Detecting Correlated Columns in Relational Databases with Mixed Data Types

Hoang Vu Nguyen[•] Emmanuel Müller[•○] Periklis Andritsos[◇] Klemens Böhm[•]

[•]Karlsruhe Institute of Technology (KIT), Germany
{hoang.nguyen, emmanuel.mueller, klemens.boehm}@kit.edu

[○]University of Antwerp, Belgium
emmanuel.mueller@uantwerp.be

[◇]University of Toronto, Canada
periklis.andritsos@utoronto.ca

ABSTRACT

In a database, besides known dependencies among columns (e.g., foreign key and primary key constraints), there are many other correlations unknown to the database users. Extraction of such hidden correlations is known to be useful for various tasks in database optimization and data analytics. However, the task is challenging due to the lack of measures to quantify column correlations. Correlations may exist among columns of different data types and value domains, which makes techniques based on value matching inapplicable. Besides, a column may have multiple semantics, which does not allow disjoint partitioning of columns. Finally, from a computational perspective, one has to consider a huge search space that grows exponentially with the number of columns.

In this paper, we present a novel method for *detecting column correlations* (DECOREL). It aims at discovering overlapping groups of correlated columns with mixed data types in relational databases. To handle the heterogeneity of data types, we propose a new correlation measure that combines the good features of Shannon entropy and cumulative entropy. To address the huge search space, we introduce an efficient algorithm for the column grouping. Compared to state of the art techniques, we show our method to be more general than one of the most recent approaches in the database literature. Experiments reveal that our method achieves both higher quality and better scalability than existing techniques.

1. INTRODUCTION

In relational databases, there are different types of column dependencies, e.g., foreign key and primary key constraints, conditional functional dependencies. Besides, columns can be correlated while not having any explicit value association. This is because from a design point of view, databases tend to use several columns with different representations to capture correlated information: zip codes, cities, states, longitudes, and latitudes. Second, correlations among columns also naturally emerge due to their statistical properties, e.g., in *TPC-H* the status of an order and the date it is placed are correlated while their value domains apparently do not have anything in common [34]. Another example of column correlations and their importance in real-world applications is given below.

SSDBM '14, June 30 - July 02 2014, Aalborg, Denmark

http://dx.doi.org/10.1145/2618243.2618251

EXAMPLE 1. *The facility management at our university (KIT) collects and stores energy consumption indicators, e.g., electricity, gas, and heating. Besides the explicitly specified column dependencies, there are correlations among the real-valued consumption indicators themselves, with the total area of buildings (real-valued numerical), the number of employees in buildings (discrete numerical), and the location of buildings (categorical). Such correlations are unknown a priori. However, they are highly interesting to and useful for the facility management. First, they help the management to better understand the data. Second, they facilitate the discovery of interesting patterns in consumption indicators by steering the management's focus to groups of correlated columns. Third, facility management can create materialized views based on interesting column correlations extracted for efficient data exploration [29]. Lastly, column correlations can be exploited to construct prediction models to monitor energy consumption and hence, save the management from investing in new energy smart meters.*

Besides the scenario described in Example 1, column correlations are also useful for relational databases from other domains. In particular, for database research, column correlations are known to be useful for schema matching/extraction [3, 13], index recommendation [15], query optimization (e.g., selectivity estimation) [31], and other efficiency improvements (by creating materialized views, data partitions, etc.). Regarding schema matching and extraction, as shown later (Section 7), groups of correlated columns encompass various types of column relationships, such as: (a) foreign key and primary key constraints, (b) two foreign keys referring to the same primary key, (c) a column in a base table and its derived columns in views, (d) two columns in two views originating from the same column in a base table, and (e) columns with no explicit relationship but semantically correlated. Besides, detecting groups of correlated columns also assists data analytics in, e.g., clustering high dimensional databases [24], and association analysis [20].

However, discovering groups of correlated columns is difficult due to several open challenges. First, since databases store information in different formats, one needs a correlation measure that can handle different data types, namely, real-valued numerical, discrete numerical, and categorical. Existing work either focuses only on the two latter types [4, 9] or is applicable to only the first type [23, 24]. Columns with heterogeneous data types in turn abound in many real-world databases.

Second, it is unclear how to decide if a group of columns with arbitrary data types is correlated. Besides being inapplicable to all

three data types, existing work focuses on pairwise correlation analysis. However, correlations among columns usually are multi-way (also called 'mutual'). Examples of mutual correlations include composite keys and hence, composite functional dependencies [8, 30] where several columns functionally govern other columns.

