com

Rajeev Rastogi
Bell Laboratories
rastogi@bell-labs.com

ABSTRACT

Approximating the joint data distribution of a multi-dimensional data set through a compact and accurate histogram synopsis is a fundamental problem arising in numerous practical scenarios, including query optimization and approximate query answering. Existing solutions either rely on simplistic independence assumptions or try to directly approximate the full joint data distribution over the complete set of attributes. Unfortunately, both approaches are doomed to fail for high-dimensional data sets with complex correlation patterns between attributes. In this paper, we propose a novel approach to histogram-based synopses that employs the solid foundation of statistical interaction models to explicitly identify and exploit the statistical characteristics of the data. Abstractly, our key idea is to break the synopsis into (1) a statistical interaction model that accurately captures significant correlation and independence patterns in data, and (2) a collection of histograms on low-dimensional marginals that, based on the model, can provide accurate approximations of the overall joint data distribution. Extensive experimental results with several real-life data sets verify the effectiveness of our approach. An important aspect of our general, model-based methodology is that it can be used to enhance the performance of other synopsis techniques that are based on data-space partitioning (e.g., wavelets) by providing an effective tool to deal with the "dimensionality curse".

1. INTRODUCTION

Capturing the joint data distribution of multi-dimensional data sets through compact and accurate *synopses* is a fundamental problem arising in a variety of practical scenarios, including *query optimization, query profiling,* and *approximate query answering.* Cost-based query optimizers employ such synopses to obtain accurate estimates of intermediate result sizes that are, in turn, needed to evaluate the quality of different execution plans [11, 18]. Similarly, query profilers and approximate query processors require compact data synopses in order to provide users with fast, useful feedback on their original query [3, 19]. Such query feedback (typically, in the form of an *approximate answer*) allows OLAP and data-mining users to identify the truly interesting regions of a data set and, thus, focus their explorations quickly and effectively, without consuming inordinate amounts of valuable system resources. Further, users can determine how well-posed/semantically-correct their query is, allowing them to make an informed decision on whether they would like to invest more time and resources to execute it to completion.

Histograms [13, 19] constitute a very general class of synopsis structures that offer several advantages, including (1) they are typically built off-line and stored in the DBMS catalog, thus incurring almost no run-time overhead, (2) they are *non-parametric*, i.e., they do not require the data to fit any particular probability distribution, and (3) for most real-world data distributions, there exist compact histograms that produce low-error approximations. As a consequence, histogram-based techniques for approximating *one-dimensional* data distributions have been extensively studied in the research literature [13, 19]. Further, one-dimensional histograms have now been adopted by several commercial database systems (e.g., DB2, Informix, Oracle, Microsoft, and Sybase) replacing the naive and rarely valid *uniformity assumption* [13].

Accurate approximations for *multi-dimensional* data distributions, pose a problem that is much harder than its well-understood one-dimensional counterpart. A simplistic approach, typically employed in the bulk of today's commercial systems, is to use one-dimensional histograms on the marginal distributions in conjunction with a *full-independence assumption,* which basically states that all data attributes are mutually uncorrelated and, thus, joint attribute distributions can be obtained as products of one-dimensional marginals. Unfortunately, experience with real-life data proves that the full-independence model is almost always invalid and can lead to gross approximation errors in practice [9, 18]. As a consequence, more recent work has proposed algorithms for building *multi-dimensional* histogram synopses that try to directly approximate the joint data distribution of a multi-attribute data set [11, 18]. These earlier results have demonstrated that reasonably simple construction procedures can give compact multi-dimensional histograms that outperform full-independence approximations by several orders of magnitude for low to medium data dimensionalities (e.g., 2 to 5 dimensions)[1].

It is a well-known fact, however, that histogram-based approaches suffer from the infamous *"curse of dimensionality"*; that is, histograms become problematic when trying to approximate the joint distributions of the high-dimensional data sets that are typical of modern decision-support applications. The reason is that, as the dimensionality of the data increases, both the *storage overhead* (i.e., number of buckets) and the *construction cost* of histograms that can achieve reasonable error rates increase in an explosive manner [3]. Consequently, as the dimensionality of the joint data distribution

*Work done while visiting Bell Laboratories.

[1] Due to space constraints, we omit a detailed discussion of related work; it can be found in the full version of this paper [7].

ACM SIGMOD 2001 May 21-24, Santa Barbara, California, USA

reaches or exceeds 6–7 dimensions the histogram approximation errors start becoming intolerably high [11].

Our Contributions. Existing work on multi-dimensional histogram synopses has essentially considered only the two extremes of a rich spectrum of possibilities, both of which are unrealistic and doomed to fail for the typically high-dimensional data sets of modern decision-support applications. The fact that the full-independence assumption is almost never satisfied in relational database practice and can lead to extremely poor approximations has been well documented (see, for example, [9, 18]). On the other hand, building multi-dimensional synopses on the full-dimensional space of a multi-attribute table implicitly assumes a "fully-correlated model" of the attribute space and is guaranteed to fail in high-dimensional spaces. The work in this paper is motivated by the observation that, in most practical application scenarios, such a "fully-correlated model" is also an unrealistic assumption. Typically, real-life data tables are characterized by complex correlation patterns, where a certain subset of correlated attributes A can be (unconditionally) independent of another attribute subset B (*partial independence*) or, alternatively, A can be (conditionally) independent of B given a third subset of attributes C (*conditional independence*). As a simple example, consider the attributes of an `Employee` relation. Even though it is natural for the `salary` attribute to be "strongly" correlated with the `age` attribute (with higher/lower salaries typically going to older/younger employees), there is no reason to believe that either of these attributes would be correlated with an employee's `height` or `weight` attributes (which are themselves correlated). Also, note that, although the `salary` and `age` attributes are pairwise correlated, the two may become largely independent given a third attribute like, for example, an employee's `YPE` (i.e., "years of practical experience"). Similar complex correlation patterns abound in real-life data.

Motivated by the above observations, in this paper, we propose DEPENDENCY-BASED (DB) histograms, a novel approach to building histogram synopses for high-dimensional data that uses the solid foundation of *statistical interaction models* [1, 4, 20] to explore the spectrum of possibilities between the existing "fully-independent" and "fully-correlated" approaches. Abstractly, our key technical idea is to break the synopsis of a high-dimensional data set into two basic components: (1) a statistical interaction model that explicitly identifies the (possibly) complex correlation and independence patterns in the data; and, (2) a collection of lower-dimensional histograms, built based on the model, that can be used to accurately approximate the overall joint data distribution. To the best of our knowledge, our work is the first to demonstrate the effectiveness and feasibility of statistical interaction models as a tool for dealing with the "curse of of dimensionality" in the construction of histogram synopses for high-dimensional data. The salient technical contributions of our work are summarized as follows.

• DB-**Histogram Construction.** Constructing accurate DB-histogram synopses raises the issue of (1) inferring a concise interaction model from the underlying data distribution, and (2) building an effective collection of histograms on the relevant marginals (dictated by the model). We propose using the broad class of *decomposable* interaction models [20] for DB histograms, since it offers several important advantages, including interpretability and closed-form model estimates. We discuss effective *forward-selection* procedures for building a concise decomposable model for the input data distribution. Our most efficient model-selection algorithm is, in fact, a novel contribution of our work that is likely to be of interest to the statistical community [8]. The construction of the marginal histograms dictated by the model discovered raises the important issue of intelligently allocating the storage budget available for the synopsis among the various marginals. We propose an optimal dynamic-programming algorithm for this problem as well as a cheaper greedy heuristic based on the concept of marginal gains that is, in fact, optimal when the histogram error functions follow a diminishing-returns law [10].

• DB-**Histogram Usage.** Efficiently estimating the selectivity of a range query predicate over (a subset of) the data attributes using DB histograms requires new techniques that effectively utilize the model structure in conjunction with the marginal histograms. We propose novel usage algorithms for our DB-histogram synopses that exploit the *junction tree* representation of the model to effectively minimize the number of histogram operations involved. An interesting side-effect of our work lies in the introduction of a new hierarchical representation (termed *split trees*) for the well-known class of MHIST histograms [18] that is considerably more space efficient than the one originally proposed by Poosala and Ioannidis. We present novel algorithms for performing the basic MHIST operations necessary for our selectivity-estimation procedure that work solely on our space-efficient, split-tree representation for both the input(s) and output of the operation.

• **Experimental Validation.** We have conducted an extensive experimental study with several real-life data sets to determine the effectiveness our methodology compared to earlier techniques, including sampling and full-dimensional MHIST histograms. Our results demonstrate that DB histograms (a) yield approximate answers of superior quality compared to existing approaches, and (b) have the ability to provide fairly accurate, concise synopses for real-life data with as many as 12 dimensions.

An important aspect of our general, dependency-based approach is that it can be extended to all data-reduction techniques that are based on data-space partitioning, such as *wavelets* [3]. Typically, all such techniques suffer from the dimensionality curse, which renders them ineffective in medium to high data dimensionalities. Building and maintaining a statistical interaction model can help identify significant attribute correlation patterns and, consequently, the interesting lower-dimensional subspaces that should be approximated independently in a DEPENDENCY-BASED synopsis. We believe that our interaction model-based methodology provides a viable and effective approach for dealing with data-dimensionality issues that opens interesting new avenues for innovative research in data reduction (e.g., incremental maintenance or approximate querying of DEPENDENCY-BASED synopses).

2. PROBLEM FORMULATION

2.1 Multi-Dimensional Histogram Synopses

Joint Data Distributions: Definitions and Notation. Consider a relational table R comprising n real- or integer-valued attributes $X_1, \ldots, X_n$. (The definitions and methodology can be extended to non-numerical attributes by first mapping their domain values into floating-point numbers.) The information in R can be accurately captured as an n-dimensional array (tensor), whose j^{th} dimension is indexed by the values of attribute X_j ($j = 1, \ldots, n$) and whose cells contain the count (or, *frequency*) of tuples in R having the corresponding combination of attribute values.

More formally, let $D_1, \ldots, D_n$ denote the *value domains* of attributes $X_1, \ldots, X_n$, respectively. Without loss of generality, we assume that each domain D_j is indexed by the set of integers $\{1, 2, \cdots, |D_j|\}$, where $|D_j|$ denotes the number of distinct elements in D_j. Given a combination of attribute values $(i_1, \ldots, i_n)$ ($1 \leq i_k \leq |D_j|$), the *joint frequency* $f(i_1, \ldots, i_n)$ of the com-

bination is exactly the number of tuples in the relation that contain the value i_k in attribute X_k, for all k. The n-dimensional, $|D_1| \times |D_2| \times \cdots \times |D_n|$ array with entries $f(i_1, \ldots, i_n)$ represents the *joint data (or, frequency) distribution* of $\{X_1, \ldots, X_n\}$ [18].

Often, we are interested in the joint distribution of only a subset of the attributes $S \subset \{X_1, \ldots, X_n\}$; this is the case, for example, when a query optimizer needs to estimate the selectivity of a range query with range selections specified only on attributes belonging to a subset S of $\{X_1, \ldots, X_n\}$. In probability terms, such scenarios require the *marginal data distribution* on attributes $S \subset \{X_1, \ldots, X_n\}$. Any such marginal can be obtained by projecting the joint data distribution array onto the relevant subset of attributes. Assuming, without loss of generality, that $S = \{X_1, \ldots, X_s\}$ ($s \leq n$), the marginal data distribution of S is defined as $f_S(i_1, \ldots, i_s) = \sum_{i_{s+1}, \cdots, i_n} f(i_1, \ldots, i_n)$ for all value combinations $(i_1, \ldots, i_s)$; that is, we compute the marginal frequencies over S (denoted by f_S) by aggregating the joint frequency counts over the domains of all "projected-away" attributes. We also use $E(f_S)$ to denote Shanon's *entropy measure* [20] for the joint frequency distribution over S. Letting $N = |R|$ denote the number of data tuples, the entropy $E(f_S)$ can be expressed as:

$$\begin{aligned} E(f_S) &= -\sum_{(i_1, \cdots, i_s)} \text{Prob}[(i_1, \cdots, i_s)] \cdot \log \text{Prob}[(i_1, \cdots, i_s)] \\ &= \log N - \frac{1}{N} \sum_{(i_1, \cdots, i_s)} f_S(i_1, \ldots, i_s) \cdot \log f_S(i_1, \ldots, i_s). \end{aligned}$$

Histogram Synopses. Building compact *synopses* structures that approximate a joint frequency distribution with reasonable accuracy in limited space is critical for numerous applications, including query optimization and profiling [19]. *Histogram-based synopses* for approximating *one-dimensional* data distributions have been extensively studied in the research literature [13, 19], and have been adopted by several commercial database systems. Briefly, a histogram on an attribute X is typically constructed by partitioning the frequency distribution of X into $\beta \geq 1$ *buckets* of consecutive attribute values and employing a *uniformity assumption* to approximate the frequencies and values present in each bucket. One-dimensional histograms can also be used to approximate (multi-dimensional) joint data distributions through the *full-independence assumption* for the attributes of interest. Given a collection of attributes $\{X_1, \ldots, X_n\}$, full independence essentially requires that all attributes have *mutually independent* frequency distributions. (Mathematical conditions for mutual independence can be found in any standard statistics textbook; see, for example [1].) Mutual independence, in turn, implies that any joint attribute distribution can be obtained as a product of the one-dimensional marginal distributions of the individual attributes; thus, for any $S = \{X_1, \ldots, X_s\}$, $f_S(i_1, \ldots, i_s) = \prod_{k=1}^{s} f_{X_k}(i_k)$. Unfortunately, experience with real-life data sets offers overwhelming evidence that the full-independence assumption is almost always invalid and can lead to gross approximation errors in practice [18].

Rather than relying on heuristic independence assumptions, *multi-dimensional histograms* [18] try to directly approximate the joint distribution of $\{X_1, \ldots, X_n\}$ by strategically partitioning the data space into n-dimensional buckets in a way that captures the variation in data frequencies and values. Similar to the one-dimensional case, uniformity assumptions are made to approximate the distribution of frequencies and values within each bucket [18]. Finding optimal histogram bucketizations is a hard optimization problem that is typically $\mathcal{NP}$-complete even for two dimensions [16]. Various greedy heuristics for multi-dimensional histogram construction have been proposed [11, 18] and shown to perform reasonably well for low to medium data dimensionalities (e.g., n=2–5). Like most techniques that rely on space partitioning, however, multi-dimensional histograms also fall victim to the "curse of dimensionality", which renders them ineffective above 6–7 dimensions [11].

2.2 Statistical Interaction Models: A Primer

For several decades, statisticians have looked into the problem of constructing accurate *interaction models* for multi-variate contingency tables (i.e., tables of counts), where the elements of an underlying population are classified according to a number of different categories (or, dimensions) [1, 6]. Traditionally, the goal of such statistical multi-variate analysis is to discover and understand "strong" association patterns in the data and find a structural model of variable interactions that accurately and concisely describes these data patterns.

Log-linear Models. *Log-linear models* comprise a broad class of statistical interaction models for contingency tables that has been extensively studied in the statistical literature. The foundations for the theory and methods of log-linear models were laid back in the early 1970s, culminating in a series of seminal books and monographs on the subject [1]. Consider an n-dimensional contingency table on n categories with entries $f(i_1, \ldots, i_n)$. (The direct correspondence between contingency tables and joint data distributions should be obvious; both just store the counts/frequencies of distinct value combinations for the data dimensions.) Briefly, the most general (or, *saturated*) log-linear model for the table expresses the logarithm of the n-dimensional cell counts $f(i_1, \ldots, i_n)$ as a summation of *effects* whose dimensionality ranges from 0 up to n. More formally, the saturated log-linear model for f is expressed as

$$\begin{aligned} \log f(i_1, \ldots, i_n) = u &+ \sum_j u_j(i_j) + \sum_{j \neq k} u_{j,k}(i_j, i_k) \\ &+ \ldots + u_{1,\ldots,n}(i_1, \ldots, i_n), \quad (1) \end{aligned}$$

where the u summands capture the interactions between the variables denoted by their subscripts with the terms in the parentheses indexing the specific position of that interaction (in the same order of variables). Thus, $u_{1,2}(j, k)$ denotes the interaction effect between values j and k of variables/dimensions 1 and 2 respectively. Abstractly, an s-dimensional interaction effect captures the deviations of the means of log-frequencies at that level of aggregation from the means at "coarser" aggregation levels. For example, the one-dimensional effects u_j (also known as *main effects*) represent the deviations of the means of log-frequencies along dimension j from the overall mean u. Note that linearity in the logarithms of the counts corresponds to *multiplicative* relationships between the actual frequencies, which means that log-linear models often have very nice interpretations in terms of the independence properties in the underlying set of variables.

The saturated log-linear model depicted in Equation (1) is the most general model for n dimensions, in the sense that it specifies enough interaction parameters to accurately capture *any* n-variate contingency table. (In that sense, the saturated model corresponds to a model of "fully-correlated" data dimensions.) Simpler log-linear models reduce the number of parameters by specifying certain interaction effects to be zero, which can typically be interpreted as a form of "independence" between the corresponding dimensions. As an example, consider the log-linear model for a 3-dimensional table over $\{X_1, X_2, X_3\}$ in which $u_{1,2}(i, j) = u_{1,3}(i, k) = u_{2,3}(j, k) = u_{1,2,3}(i, j, k) = 0$ for all i, j, k; that is, $\log f(i, j, k) = u + u_1(i) + u_2(j) + u_3(k)$. It is fairly simple to verify that, for data conforming to this model, the underlying frequencies $f(i, j, k)$ can be derived based on the one-dimensional

marginals as $\hat{f}(i,j,k) = f_{X_1}(i) \cdot f_{X_2}(j) \cdot f_{X_3}(k)/N^2$ for all i,j,k, where $N = \sum_{i,j,k} f(i,j,k)$ is the sum of all frequencies (i.e., the number of all data points) in the table. (Throughout the paper, we use $\hat{f}_{\mathcal{M}}()$ to denote *frequency estimates* based on model $\mathcal{M}$, often omitting the subscript when the model used is obvious.) Thus, our example log-linear model is exactly equivalent to the full-independence model for 3 dimensions. Intuitively, this is exactly what we would expect, given that this model sets all interaction effects between any two or three dimensions to zero.

In general, a log-linear model involves specifying certain interaction effects to vanish and letting the remaining interactions be arbitrary and unknown. The practice and theory of log-linear models typically revolves around a smaller class of models, termed *hierarchical models*, that are characterized by the following key property: *for any $S \subseteq \{1,\ldots,n\}$, if u_S is specified to vanish then, for any $T \supset S$, u_T is also specified to vanish.* A simple notation used for describing hierarchical log-linear models is to list their *maximal* variable interaction components (also known as the model *generators* [15]) in square brackets. For instance, the saturated 3-dimensional log-linear model can be denoted as $[123]$ and the full-independence model is simply $[1][2][3]$. More interesting correlation/independence patterns can also be captured with hierarchical models.

- *Partial Independence.* That is, two disjoint subsets of the data dimensions are (unconditionally) independent of each other. For example, the model $\mathcal{M} = [1][23]$ specifies that dimension 1 is independent of the dimension pair $\{2,3\}$. Frequency estimates for this model are derived directly from the one- and two-dimensional marginals for variables 1 and $\{2,3\}$ respectively: $\hat{f}_{\mathcal{M}}(i,j,k) = f_{X_1}(i) \cdot f_{\{X_2,X_3\}}(j,k)/N$, for all i,j,k (again, N denotes the frequency total).
- *Conditional Independence.* That is, two disjoint subsets of the data dimensions are independent *given* the value(s) for a third subset. For example, the model $\mathcal{M} = [12][13]$ states that dimensions 2 and 3 are independent *given* a value for dimension 1. Frequency estimates are again directly derived from the relevant two-dimensional marginals as: $\hat{f}_{\mathcal{M}}(i,j,k) = f_{\{X_1,X_2\}}(i,j) \cdot f_{\{X_1,X_3\}}(i,k)/f_{X_1}(i)$ for all i,j,k.