Third, a column may have multiple semantics making disjoint partitioning of columns undesirable. For instance, in our *Energy* database, the electricity column belongs to a group representing the interaction between energy consumption and time. It is also part of a group capturing the correlation between energy consumption of buildings and their characteristics (e.g., total area, location). Hence, a grouping where columns may participate in multiple groups is required. Recent studies [3, 12, 34] target disjoint groupings and do not address this issue.

Fourth, the induced search space grows exponentially with the number of columns. It does not have any explicit (anti-)monotonicity property as correlation scores in general are not (anti-)monotonic [27, 35]. Thus, techniques reminiscent of the Apriori search [8, 14, 24, 30] are inapplicable. Work not relying on the Apriori search, such as [23], however focuses exclusively on numerical data. This leaves space for further research on how to efficiently detect groups of correlated columns in mixed data typed databases.

Finally, when mining overlapping groups, one may end up with a large number of groups, which essentially convey similar information regarding the underlying database. This hinders manual inspection and post-analysis, e.g., when the facility management of KIT wants to select a few interesting groups for further analysis. Most existing studies [8, 14, 24, 30] do not take this into account and overwhelm users with redundant output.

We address all five challenges by proposing DECOREL, our novel method for *Detecting Column Correlations*. In short, to tackle the large search space without any explicit (anti-)monotonicity property, we show that one can find mutual correlations through pairwise correlations. Based on this result, DECOREL solves the problem based on efficient pairwise correlation analysis in a graph of database columns. This helps DECOREL to both achieve high quality of groups detected and scale for large databases. For correlation analysis, we propose CORR—our novel pairwise correlation measure that works with both cumulative distribution functions and probability mass functions, i.e., it is applicable to different data types. To ensure succinctness of the output, we present a method based on the Minimum Description Length principle [7]. It produces a lossless summary of the groups output, and achieves up to an order of magnitude size reduction ratio in our experiments.

Overall, we make the following contributions:
(1) We show the feasibility of estimating mutual correlations by pairwise correlations. Thereby, we transform the search space into a column graph based on pairwise correlations. We efficiently mine this graph to find overlapping groups of correlated columns.
(2) We introduce a new correlation measure that is applicable to different data types, in particular, real-valued numerical, discrete numerical, and categorical. We describe how to compute our measure efficiently on empirical data.
(3) We present an MDL-based output summarization phase, which is geared towards guaranteeing succinctness of output groups, and hence, facilitates manual inspection and post-analysis.
(4) As an example of the generality and applicability of correlation analysis to discovering column relationships, we show that our correlation test is more general than a state of the art distribution test [34], and hence, can discover column relationships missed by previous approaches.
(5) Experiments on both synthetic and real-world databases show DECOREL to achieve higher quality than existing methods, in particular when handling databases with mixed data types. Further, DECOREL also outperforms competitors in terms of scalability with both the database size and the number of columns.

Our paper is organized as follows. Section 2 covers related work. Section 3 provides preliminaries and general notions. Section 4 gives the details of our correlation measure. Section 5 explains how to find mutual correlations through pairwise correlations. Section 6 describes our algorithm for grouping correlated columns. Section 7 presents our analysis on the generality of DECOREL compared to a recent approach [34]. Section 8 reports our empirical study. Section 9 concludes the paper with directions for future work.

2. RELATED WORK

ECLUS [34] groups database columns that have similar distributions. It differs from DECOREL in many aspects. First, it uses the EMD test [18] which is bound to discover groups of columns with the same data type and overlapping value domains. A detailed analysis on this issue is given in Section 7. Second, it only detects non-overlapping groups and hence, misses other semantics that a column may have.

CORDS [9] mines overlapping pairs of correlated columns. Nevertheless, it lacks a mechanism to combine those pairs into larger meaningful groups. In addition, its χ^2 test is only applicable to discrete and categorical data. DECOREL in turn detects overlapping groups of two or more columns and is applicable to real-valued, discrete, and categorical data.

GORDIAN [30] and DUCC [8] aim at finding overlapping composite primary keys. That is, unlike DECOREL, they do not detect groups of columns that are correlated but do not form any key.