We should stress, however, that not all hierarchical log-linear models admit interpretations in terms of correlation/independence patterns among dimensions. A simple example is the model $[12][23][13]$ which, in fact, is the smallest non-interpretable hierarchical model. Besides not admitting a "statistical interpretation", an important practical problem with such models is that they also do not admit closed-form estimates based on marginal data distributions. Instead, frequency estimates must be obtained through an iterative numerical procedure known as *Iterative Proportional Fitting (IPF)* that, briefly, tries to discover a "maximum entropy" collection of frequencies that also satisfies the constraints on the marginals imposed by the specific model [1, 4].

Decomposable Models: Chordal Graphs and Junction Trees. As a consequence of these problems with general log-linear models, statisticians often focus on restricted subclasses of models that are easier to interpret and/or use. An important such subclass is the class of *decomposable models*. Abstractly, the class of decomposable models comprises the subset of log-linear models that satisfy the following two important properties.

1. *Interpretability:* Decomposable models can be described in terms of correlation/independence relationships between the relevant variables. Furthermore, decomposable models can always be represented graphically as a *Markov network* [20] from which the underlying correlation/independence relationships can be directly inferred.
2. *Direct frequency estimates:* The frequency estimates for a decomposable model can always be obtained directly from the marginals corresponding to the model generators as a closed, "product-form" expression. That is, decomposable models obviate the need for iterative numerical techniques like IPF.

Our example full independence ($[1][2][3]$), partial independence ($[1][23]$), and conditional independence ($[12][13]$) models are decomposable, with the closed-form frequency estimates $\hat{f}(i,j,k)$ as defined above for each individual case. Figure 1(a) gives the Markov networks corresponding to these three models, where the nodes correspond to the model variables and edges represent the existence of an interaction between variables. Note that the generators of the model correspond directly to the *cliques* (i.e., maximal complete subgraphs) of its Markov network representation. The key probabilistic property of a Markov network (known as the *global Markov property* [20]) is that if two node sets A and B are *separated* by a third node set C, then A and B are conditionally independent given C. (For a more thorough introduction to the theory and practice of Markov networks, the interested reader is referred to [20].) It is easy to see that the number and complexity of possible decomposable models grows explosively with the number of variables [6, 15, 17]. Figure 1(b) shows a somewhat more complex example of a 5-dimensional decomposable model $\mathcal{M}$, namely $\mathcal{M} = [123][124][15]$. From the Markov graph, it is easy to read, for example, that variables $\{3,4\}$ are conditionally independent given variables $\{1,2\}$ and, similarly, that variable 5 is independent of $\{2,3,4\}$ given variable 1. The frequency estimates for $\mathcal{M}$ can be simply derived as $\hat{f}_{\mathcal{M}}(i,j,k,l,m) = f_{\{X_1,X_2,X_3\}}(i,j,k) \cdot f_{\{X_1,X_2,X_4\}}(i,j,l) \cdot f_{\{X_1,X_5\}}(i,m)/[f_{X_1}(i) \cdot f_{\{X_1,X_2\}}(i,j)]$. Our work relies heavily on the graph-theoretic properties of the Markov network representation of a model $\mathcal{M}$; therefore, we use $\mathcal{M}$ to refer to both the interaction model and its Markov graph in the remainder of this paper.

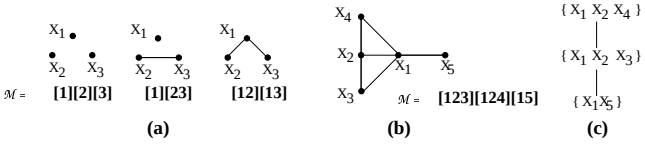


Figure 1: Markov networks for (a) three simple 3-dimensional decomposable models, and (b) a more complex 5-dimensional decomposable model. (c) A junction tree for the 5-dimensional example model.

An interesting property of decomposable interaction models is that they correspond exactly to the class of *chordal* (or, *triangulated*) Markov network graphs. A chordal graph has the property that any cycle of length four or more has a shortcut (*chord*) between any two non-consecutive nodes. Thus, for example, the model $[12][23][34][14]$ is not decomposable, since it corresponds to a non-chordal Markov graph (a 4-cycle).

A compact and particularly useful representation of chordal graphs is provided by *junction trees* (also known as *clique trees*) [2, 14]. Briefly, given a chordal graph $\mathcal{M}$, a junction tree $J(\mathcal{M})$ is a tree structure defined over the cliques (i.e., generators) of $\mathcal{M}$ characterized by the following *clique-intersection property*: "For each pair C_i and C_j of cliques in $\mathcal{M}$, the set $C_i \cap C_j$ is contained in every clique on the path connecting C_i and C_j in $J(\mathcal{M})$." Given

a connected graph, a junction tree exists if and only if the graph is chordal [2]; thus, a junction tree offers a complete characterization of a chordal graph and the corresponding decomposable model. Figure 1(c) depicts a junction tree for the model $\mathcal{M} = [123][124][15]$. An important property of the junction-tree representation for a decomposable model $\mathcal{M}$ that the frequency estimates for $\mathcal{M}$ are always equal to the ratio of the product of the marginal frequencies on all the model cliques divided by the product of the marginal frequencies on the intersections of cliques that are immediately adjacent in the junction tree [15]. As an example, it is simple to verify that the junction tree shown in Figure 1(c) directly gives the formula mentioned earlier in this section for $\hat{f}_{\mathcal{M}}$.

2.3 Problem Statement

Given the aforementioned problems with the full-independence assumption and conventional multi-dimensional histograms, we propose a novel approach that employs the solid foundation of interaction models to explicitly identify and exploit statistical patterns in the data during the construction and usage of histogram synopses. Our key technical idea lies in breaking the synopsis into (1) a statistical interaction model that accurately captures the independence/correlation patterns in the data, and (2) a collection of (low-dimensional) histograms on marginals that, based on the model, can approximate the overall joint data distribution. More formally, in this paper, we define the notion of a DEPENDENCY-BASED histogram synopsis as follows.

DEFINITION 2.1. *A* DEPENDENCY-BASED *(*DB*) histogram for a collection of attributes $\{X_1, \ldots, X_n\}$ is defined as a pair $\mathcal{H} = < \mathcal{M}, \mathcal{C} >$, where: (1) $\mathcal{M}$ is an accurate* decomposable model *for the joint data distribution of $\{X_1, \ldots, X_n\}$; and, (2) $\mathcal{C}$ is a collection of (low-dimensional) histograms that approximate the marginal distributions for each generator (i.e., clique) of $\mathcal{M}$. (We refer to these histograms as the* clique histograms *of $\mathcal{H}$.)* ∎

Decomposable vs. General Interaction Models. Definition 2.1 restricts the class of statistical interaction models for DEPENDENCY-BASED histograms to that of decomposable models. There are several reasons for this. First, even though the space of decomposable models is smaller that that of general hierarchical log-linear models, it is still large enough to include a multitude of models that capture complex (interpretable) interaction patterns among the data dimensions. For example, in less than or equal to 5 dimensions, there are $7,580$ hierarchical and $1,233$ decomposable models [6]. (Also note that, for any model, there always exists a more general model that is decomposable, since the saturated model is trivially decomposable.) Second, as already mentioned above, decomposable models have the benefits of *interpretability* and *closed-form frequency estimates*. Models that are interpretable in terms of partial and conditional independence can provide useful insights into the intrinsic properties and correlations in the data, even for purposes other than synopsis construction (e.g., data mining). Closed-form model estimates are obviously desirable, since they avoid expensive iterative procedures like IPF. IPF-based estimation is especially bad when estimates on certain marginals are required, since the only way to estimate the marginals is to go through the full joint distribution (e.g., given the model $[12][23][34][45][15]$, computing the marginal on $\{X_1, X_4\}$ requires IPF to rebuild the full 5-dimensional frequency distribution). This is not the case for decomposable models where, as we will see, such marginal computations can be carefully optimized based on the model structure (Section 3.3). Finally, besides the obvious computational benefits in their *usage* (due to the closed-form estimates), decomposable models have properties that can also substantially reduce the computational effort involved in their *construction* from the joint distribution data. These properties are thoroughly exploited in our model-selection algorithms (Section 3.1).

Quantifying Model Accuracy: "Goodness-of-fit" Testing. Our goal is to discover interaction models that accurately capture the correlation patterns in the joint data distribution. Thus, we need a measure that quantifies the "distance" of the set frequency estimates based on a given interaction model $\mathcal{M}$ to the true set of frequencies in the joint distribution. Following standard statistical practice [4, 6, 15], we employ the usual distance measure for probability distributions, namely the *Kullback-Leibler information divergence* (also known as "discrimination information" or "cross-entropy"), defined as

$$D(f, \hat{f}_{\mathcal{M}}) = \frac{1}{N} \cdot \sum_{(i_1, \cdots, i_n)} f(i_1, \ldots, i_n) \cdot \log \frac{f(i_1, \ldots, i_n)}{\hat{f}_{\mathcal{M}}(i_1, \ldots, i_n)}.$$

The information divergence $D(f, \hat{f}_{\mathcal{M}})$ can essentially be interpreted as the difference of the information contained in f and that contained in $\hat{f}_{\mathcal{M}}$ about f. The quantity $D(f, \hat{f}_{\mathcal{M}})$ is always non-negative and vanishes if and only if $\hat{f}_{\mathcal{M}}$ coincides with f; that is, if and only if the model $\mathcal{M}$ *exactly* captures the joint data distribution. Furthermore, information divergence is equivalent to the *likelihood ratio* test statistic for the null hypothesis H_0: "Model $\mathcal{M}$ generated f" [4], and is directly correlated to the *chi-square (χ^2)* distance of the two distributions; in fact, it is well known that $D(f, \hat{f}_{\mathcal{M}}) \approx \frac{1}{2}\chi^2(f, \hat{f}_{\mathcal{M}})$, so that minimizing divergence is equivalent to minimizing χ^2 distance [15].

Problem Statement: DEPENDENCY-BASED **Histogram Construction and Usage.** Motivated from the inadequacies of earlier solutions and the success of statistical interaction models in the analysis of contingency tables, this paper proposes a novel approach to building multi-dimensional data synopses grounded on the concept of DB histograms. More formally, the problem we attack in this paper can be stated as follows.

• **Given:** A relational table R on n attributes $\{X_1, \ldots, X_n\}$ and a space budget B for constructing a synopsis on R.

• **Need:** Efficient algorithms for: (1) building an *accurate* and *concise* DB histogram $\mathcal{H} =< \mathcal{M}, \mathcal{C} >$ on R that uses at most B units of space; and, (2) effectively utilizing $\mathcal{H}$ for providing quick approximate answers to selectivity-estimation queries over the joint data distribution and any one of its marginals.

The *accuracy* of a DB histogram $\mathcal{H} =< \mathcal{M}, \mathcal{C} >$ obviously depends on the accuracy of each of its components. Our algorithms quantify the accuracy of the interaction model $\mathcal{M}$ using the standard Kullback-Leibler information divergence measure, as described above. The accuracy of the clique histograms in $\mathcal{C}$ can be quantified using any of the standard quality measures for histogram synopses (e.g., total variance) [18, 19].

The *conciseness* of the interaction model $\mathcal{M}$ is another important practical requirement for our DB histogram synopsis. The observation here is that a more complex model will always fit the given data better; for example, the saturated model can always provide an exact fit but is not always the correct interaction model for our purposes. Essentially, we are interested in the most parsimonious model that captures all the *significant* correlations and independencies in the data, so that these patterns are explicitly accounted for during the construction of the synopsis. Our DB-histogram construction algorithms use two distinct strategies to control the conciseness of the induced model $\mathcal{M}$.

• *Enforce dimensionality bound on marginals.* The idea here is to impose an upper bound k_{max} on the dimensionality of each gener-

ator in $\mathcal{M}$. This bound limits the complexity of the model by explicitly restricting the dimensionality of each marginal distribution that we need to approximate through a clique histogram in $\mathcal{H}$. Such a bound can be set heuristically, based on pragmatic constraints on the effectiveness of histogram approximations; for example, since it is well-known that the accuracy of histogram-based synopses drops rapidly above 3–4 dimensions [11], a value of $k_{max} = 3$ may be a reasonable choice for restricting model complexity while not compromising the accuracy of the DB histogram. (A more principled approach would be to set the value of k_{max} based on intrinsic characteristics of the data distribution, such as its *fractal dimension* [9] that, in a sense, measures its "distance" from full independence; we intend to explore such methods in future work.)

• *Enforce statistical significance thresholds.* This method is based on the use of *significance levels* to enforce a lower bound on the statistical significance at which additional complexity is introduced into the model. Abstractly, what this means is that when deciding between a "simple" model $\mathcal{M}_1$ and a more complex model $\mathcal{M}_2$, $\mathcal{M}_2$ is preferred only if the difference in the quality of the estimates (i.e., the "goodness-of-fit") is sufficiently high to be statistically significant. The goal, of course, is to ensure that the selected model captures only the truly significant patterns in the data while ignoring details that can be attributed to statistical noise (i.e., avoiding "overfitting" the data).

3. DEPENDENCY-BASED HISTOGRAMS

3.1 Model Selection

Apart from certain special cases, the general problem of inferring an "optimal" statistical model (decomposable or otherwise) for a given data set is a very hard search problem that can be solved exactly only by exhaustive search [17]. Thus, statisticians typically resort to computationally-efficient heuristic search strategies that, although suboptimal, often perform well in practice. The statistical literature offers two broad greedy-search paradigms that form the basis of these heuristics, namely *forward selection* and *backward elimination* [4]. Briefly, forward selection starts out with a very simple model (e.g., full independence) and incrementally adds more complexity by including in the model, at each step, the variable interaction(s) that result in the highest increase in model accuracy. Backward elimination, on the other hand, begins with the most complex, saturated interaction model (corresponding to a complete Markov graph) and iteratively reduces its complexity by removing, at each step, the variable interaction(s) resulting in the smallest decrease in model accuracy.

When restricting the search space of the model-selection process to *decomposable* models (as we do in this work), special care needs to be taken to ensure that only such models are considered by the search strategy. Backward-elimination strategies for decomposable models are well established in the statistical literature. Unfortunately, such backward-elimination schemes have the disadvantage that they typically require evaluating a large number of intermediate models, especially when the number of attributes (i.e., the dimensionality of the data) involved is high [15]. Backward-elimination strategies are, in fact, particularly bad for building DB histograms since, as already mentioned above, histogram synopses are only effective for relatively low dimensionalities, implying that most of the interaction edges in the complete model graph may need to be checked and removed.

Based on the above discussion, we have chosen to employ a forward-selection strategy for our DB-histogram construction algorithms. Unfortunately, efficient forward selection schemes for decomposable models are not as well understood or researched in the statistical literature [8, 15]. In our work, we propose two distinct forward-selection algorithms for building decomposable models for DB histograms. Our first algorithm is based on a "naive" strategy that operates by introducing arbitrary edges (based on their benefit) and running repeated chordality tests to ensure the decomposability of the model. Our second, more efficient algorithm is based on a novel edge-selection strategy that guarantees that only edges that retain the decomposability property are introduced to the current model in, essentially, *constant time-per-edge*. Note that this forward-selection scheme is a novel contribution of our work that is very likely to be of interest to the statistical-modeling community [8]. Due to space constraints and to keep the discussion more focused on the histograming issues, the detailed development of our model-selection algorithms for DB histograms can be found in the full version of this paper [7]. In the full paper, we also discuss certain important properties of decomposable interaction models and describe how our algorithms exploit them to optimize the computational effort (i.e., number of entropy calculations) involved in the forward-selection process [7].

3.2 Building the Clique-Histogram Collection

The model-selection process provides us with a concise decomposable interaction model $\mathcal{M}$ that accurately captures the correlations in the data. The next step in building the DB-histogram synopsis is the construction of the *clique-histogram* collection $\mathcal{C}$. As their name reveals, our clique histograms are built on the marginal frequency distributions defined by the cliques/generators of $\mathcal{M}$, which, by the model-selection process, are guaranteed to capture the most significant correlation patterns in the data. Histogram synopses for the clique marginals can then be used, in conjunction with the decomposable model, to effectively approximate the overall joint data distribution f as well as *any* other marginal, by taking advantage of the closed-form model estimates and the junction tree representation of the model (Section 3.3).

Two key issues arise in the construction of the clique histograms. First, we need to decide on the *type* of (multidimensional) histograms that will be used to capture the clique marginals. Second, given that our end goal is to maximize some overall accuracy measure in the DB-histogram approximation within a specified space budget of B, we have to devise methods for intelligently *allocating* the B storage units among the different clique histograms being built. We elaborate further on these two issues in what follows.

Clique-Histogram Types. The main goal of our work is to demonstrate the importance and viability of statistical interaction modeling techniques for building high-dimensional data synopses and not to introduce a new family of histogram structures and construction algorithms. Thus, our DB-histogram construction algorithms use variants of known histograming techniques to capture the clique marginal distributions for a given decomposable model $\mathcal{M}$.

• **MHIST Histograms** are a straightforward adaptation of the state-of-the-art MHIST-2 technique proposed by Poosala and Ioannidis [18]. Abstractly, the key idea behind MHIST-2 is to produce a hierarchical partitioning of the data space by recursively partitioning the bucket that is *in most need of partitioning* until the space budget (i.e., number of available buckets) is exhausted.

• **Grid Histograms** are based on a simple generalization of $p \times q$ rectangular array partitionings (e.g., [16]) to higher dimensionalities. We use a simple greedy algorithm for building grid histograms that, at each step, partitions the entire data distribution along the dimension that is in most need of partitioning while making sure that the allotted number of buckets is not exceeded.

For both histogram types, the "need of partitioning" is deter-

mined based on the histogram *partitioning constraint*, e.g., *V-optimal* or *MaxDiff* [19]. As an example, a *MaxDiff* constraint means that the construction algorithm always splits (the bucket or the data distribution) along the dimension with the largest difference in frequency between two adjacent values by placing a bucket boundary between these values [18].

Space Allocation. Given the limited storage budget available for our DB-histogram synopsis, it is crucial to allocate space to the distinct clique histograms in $\mathcal{C}$ in an intelligent manner. Typically, the end goal is to minimize an overall error of the histogram approximation, such as the normalized mean square error or the total variance over all histogram buckets [19]. The problem is somewhat complicated by the fact that the model cliques can be of different arities and, as a consequence, the space required for storing a bucket can vary among the different histograms. More formally, let $C_1, \ldots, C_{|\mathcal{C}|}$ denote the different cliques in $\mathcal{M}$ and let β_i, s_i denote the number of buckets allotted to the histogram for clique C_i and the space requirements of each such bucket, respectively. Also, given an algorithm for building the i^{th} clique histogram, let $\mathrm{ERR}_i(\beta_i)$ denote the overall error of that histogram in approximating the marginal distribution over C_i when exactly β_i buckets are used. Our space-allocation problem for the clique-histogram collection can now be stated as follows:

$$\text{Minimize} \quad \sum_{i=1}^{|\mathcal{C}|} \mathrm{ERR}_i(\beta_i) \quad , \quad \text{subject to} \quad \sum_{i=1}^{|\mathcal{C}|} \beta_i s_i \leq B.$$

This is essentially a discrete resource allocation problem with a *separable* objective function [12], which can be solved *optimally* in pseudo-polynomial time using dynamic programming. More specifically, let $F^{(b)}(s)$ denote the minimum achievable total error for the first c clique histograms when at most s units of storage are used; that is,

$$F^{(c)}(b) = \min_{(\beta_1,\ldots,\beta_c)} \{ \sum_{i=1}^{c} \mathrm{ERR}_i(\beta_i) \ : \ \sum_{i=1}^{c} \beta_i s_i \leq b \}.$$

Then, obviously, $F^{(|\mathcal{C}|)}{}_{(B)}$ gives the optimal objective value of our clique-histogram space-allocation problem. The computation of the optimal space allotment is done based on the following dynamic-programming formulation:

$$F^{(c)}(b) = \min_{l=0,\ldots,b} \{ F^{(c-1)}(b-l) + \mathrm{ERR}_c(\left\lfloor \frac{b}{s_c} \right\rfloor) \}$$

with the boundary conditions $F^{(1)}(b) = \mathrm{ERR}_1(\left\lfloor \frac{b}{s_1} \right\rfloor)$ for all $b = 0, \ldots, B$. The running-time complexity of our dynamic-programming solution is $O(|\mathcal{C}|B^2)$, assuming the error-function values ERR_i for individual histograms have been precomputed. Note that for grid histograms each split introduced by the construction algorithm can result in a multitude of new buckets; thus, the ERR_i functions for grid will have a "piecewise constant" form with the error dropping only when the extra buckets are sufficient for a new split. MHIST histograms, on the other hand, have smoother error curves, since each split introduces only one new bucket. We defer presentation of the full details of our dynamic-programming algorithm for space allocation to the full paper.