Finding overlapping groups of correlated columns is also addressed by selectivity estimation methods, such as [31]. Nevertheless, for efficiency reasons, they keep the group sizes small (typically 2). Thus, their goal is not to optimize the quality of groups. In contrast, DECOREL discovers groups that are potentially useful for multiple purposes, e.g., selectivity estimation. Applying DECOREL to each specific task is a subject of our future work.

Clustering relational columns has also been explored in [1]. It groups columns based on generic data types. The type information of columns is captured using q-grams. Thus, this method can also benefit DECOREL when there is insufficient information to correctly identify data types of columns.

Performing correlation analysis in relational database schemas has already been studied in, for example, [9, 13, 33]. However, existing techniques resort to either the χ^2 test, or the traditional information theoretic correlation measures (e.g., mutual information). These measures are in turn designed specifically for discrete and categorical data. Hence, they are unable to handle real-valued data well. DECOREL in contrast works on mixed data types, i.e. with both cumulative distribution functions and probability mass functions. Thus, it is able to avoid the limitations of existing correlation measures.

Mining groups of correlated columns can be seen as mining correlated subspaces. Most existing solutions [5, 14, 24] model the search space as a lattice and explore it in an Apriori manner. Nevertheless, this assumption does not always hold since correlation scores in general are not (anti-)monotonic [27, 35]. Moreover, Apriori search also suffers from efficiency issues and tends to detect only fragments of high dimensional correlated subspaces [23]. Non-levelwise search does exist [23], however focuses exclusively on numerical data. DECOREL in turn is both efficient and able to detect groups of correlated columns in databases with heterogeneous data types.

3. PRELIMINARIES

Let $\mathcal{R}$ be the set of relations in the database. We write $\mathcal{C}$ as the set of all columns in $\mathcal{R}$. We regard each column $C \in \mathcal{C}$ as a random variable with a distribution $p(C)$. For notational convenience, we use C to represent both the column C and its associated random variable. If C is discrete or categorical, $p(C)$ is named the *probability mass function* (pmf for short). Otherwise, i.e., C is real-valued, $p(C)$ is called the *probability density function* (pdf for short). Let $g = \{C_i\}_{i=1}^{d} \subseteq \mathcal{C}$ be a group of columns. We write $p(C_1, \ldots, C_d)$ as the joint probability function of all columns in g, or of g for short. If g contains all discrete, or all categorical, or a mix of discrete and categorical columns, $p(C_1, \ldots, C_d)$ is a pmf. If g contains all real-valued columns, $p(C_1, \ldots, C_d)$ is a pdf. For simplicity, we call it probability function in any case.

A correlation function CORR assigns each group g with at least two columns a non-negative real-valued correlation score, denoted as both CORR(g) and CORR$(C_1, \ldots, C_d)$. In principle, CORR(g) quantifies the extent to which its joint probability function differs from the product of its marginal probability functions [27]. The larger the difference, the higher CORR(g) is, i.e., the more correlated the columns of g are. When g has a high correlation score, we regard g to be a group of correlated columns. In that case, the columns of g are (a) mutually (multi-way) correlated if g has more than two columns, and (b) pairwise correlated otherwise. (We discuss later how to decide if a correlation score is high.) When the context is clear, we omit 'pairwise' and 'mutually'.

To facilitate our discussion, we now review two popular correlation measures: mutual information (for two columns only) and total correlation (for more than two columns) [6]. They are based on (differential) Shannon entropy and are widely used in the database literature [13, 31, 33]. Their definitions are as follows:

DEFINITION 1. *(*MUTUAL INFORMATION*) The mutual information of two columns C_1 and C_2 is*

$$I(C_1, C_2) = H(C_1) - H(C_1|C_2) = H(C_2) - H(C_2|C_1)$$

where $H(.)$ is the (differential) Shannon entropy.

DEFINITION 2. *(*TOTAL CORRELATION*) The total correlation of $\{C_i\}_{i=1}^{d}$ is*

$$T(C_1, \ldots, C_d) = \sum_{i=1}^{d-1} H(C_{i+1}) - H(C_{i+1}|C_1, \ldots, C_i) \quad .$$

To ease presentation, we first describe CORR in the next section. Then we explain the intuition of reasoning about mutual correlations by means of pairwise correlations in Section 5. Finally, we introduce our efficient group discovery algorithm in Section 6.