A simpler and more efficient heuristic algorithm for our space-allocation problem (termed **IncrementalGains**) is depicted in Figure 2. The key idea in **IncrementalGains** is to incrementally distribute the space budget among the clique histograms based on *marginal gains* [10]. Abstractly, the **IncrementalGains** algorithm works in parallel with the histogram construction. At each step, the next split for each clique histogram (as dictated by the construction algorithm) is evaluated in terms of (a) the improvement that the extra split brings in terms of the overall error in the histogram approximation, and (b) the increase in the amount of space required for storing the new bucket(s) introduced by the split. Among all candidate splits, **IncrementalGains** selects the split that maximizes the decrease in error per unit of required bucket space and, of course, does not violate our storage-space constraint. The running-time complexity of the **IncrementalGains** algorithm is only $O(|\mathcal{C}| + B \log |\mathcal{C}|)$. Furthermore, it is a well-known result (e.g., [10]) that incremental allocation based on marginal gains is, in fact, *optimal* if the components of the separable objective function (i.e., the ERR_i's) are *convex*; that is, when the histogram error functions follow a law of *diminishing returns* with respect to allotted space. Such an assumption may often be satisfied during histogram construction and, therefore, we expect that **IncrementalGains** will typically perform well in practice.

procedure IncrementalGains($\{ C_i, \mathrm{ERR}_i, s_i \}$ $(i = 1, \ldots, |\mathcal{C}|)$,B)
Input: Attribute sets $\{C_1, \ldots, C_{|\mathcal{C}|}\}$) corresponding to model cliques; total storage space budget B; error measure $\mathrm{ERR}_i()$ and per bucket storage requirement s_i for the i^{th} clique histogram.
Output: Feasible bucket allocation $< \beta_1, \ldots, \beta_{|\mathcal{C}|} >$ that minimizes the overall histogram approximation error.
begin
1. /* all histograms start as one bucket containing the average frequency */
2. $< \beta_1, \ldots, \beta_{|\mathcal{C}|} >=< 1, \ldots, 1 >$, used := $\sum_i s_i$
3. foundOne := **true**
4. **while** (foundOne = **true**) **do**
5. foundOne := **false**
6. let n_i be the extra buckets required for adding a new split (as dictated by the construction algorithm) to the histogram for clique C_i, $i = 1, \ldots, |\mathcal{C}|$
7. let $\Delta\mathrm{ERR}_i := \mathrm{ERR}_i(\beta_i + n_i) - \mathrm{ERR}_i(\beta_i)$, $i = 1, \ldots, |\mathcal{C}|$
8. sort all candidate splits in the order $< i_1, \ldots, i_{|\mathcal{C}|} >$ such that for each j: $\frac{\Delta\mathrm{ERR}_{i_j}}{n_{i_j} \cdot s_{i_j}} \geq \frac{\Delta\mathrm{ERR}_{i_{j+1}}}{n_{i_{j+1}} \cdot s_{i_{j+1}}}$
9. $l := 1$
10. **while** (used $+ n_{i_l} \cdot s_{i_l} > B$) **do** $l := l + 1$
11. **if** ($l \leq n$) **then**
12. add a new split to the histogram for clique C_{i_l}
13. $\beta_{i_l} := \beta_{i_l} + n_{i_l}$, used := used - $n_{i_l} \cdot s_{i_l}$
14. foundOne := **true**
15. **end**
16. **end**
end

Figure 2: The IncrementalGains space-allocation algorithm.

3.3 Using DEPENDENCY-BASED Histograms

Efficiently estimating the selectivity of a range-query predicate over (a subset of) the n data attributes using DB histograms requires novel usage techniques that effectively utilize the model structure in conjunction with the clique histograms. In this section, we propose algorithms for estimating query-predicate selectivities using DB histograms. Let $\mathcal{H} =< \mathcal{M}, \mathcal{C} >$ denote a DB histogram on attributes $\{X_1, \ldots, X_n\}$ and let P be a range-selection predicate defined, without loss of generality, over the attributes $X_1, \ldots, X_m$, where $m \leq n$; that is, P specifies no ranges over $X_{m+1}, \ldots, X_n$. Also, let $C_1, \ldots, C_{|\mathcal{C}|}$ denote the cliques/generators in the constructed decomposable model $\mathcal{M}$ and let $\mathcal{C} = \{H(C_1), \ldots, H(C_{|\mathcal{C}|})\}$, where $H(C_i)$ is the histogram built for clique C_i $(i = 1, \ldots, |\mathcal{C}|)$.

Estimating the selectivity of P is accomplished by using $\mathcal{H}$ to compute the (approximate) marginal frequency distribution over the specified attributes $X_1, \ldots, X_m$. We demonstrate how this computation can be carried out efficiently by exploiting the *junc-*

tion tree representation of our decomposable model $\mathcal{M}$ (Section 2.2). Given the "product form" of the selectivity estimates dictated by $\mathcal{M}$, we also need effective algorithms for computing the *product* and *projection* (i.e., "marginalization") of our histogram-based approximations to the model's clique marginals. We propose algorithms for these histogram operations in the second part of this section. (Our DB-histogram implementation also incorporates several practical optimizations; a discussion can be found in the full paper.)

3.3.1 Computing Arbitrary Marginal Distributions

As we already pointed out in Section 2, one of the main advantages of using a decomposable interaction model $\mathcal{M}$ in $\mathcal{H}$ is that it allows for direct, closed-form estimates for the joint data distribution f that we want to approximate. These estimates have a product form that can be directly read off the *junction tree* representation of $\mathcal{M}$ [2, 14], denoted by $J(\mathcal{M})$. Recall from Section 2.2 that $J(\mathcal{M})$ is a tree over the cliques of $\mathcal{M}$ that satisfies the *clique-intersection property*; that is, for each pair C_i and C_k of cliques, the set $C_i \cap C_k$ is contained in every clique on the path connecting C_i and C_k in $J(\mathcal{M})$. Another important property of the junction tree structure is the *separation property*: "For each edge (C_i, C_j) in $J(\mathcal{M})$ the set $S_{ij} = C_1 \cap C_2$ separates $C_i - S_{ij}$ and $C_j - S_{ij}$ in $\mathcal{M}$." Based on this property and the interpretation of separation in terms of conditional independence in decomposable models, given the edge (C_1, C_2) in $J(\mathcal{M})$ we can estimate the joint frequency distribution of $C_1 \cup C_2$ using the formula $f_{C_1 \cup C_2} = f_{C_1} \cdot f_{C_2} / f_{C_1 \cap C_2}$. Similarly, the form of the estimates for the overall joint frequency distribution can be read off $J(\mathcal{M})$ based on the following formula [15]:

$$\hat{f} = \frac{\prod_{i=1}^{|\mathcal{C}|} f_{C_i}}{\prod_{(C_i,C_j)\in J(\mathcal{M})} f_{C_i \cap C_j}}. \qquad (2)$$

The above result directly provides us with a naive method for computing the marginal frequency distribution over $X_1, \ldots, X_m$ and, therefore, estimating the selectivity of P: simply build the junction tree $J(\mathcal{M})$, then use Equation (2) to reconstruct the full joint frequency distribution $\hat{f}$ based on the clique histograms and, finally, project $\hat{f}$ onto the attributes of interest $X_1, \ldots, X_m$. (Note that, given a chordal graph $G = (V, E)$, a junction tree representation of G can be computed very quickly in $O(|V|+|E|)$ time [14].) However, computing the full frequency distribution is obviously an overkill since, in most practical scenarios, the range predicate P specifies selections over only a small subset of attributes (i.e., $m << n$). We now propose a more efficient algorithm that makes more effective use of the junction tree representation to minimize the computation required for obtaining arbitrary marginals.

Our algorithm views the junction tree $J(\mathcal{M})$ as a rooted tree with an arbitrarily-chosen root node (clique). Also, with each node C_i in $J(\mathcal{M})$ we associate the union of the cliques corresponding to all the descendents of the node in $J(\mathcal{M})$, including itself. This union, denoted by `cover`(C_i), can obviously be computed in a single bottom-up traversal of the tree. The complete outline of our recursive marginal-computation algorithm (termed **ComputeMarginal**) is shown in Figure 3. The input arguments of **ComputeMarginal** are (1) a node C_i of the (rooted) junction tree $J(\mathcal{M})$, and (2) a set of attributes S_Q for which a histogram of their joint frequency distribution is required from the (sub)tree rooted at C_i. Our description uses the functions `project` and `product`; briefly, `project`($H(C)$, S) returns the projection of histogram $H(C)$ over C onto the subset of attributes $S \subseteq C$, whereas `product`($H(C_i)$, $H(C_j)$) returns the histogram $H(C_i \cup C_j)$ over $C_i \cup C_j$ that results from multiplying $H(C_i)$ and $H(C_j)$ using the separation formula for obtaining frequencies, i.e., $f_{C_i \cup C_j} = f_{C_i} \cdot f_{C_j} / f_{C_i \cap C_j}$. (The relevant algorithms are described in Section 3.3.2.) Briefly, **ComputeMarginal** first checks whether the clique C_i is a superset of the specified attributes S_Q since, in that case, we can get the histogram for S_Q by simply projecting out the unnecessary attributes from $H(C_i)$ (Step 1). If C_i cannot compute the required frequency distribution by itself then we keep the portion of S_Q whose distribution can be determined from C_i ($C_i \cap S_Q$) and recursively use the children of C_i to compute the relevant marginals for the remaining attributes ($S_Q - C_i$). In the simpler case (Steps 4–10), there exists a single child C_j of C_i that covers all remaining attributes. Then , we can then simply pass all remaining attributes to that child but, of course, we also have to ensure that we augment this set with all attributes in the separator $S_{ij} = C_i \cap C_j$, so that the resulting frequency distributions can be properly multiplied out based on the tree's separation property (Steps 7–9). In the more complex case (Steps 11–19), none of the children can compute the distribution of $S_Q - C_i$ by itself, so we must break up $S_Q - C_i$ into multiple parts such that each part can be handled by a single child node. The resulting marginals are again multiplied out based on the separation property to obtain the overall frequency estimates for S_Q (Steps 14–16). The initial invocation of algorithm **ComputeMarginal** sets C_i equal to the root node of $J(\mathcal{M})$ and $S_Q = \{X_1, \ldots, X_m\}$.

procedure ComputeMarginal(C_i, S_Q)
Input: Node/clique C_i in the (rooted) junction tree $J(\mathcal{M})$; collection of attributes S_Q whose (approximate) joint frequency distribution is required from the (sub)tree rooted at node C_i.
Output: Histogram giving the approximate frequency distribution of S_Q.
begin
1. **if** $S_Q \subseteq C_i$ **then return** `project`($H(C_i)$, S_Q)
2. **else**
3. let int := $C_i \cap S_Q$ and diff :=$S_Q - C_i$
4. **if** (diff $\subseteq$ `cover`(C_j) for some child C_j of C_i) **then**
5. **if** (int = ϕ) **then return ComputeMarginal**(C_j, S_Q)
6. **else**
7. let S_{ij} := $C_i \cap C_j$
8. H_1 := **ComputeMarginal**(C_j, diff $\cup S_{ij}$)
9. **return** `project`(`product`($H(C_i)$, H_1) , S_Q)
10. **end**
11. **else**
12. let $H := H(C_i)$
13. **for** each child C_j of C_i such that $C_j \cap$ diff $\neq \phi$ **do**
14. let S_{ij} := $C_i \cap C_j$
15. H_1 := **ComputeMarginal**(C_j, ($C_j \cap$ diff) $\cup S_{ij}$)
16. H := `product`(H, H_1)
17. **end**
18. **return** `project`(H , S_Q)
19. **end**
20. **end**
end

Figure 3: Computing marginal histograms using junction trees.

Algorithm **ComputeMarginal** is much more efficient than the naive technique described above and, in fact, it can be shown that **ComputeMarginal** is *optimal* in terms of the total number of histogram multiplications and projections required. On the other hand, **ComputeMarginal** does not address the issue of finding the optimal order for multiplying the relevant histograms for obtaining a given marginal. This problem is similar in spirit to the well-known *matrix-chain multiplication problem* [5] and, in fact, reduces to that problem in the case of a *simple path* model graph $\mathcal{M}$ and a two-variable range predicate P. Optimizing the multiplication operations for general model graphs and range predicates requires extending these earlier results to much more general *tensor products*, which is, to the best of our knowledge, an open problem. We intend

to address this issue as part of our future work on DB histograms.

3.3.2 Multiplying and Projecting Clique Histograms

We now discuss the implementation of the basic clique-histogram operations (i.e., `project`() and `product`()) that are used in our selectivity-estimation procedure. We focus on projection and multiplication algorithms for MHIST histograms; the algorithms for grid histograms are rather straightforward (this was actually the main reason we included them in our study) and they can be found in the full paper.

We propose a more space-efficient representation for multi-dimensional MHIST histograms than the one described in the original paper of Poosala and Ioannidis [18]. For an n-dimensional MHIST bucket, their representation requires storing a frequency and the high and low value boundaries in each of the n dimensions, resulting in a total of $(2n + 1)$ numeric values per bucket. Our key observation here is that an MHIST histogram is basically a hierarchical binary partitioning of the data space. Thus, instead of storing each MHIST bucket explicitly, we propose storing the histogram as a tree structure (termed a *split tree*) that captures the splits performed by the MHIST-construction algorithm. Each internal node of the split tree just needs to store the split value and the dimension along which the split was performed, whereas each leaf node just needs to store the frequency for the corresponding bucket. Given a b-bucket n-dimensional MHIST histogram, it is easy to verify that our split-tree representation requires storing only $(3b - 2)$ numbers, which is clearly a significant improvement over the $b(2n+1)$ numbers required by the naive representation.

We propose multiplication and projection algorithms for MHIST histograms that work solely on our space-efficient, split-tree representation for both the input and output histograms of the operator. Both algorithms make use of a simple procedure termed `restrictNode`(N, R) that takes as input a node N in a split tree for a collection of attributes and a restriction R on the ranges of (a subset of) these attributes. The result of `restrictNode` is a split tree derived from the subtree rooted at N that only contains the split and leaf nodes that pertain to the input range restriction R. The `restrictNode` operator requires (at most) one traversal of the subtree rooted at N. The pseudo-code for projecting an MHIST histogram $H(C)$ onto a subset $S \subset C$ of its attributes is shown in Figure 4. Most of the work for `project`($H(C), S$) is done in a recursive subroutine, termed `genSplits`, that essentially generates the split-tree structure for the projected histogram. The key requirement for `genSplits` is to ensure that, in the end, *all* splits along dimensions in S for any of the hierarchically-generated buckets in $H(C)$ are reflected in the split tree for the projected histogram. For example, if one bucket of $H(C)$ is split on $X = 10$ and another on $X = 20$ both 10 and 20 need to appear as split points when the histogram is projected onto X. This is handled by using the `restrictNode` procedure to "transfer" all the relevant splits into the currently explored subtree (Steps 9–12). (Note that N'_l and N'_r are interchangeable in Steps 8–12.) Finally, `project` computes the frequencies for the leaf nodes (i.e., buckets) in the split tree for the projected histogram by summing the frequencies over all the contributing leaves of $H(C)$; of course, the summed frequencies have to be appropriately scaled by the relative volume of the bucket along S (intra-bucket uniformity assumption) (Steps 2–4).

The pseudo-code of our algorithm for multiplying two MHIST clique histograms $H(C_i)$ and $H(C_j)$ to obtain an MHIST histogram $H(C_i \cup C_j)$ on the joint distribution of $C_i \cup C_j$ is depicted in Figure 5. The key intuition underlying our algorithm is as follows. Let $S_{ij} = C_i \cap C_j$ and consider two buckets $b_i \in H(C_i)$ and $b_i \in H(C_i)$. If b_i and b_j overlap along any of the dimensions in

---

procedure `project`($H(C)$, S)
Input: MHIST histogram (split tree) $H(C)$ on C; attributes $S \subset C$ on which we want to project $H(C)$.
Output: $H(S)$ MHIST histogram on S.
begin
1. $H(S)$:= `genSplits`(root($H(C)$) , S) /* split-tree for the projection */
2. **for** every leaf l of $H(S)$ **do**
3. frequency(l) := $\sum_{l'} w_{l'} \cdot$ frequency(l'), where the summation is over all leaves l' of $H(C)$ that contain l along the dimensions in S and $w_{l'}$ = volume(l along S)/volume(l' along S)
4. **end**
end

subroutine `genSplits`(N, S)
Input: Split-tree node N; subset S of the split-tree attributes on which we want to "project" the subtree rooted at N.
Output: N' root for a new split-tree structure resulting from the projection of the N-rooted subtree on S'.
begin
1. let X and v be the splitting attribute and split value at node N
2. let N_l and N_r denote the left and right child of N
3. N'_l := `genSplits`(N_l, S)
4. N'_r := `genSplits`(N_r, S)
5. **if** $X \in S'$ **then**
6. create new internal node N' with children N'_l and N'_r, and split (X, v)
7. **else**
8. set N' := N'_l
9. **for** every leaf l of N' **do**
10. let R denote the attribute ranges occupied by l
11. replace l with `restrictNode`(N'_r, R)
12. **end**
13. **end**
end

Figure 4: MHIST projection algorithm.

S_{ij}, then they result in a new bucket in the product histogram whose boundaries along the dimensions $C_i - S_{ij}$ ($C_j - S_{ij}$) are exactly those of b_i (resp., b_j), whereas its boundaries along dimensions in S_{ij} are defined by the intersection of b_i and b_j in those dimensions. Our `product` algorithm employs this observation to generate the split-tree structure for the product histogram by first initializing that structure with one of the input split trees, say $H(C_i)$ (Step 1), and then, for each leaf node of $H(C_i)$, using the `restrictNode` procedure on the other input tree to generate the tree structure for the product buckets (Steps 2–5). Finally, the frequencies for the leaf nodes in the product split tree are computed using the separation formula on the frequencies of the "enclosing" buckets of $H(C_i)$, $H(C_j)$, and $H(S_{ij}) =$ `project`($H(C_i)$, S_{ij}), which, of course, have to be properly scaled by the relative volume of the bucket intersection along S_{ij} (intra-bucket uniformity assumption) (Steps 7–11).

4. EXPERIMENTAL STUDY

In this section, we present the results of an extensive empirical study in which we compare the quality of approximate answers to selectivity estimation queries obtained using DB histograms with various prevalent selectivity estimation algorithms. The major findings of our study can be summarized as follows.

• **Decomposable Models are Effective.** For all data sets, decomposable models with small complexity (that is, few edges) yield good approximations to the original data set. Thus, decomposable models provide us with an effective mechanism for accurately capturing the data distribution of multi-dimensional data sets.