4. OUR CORRELATION MEASURE

As a basic building block of our method, we present our correlation measure in this section. Our general goal is to mine groups with arbitrary numbers of correlated columns. Yet we will show later that one can find such groups by analyzing pairs of correlated columns. As a result, our CORR measure of correlation is *pairwise* (we describe how to form the joint distribution of two columns in Section 4.3.1). To understand CORR, we first discuss the limitations of mutual information and total correlation on real-valued data. For brevity, we focus on mutual information. However, the same arguments hold for total correlation.

4.1 Issues of Mutual Information

In short, mutual information is unsuited for real-valued columns in both theoretical and practical aspects. From a theoretical point of view, for two columns $X, Y \in \mathcal{C}$, $I(X, Y) \geq 0$ with equality iff X and Y are independent, i.e., $p(X, Y) = p(X) \cdot p(Y)$. Intuitively, the more X and Y are correlated, the lower the conditional entropies $H(X|Y)$ and $H(Y|X)$ are, i.e., the higher their mutual information (see Definition 1). Thus, a high mutual information score *often* indicates a correlation between X and Y. We use the word *often* because Shannon entropy, a constituent element of traditional mutual information, is only well defined for discrete/categorical data. Its continuous form (differential entropy) suffers from some unexpected properties, such as [26]: (a) it can be negative, and (b) that the differential entropy of X given Y equals to zero does not imply that X is a function of Y. Thus, unlike the discrete/categorical case, a high mutual information score between real-valued X and Y, though indicating that X and Y are correlated, conveys less information on their actual correlation. In particular, it does not say if X is a function of Y or vice versa.

From a practical point of view, to compute mutual information for the real-valued case, we need the pdfs, which usually are unavailable and need to be estimated, e.g., by discretization. Such an estimation in turn tends to produce overly simple or complex histograms due to the lack of knowledge on the number and width of histogram bins. As a consequence, computing mutual information on real-valued data may cause inaccurate correlation scores, i.e., wrong groupings of columns.

To overcome the problems of mutual information, next we introduce a specific handling of real-valued columns based on cumulative entropy—a substitute of Shannon entropy for real-valued data.

4.2 Cumulative Entropy

To address the drawbacks of differential entropy and hence, mutual information on real-valued data, we propose to work with *cumulative distributions* that can be computed directly on empirical data. In particular, we aim at a notion of entropy defined using cumulative distributions of real-valued columns. Therefore, we introduce *cumulative entropy (CE)*, as follows:

DEFINITION 3. *(*CUMULATIVE ENTROPY CE*) The cumulative entropy of a real-valued column X, denoted as $h(X)$, is*

$$h(X) = -\int P(X \leq x) \log P(X \leq x) dx \quad .$$

Our notion of cumulative entropy is based on [26, 24]. Similarly to differential entropy, the cumulative entropy of X captures the amount of uncertainty contained in X. Differently from differential entropy, it is defined based on the cumulative distribution $P(X \leq x)$. Since $0 \leq P(X \leq x) \leq 1$, we obtain $h(X) \geq 0$ (we adopt the standard convention that $0 \log 0 = 0$ [20]). This means that CE is always non-negative, just like Shannon entropy defined on discrete/categorical data. The conditional CE is given as.

DEFINITION 4. *(*CONDITIONAL CE*) Consider a column Y. If Y is discrete or categorical, then*

$$h(X|Y) = \sum h(X|y)p(y)$$

where $p(Y)$ is the pmf of Y. Otherwise,

$$h(X|Y) = \int h(X|y)p(y)dy$$

where $p(Y)$ is the pdf of Y.

The conditional CE has two important properties as follows.

THEOREM 1. *First, $h(X|Y) \geq 0$ with equality iff X is a function of Y. Second, $h(X|Y) \leq h(X)$ with equality iff X is independent of Y, i.e., $p(X,Y) = p(X) \cdot p(Y)$.*

The proof of Theorem 1 is available on our website.[1] These two properties of CE are identical to those of Shannon entropy on discrete and categorical data. Thus, CE is a reliable replacement of differential entropy on real-valued data. Following [24], $h(X)$ can be computed in closed form. We now proceed to introduce CORR.