• **Better Approximate Answer Quality.** In general, the quality of the approximate answers returned by DB histograms is superior

procedure product($H(C_i)$, $H(C_j)$)
Input: MHIST histograms (split trees) $H(C_i)$ and $H(C_j)$ on attribute sets C_i and C_j, respectively.
Output: MHIST histogram $H(C_i \cup C_j)$ on $C_i \cup C_j$.
begin
1. set $H(C_i \cup C_j) := H(C_i)$ /* initialize with split tree for an input */
2. **for** every leaf l of $H(C_i \cup C_j)$ **do**
3. let R_S be the ranges of attributes in S occupied by the bucket at l
4. replace l with restrictNode($H(C_j)$, R_S)
5. **end**
6. $H(S_{ij}) :=$ project($H(C_i)$, S_{ij}), where $S_{ij} = C_i \cap C_j$
7. **for** every leaf l of $H(C_i \cup C_j)$ **do**
8. let l_i, l_j, l_{ij} be the leaves of $H(C_i)$, $H(C_j)$, and S_{ij} (respectively) that contain l along the respective dimensions
9. let w_{l_i} := volume(l along C_i)/volume(l_i along C_i), with w_{l_j} and $w_{l_{ij}}$ defined similarly
10. frequency(l) := $\frac{(w_{l_i} \cdot \text{frequency}(l_i)) \cdot (w_{l_j} \cdot \text{frequency}(l_j))}{(w_{l_{ij}} \cdot \text{frequency}(l_{ij}))}$
11. **end**
end

Figure 5: MHIST multiplication algorithm.

to that of competing histograming methods. Further, for a number of selectivity estimation queries, the approximation error with DB histograms is as much as 5 times smaller than the error for the best competing algorithm.

• DB **histograms are Storage Efficient.** DB histograms provide fairly accurate (less than 50% error) answers to range queries on most real-life data sets (one of which has a dimensionality as high as 12), while requiring less than 1% of the storage space consumed by the original data set.

Thus, our experimental results validate the thesis of this paper that DB histogram synopses provide a viable and effective means for approximating the joint distributions of high-dimensional data sets. Due to space constraints, we do not present the results of our experiments on construction times or query answering times for DB histograms; they can be found in the full paper [7]. Note, however, that histogram construction can be speeded up by using random samples of the data to build the interaction models, while the histogram usage techniques described in Section 3.3 can be used to process queries efficiently.

4.1 Experimental Testbed and Methodology

Selectivity Estimation Techniques. We consider three different selectivity estimation techniques in our study.

• $\mathcal{MHIST}$. We build a multi-dimensional histogram on all the attributes in the base relation using the MHIST-2 technique proposed by Poosala and Ioannidis [18]. Using our space-efficient tree representation for these histograms, the storage space required is approximately $9b$ bytes, where b is the number of histogram buckets. Here, $4b$ bytes are required to store the counts for the b leaves of the tree, and storing the split dimension and value for the $b-1$ internal nodes requires $b-1$ and $4b-4$ bytes, respectively[2].

• $\mathcal{IND}$ (*Independence Assumption).* In this approach, a separate one-dimensional histogram for each attribute of the base table is built. For the purpose of answering queries, it is assumed that all data attributes are mutually independent, and the joint distribution can be obtained as the product of the individual one-dimensional marginals. The storage space required for b buckets of $\mathcal{IND}$ histograms is $8b$ bytes (4 bytes to specify the separator for each bucket and 4 bytes for the bucket frequency). The buckets for $\mathcal{IND}$ are constructed using a procedure similar to **IncrementalGains** (see Figure 2) using the set of all attributes as the set of cliques input to the procedure. We used the total variance across all the histogram buckets as the error function.

• DB *Histograms.* We construct DB histograms using the algorithms described in Sections 3.1 and 3.2. We consider the following two heuristics for selecting the next best interaction edge to add to model $\mathcal{M}$ in the forward-selection process.

- $\mathcal{DB}_1$: The edge with the highest statistical significance for the improvement in divergence is chosen.
- $\mathcal{DB}_2$: The edge for whom the ratio of the improvement in divergence and the increase in the total model state space (that is, the sum of the product of the domain sizes for all the attributes in each clique) is maximum, is selected. (The justification for the $\mathcal{DB}_2$ heuristic can be found in [7].)

We use $\mathcal{MHIST}$ histograms for approximating the frequency distributions of the cliques in the decomposable model $\mathcal{M}$ of the DB histogram. Thus, the storage space required for b buckets is $9b$ bytes, similar to $\mathcal{MHIST}$. Further, procedure **IncrementalGains** is used to allocate space among the various clique histograms (with the total variance across all buckets as the error function). There is an additional overhead of storing the junction tree associated with the model, but that is negligible compared to the size of the histogram synopsis.

Unless stated otherwise, when inferring the model $\mathcal{M}$ for a data set, we fix k_{max}, the maximum clique size, to be 2, since we found that including 3-dimensional clique histograms decreases the accuracy of DB histograms considerably. As a consequence, for a relation containing n attributes, $\mathcal{M}$ is a tree and the DB histogram for the table contains $n-1$ clique histograms. We also set the threshold for the statistical significance level, θ, to 90%. However, we found that due to the small value for k_{max}, the statistical significance rarely eliminated edges during model construction.

One important selectivity estimation technique that we do not include in our empirical study is *random sampling.* This technique involves keeping a random sample of the data in memory, against which the queries are run and the answer is appropriately scaled to estimate the result of executing the query on the complete database. In our experiments, we observed that, because of the small storage space allocated to the synopses, hardly any tuples in the sample satisfied the query, and the returned answer was almost always 0. Therefore, we do not consider this technique any further.

Real-life Data Sets. In our experiments, we used real-life data sets obtained from the US Census Bureau (www.census.gov/) as well as a data set on California housing from a 1990 survey (lib.stat.cmu.edu/). Due to space constraints, our results with the California housing data set can be found in [7].

• *Census Data Set.* We use the Current Population Survey (CPS) data source and within it, the Person Data Files of the March Questionnaire Supplement. We use two different attribute subsets.

1. *Data Set 1.* This data set consists of the following attributes (the size of each attribute domain is included in parentheses): *race (4), native country of the sample person (113), native country of mother (113), native country of father (113), citizenship (5)* and *age (91).* We expect the first five attributes to be highly correlated, while the last attribute is relatively independent of the rest of the attributes. This data set contains a total of 125705 tuples with 13449 distinct tuples, and has a total size of approximately 315KB.
2. *Data Set 2.* To demonstrate that our techniques can handle data sets with high dimensionality well, we use a 12 attribute

[2] Here we assume that, for an internal node in the split tree, storing the split dimension requires 1 byte, regardless of the dimensionality of the data set.

projection of the Census data set. In addition to the attributes described above, in this data set, we include the following attributes: *industry code(237), No. of hours usually worked at the main job (88), educational attainment (17), census state code (51), county code (91).* Data set 2 contains a total of 83566 tuples with 63090 distinct tuples, and has a total size of approximately 2.88MB.

Query Workload. We compare the selectivity estimation techniques on a randomly generated range-selectivity query workload. Each query on a single relation specifies ranges for a subset of attributes in the relation and leaves the ranges for the remainder of the attributes unspecified. We refer to a query with k specified attribute ranges as a k-D query. Given a k, the k-D query workload consists of 100 randomly generated k-D queries; for each query, the k attributes and the range extents for those attributes are randomly chosen. In our final reported results, we do not consider queries that cover less than 100 tuples in the base relation.

Answer-quality Metrics. The answers obtained using the three histograming techniques described above are compared with the correct answer computed using the original data set. We use one of two metrics (described below) to gauge the quality of an approximate answer to a query. In the following, the correct answer to a query is a and the approximate answer computed using a synopsis is a_s.

1. *Absolute relative error,* defined as $|a_s - a|/a$.
2. *Multiplicative error,* defined as $\max\{a_s, a\}/\min\{a_s, a\}$.

While the relative error is fairly standard, the main reason to use the less common multiplative error metric is that, for higher dimensionality queries, $\mathcal{IND}$ tends to give very small answers for most queries. But even for an answer equal to 0, the absolute relative error is still at most 1. As we will see later, because of this, $\mathcal{IND}$ appears to perform better for high dimensionality queries. The multiplicative error metric corrects this shortcoming of the relative error metric by penalizing very small answers as well.

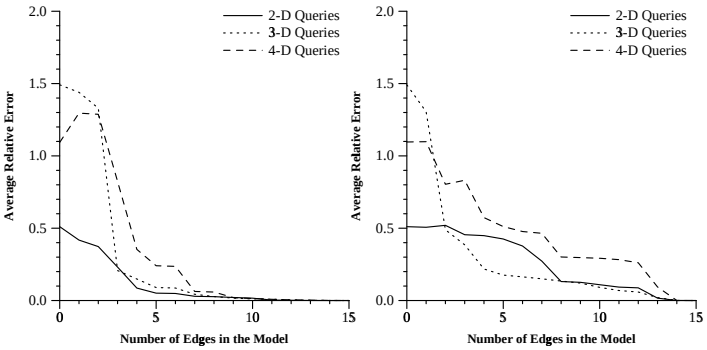
In the graphs depicting the results of our experiments, the final reported error for each query workload is obtained by taking the average of the errors for the 100 random queries in the workload.

4.2 Experimental Results

4.2.1 Census Data Set 1

How *good* are Decomposable Models? Our first experiment demonstrates the effectiveness of decomposable models at approximating the original data set with respect to our range-query workloads. In this experiment, edges are added to the model in decreasing order of the statistical significance of the improvement in approximation due to the edge (disregarding the parameters k_{max} and θ). Further, for each clique in the model, we store the projection of the entire data set on clique attributes, and use these projections to answer queries. Each projection, in effect, corresponds to a clique histogram with an unlimited number of buckets, and thus captures the data distribution on clique attributes completely and accurately. Consequently, errors in the final query answer are solely due to the model and approximation errors due to clique histograms are factored out from the final result.

Figure 6 depicts the performance of the models selected by $\mathcal{DB}_1$ and $\mathcal{DB}_2$ for query workloads with different dimensionalities. As we can see, the average errors for both these edge selection heuristics drop rapidly as the complexity of the model increases. For instance, for the decomposable model containing only 4 edges and selected using $\mathcal{DB}_1$, the error is less 10%. Thus, it follows that decomposable models are good at accurately capturing correlations in the underlying data. Further, observe that the error rates for the model chosen using $\mathcal{DB}_1$ drop much more rapidly compared to $\mathcal{DB}_2$. This is not entirely surprising since $\mathcal{DB}_1$ selects edges for the model assuming perfect clique histograms, which is true in this experiment. However, as we will see later, $\mathcal{DB}_2$ is perhaps more suitable in practice, since for a majority of realistic environments, due to storage space limitations, each clique histogram can be expected to only coarsely approximate the original distribution.



(i) DB1 (ii) DB2

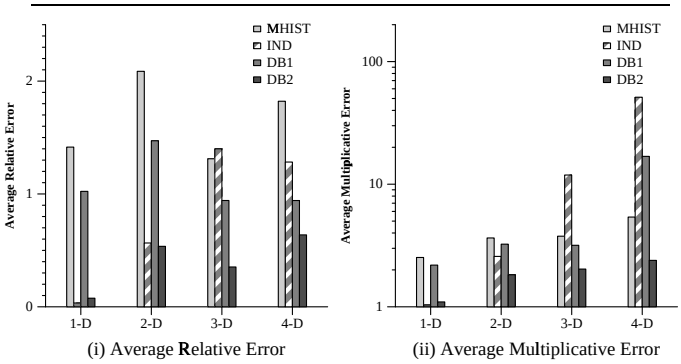
Figure 6: Effectiveness of Decomposable Models.

DB **Histogram Accuracy.** In Figure 7, we plot the relative and multiplicative errors for the various selectivity estimation techniques on k-D query workloads ($k = 1, \ldots, 4$). The storage space for histograms allocated to each method was fixed at 3KB. From the graphs, it follows that $\mathcal{DB}_2$ outperforms the other techniques for all k-D query workloads (except $k = 1$); further, for certain query workloads ($k = 3$), the error for $\mathcal{DB}_2$ is half the error for competing methods. In general, both the DB-histograming techniques $\mathcal{DB}_1$ and $\mathcal{DB}_2$ perform quite well since decomposable models are good at separating unrelated attributes, and traditional histogram methods like $\mathcal{MHIST}$ approximate low-dimensional data sets quite well. Further, unlike $\mathcal{DB}_1$, since $\mathcal{DB}_2$ also takes into account the space requirements of clique histograms when selecting edges for the model and the amount of storage is limited, it results in the smallest values for error.

Note that the poor performance of $\mathcal{MHIST}$ can be attributed to the well-known fact that traditional histograming approaches suffer from the "curse of dimensionality" when trying to approximate distributions for high-dimensional data sets. Also, since $\mathcal{IND}$ builds only one-dimensional histograms, it should come as no surprise that it results in the lowest errors for 1-D queries. However, its performance with respect to relative error is somewhat misleading since $\mathcal{IND}$ typically returns small answers for most queries. The multiplicative error is more indicative of $\mathcal{IND}$'s overall performance, which suffers because of the invalid assumption made by it that attributes are mutually independent.

Effect of Storage Space on Histogram Accuracy. Figure 8 depicts errors for a 3-D query workload as storage space allocated to the selectivity estimation techniques is increased. The behavior for other query workloads show a similar trend.

From Figure 8, we can see that errors for $\mathcal{DB}_2$ and $\mathcal{DB}_1$ decrease as the amount of storage space is increased. This is because the increased storage helps each low-dimensional clique histogram to approximate the data more accurately, and thus improves the overall accuracy of the DB histogram. However, the extra space has little effect on $\mathcal{MHIST}$ and $\mathcal{IND}$ due to inherent problems related to approximating high-dimensional data sets and the attribute independence assumption.



(i) Average Relative Error (ii) Average Multiplicative Error

Figure 7: Results for the 6-D Census Data Set.

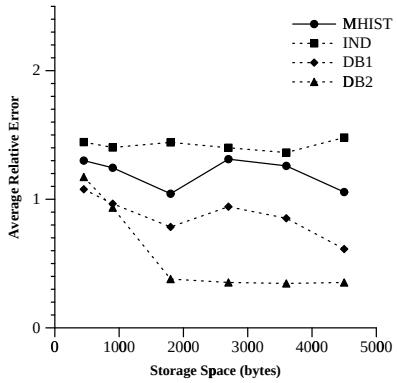
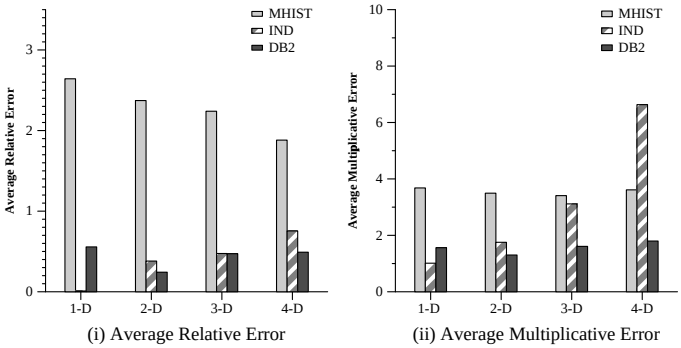


Figure 8: Effect of Storage Space.

4.2.2 Census Data Set 2

We present results from our experiments on the Census data set 2 in Figure 9. The total storage space allocated to the synopses for this data set was 20KB (approximately 0.67% of the original data set). The results are similar to those for the Census data set 1, with $\mathcal{DB}_2$ doing much better than the other two techniques on the combination of our two error metrics.



(i) Average Relative Error (ii) Average Multiplicative Error

Figure 9: Results for the 12-D Census Data Set.

5. CONCLUSIONS

In this paper, we have proposed DEPENDENCY-BASED (DB) histograms, a novel approach to histogram-based synopses that effectively overcomes the "curse of dimensionality" by employing the solid foundation of statistical interaction models to explicitly identify and exploit the dependence patterns in the data. The basic idea is to break the synopsis into (1) a decomposable interaction model that accurately captures the significant attribute correlations and independencies in the data, and (2) a collection of histograms on low-dimensional marginals that, based on the model, can be used to derive accurate approximations of the overall joint data distribution. Our experimentation with different real-life data sets has validated our approach, demonstrating that DB histograms provide significantly better approximations than conventional histograming techniques. An important feature of our general methodology is that it can be used to enhance the performance of several other data-reduction techniques in medium- to high-dimensionality spaces: statistical interaction models can help identify the significant attribute correlation patterns in the data and, therefore, the interesting lower-dimensional subspaces that should be approximated independently in a synopsis.

Acknowledgments: We would like to thank José Pinheiro and Vishy Poosala for several helpful discussions related to this work.

6. REFERENCES

[1] Y. M.M. Bishop, S. E. Fienberg, and P. W. Holland. *"Discrete Multivariate Analysis"*. The MIT Press, 1975.

[2] J. R.S. Blair and B. Peyton. "An Introduction to Chordal Graphs and Clique Trees". In *"Graph Theory and Sparse Matrix Computation"*. Springer-Verlag NY, Inc., 1993.

[3] K. Chakrabarti, M. Garofalakis, R. Rastogi, and K. Shim. "Approximate Query Processing Using Wavelets". In *Proc. of the 26th Intl. Conf. on Very Large Data Bases*, 2000.

[4] R. Christensen. *"Log-Linear Models and Logistic Regression"*. Springer-Verlag NY, Inc. (Springer Series in Statistics), 1997.

[5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *"Introduction to Algorithms"*. MIT Press, 1990.

[6] J.N. Darroch, S.L. Lauritzen, and T.P. Speed. "Markov Fields and Log-Linear Interaction Models for Contigency Tables". *The Annals of Statistics*, 8(3):522–539, 1980.

[7] A. Deshpande, M. Garofalakis, and R. Rastogi. "Independence is Good: Dependency-Based Histogram Synopses for High-Dimensional Data". Bell Labs Tech. Report, 2001.

[8] D. Edwards. Personal Communication, September 2000.

[9] C. Faloutsos and I. Kamel. "Relaxing the Uniformity and Independence Assumptions Using the Concept of Fractal Dimension". *Journal of Computer and Systems Sciences*, 55(2):229–240, 1997.

[10] B. Fox. "Discrete Optimization Via Marginal Analysis". *Management Science*, 13(3):211–216, 1966.

[11] D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi. "Approximating Multi-Dimensional Aggregate Range Queries Over Real Attributes". In *Proc. of the 2000 ACM SIGMOD Intl. Conf. on Management of Data*, 2000.

[12] T. Ibaraki and N. Katoh. *"Resource Allocation Problems – Algorithmic Approaches"*. MIT Press, 1988.

[13] Y. E. Ioannidis and S. Christodoulakis. "Optimal Histograms for Limiting Worst-Case Error Propagation in the Size of Join Results". *ACM Trans. on Database Systems*, 18(4):709–748, 1993.

[14] Finn V. Jensen and Frank Jensen. "Optimal Junction Trees". In *Proc. of the 10th Annual Conf. on Uncertainty in AI*, 1994.

[15] Francesco M. Malvestuto. "Approximating Discrete Probability Distributions with Decomposable Models". *IEEE Trans. on Systems, Man, and Cybernetics*, 21(5):1287–1294, 1991.

[16] S. Muthukrishnan, V. Poosala, and T. Suel. "On Rectangular Partitionings in Two Dimensions: Algorithms, Complexity, and Applications". In *Proc. of the Intl. Conf. on Database Theory*, 1999.

[17] J. Pearl. *"Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference"*. Morgan Kaufmann Publishers, Inc., 1988.

[18] V. Poosala and Y. E. Ioannidis. "Selectivity Estimation Without the Attribute Value Independence Assumption". In *Proc. of the 23rd Intl. Conf. on Very Large Data Bases*, 1997.

[19] V. Poosala, Y. E. Ioannidis, P. J. Haas, and E. J. Shekita. "Improved Histograms for Selectivity Estimation of Range Predicates". In *Proc. of the 1996 ACM SIGMOD Intl. Conf. on Management of Data*, 1996.

[20] J. Whittaker. *"Graphical Models in Applied Multivariate Statistics"*. John Wiley & Sons, Inc., 1990.