4.3 Correlation Measure CORR

Our correlation measure CORR essentially combines the nice properties of both Shannon entropy and cumulative entropy. In particular, assume that we want to measure the correlation between two columns X and Y. $\text{CORR}(X,Y)$ is defined as:

DEFINITION 5. *(CORRELATION MEASURE CORR)* $\text{CORR}(X,Y)$ *is equal to*

- $H(X) - H(X|Y)$ *if X is categorical,*
- $h(X) - h(X|Y)$ *if X is numerical (either real-valued or discrete).*

Note that if Y is discrete or categorical, then

$$H(X|Y) = \sum H(X|y)p(y) \quad .$$

Otherwise,

$$H(X|Y) = \int H(X|y)p(y)dy \quad .$$

Since $\text{CORR}(X,Y)$ may not be equal to $\text{CORR}(Y,X)$, CORR is asymmetric. This is beneficial for analyzing asymmetric dependencies among columns; one of which is the functional dependency [9]. For instance, zip code functionally determines the city name but the reverse may not be true.

EXAMPLE 2. *Consider our Energy database from Example 1. Let E be the column capturing the electricity consumption (real-valued) of each building, N be the column for its number of staffs (discrete), and L be the column for its location (categorical). Following Definition 5, we have:*

$$\text{CORR}(E,N) = h(E) - h(E|N)$$

$$\text{CORR}(E,L) = h(E) - h(E|L)$$

$$\text{CORR}(N,E) = h(N) - h(N|E)$$

$$\text{CORR}(N,L) = h(N) - h(N|L)$$

$$\text{CORR}(L,E) = H(L) - H(L|E)$$

$$\text{CORR}(L,N) = H(L) - H(L|N)$$

CORR *is asymmetric, e.g., in general* $\text{CORR}(E,L) \neq \text{CORR}(L,E)$.

In Definition 5, when X is numerical, we compute $\text{CORR}(X,Y)$ using CE since CE is applicable to both real-valued and discrete data. This helps us to avoid further checks of data types. In reality, database programmers sometimes declare discrete columns as real-valued, and such a check may cost additional effort. We prove the following result:

[1] http://www.ipd.kit.edu/~nguyenh/ssdbm14-appendix.pdf

		X		
		r	**d**	**c**
Y	**r**	Case 1	Case 1	Case 1
	d	Case 2	Case 2	Case 2
	c	Case 3	Case 3	Case 3

Table 1: Matrix of computation. 'r' is for real-valued, 'd' for discrete, and 'c' for categorical.

THEOREM 2. $\text{CORR}(X,Y) \geq 0$ *with equality iff X and Y are statistically independent.* $\text{CORR}(X,Y)$ *attains its maximal value when X is a function of Y.*

PROOF. If X is categorical, we have $\text{CORR}(X,Y) = H(X) - H(X|Y)$. Following [6], $H(X) \geq H(X|Y)$, i.e., $\text{CORR}(X,Y) \geq 0$. Equality happens iff $H(X) = H(X|Y)$, i.e., X and Y are statistically independent. In addition, $\text{CORR}(X,Y) \leq H(X)$ with equality iff $H(X|Y) = 0$, which means X is a function of Y.

If X is numerical, we have $\text{CORR}(X,Y) = h(X) - h(X|Y)$. Using Theorem 1, we arrive at the results. □

Based on Theorem 2, we derive the following lemma:

LEMMA 1. $\text{CORR}(X,Y) > 0$ *iff $p(X,Y)$ and $p(X) \cdot p(Y)$ are different.*

We will use Lemma 1 in Section 5 where we explain how to find mutual correlations by means of pairwise correlations.

4.3.1 Forming joint distributions

Assume that we want to compute $\text{CORR}(X,Y)$ where X and Y are two columns. If X and Y belong to the same relation R, their joint distribution is defined as the projection of R onto (X,Y) (duplicates are kept).

If X and Y belong to two different relations R_1 and R_2, respectively, without any join relationship, their joint distribution is undefined, and they are considered to be independent.

If R_1 and R_2 have some join relationship, then we form a joint distribution for X and Y by performing a *left outer join* between R_1 and R_2. In this scenario, the outer join is preferred to the inner join since we want to punish unmatched values. The left outer join is used to reflect the asymmetric nature of CORR.

4.3.2 Computing CORR on empirical data

Given the definition of the CORR measure, we now describe how we compute it based on the type of the values stored in column Y. A detailed mapping of which case to apply for $\text{CORR}(X,Y)$ is given in Table 1.

Case 1: Y is real-valued

W.l.o.g., we assume that X is categorical. The case for when X is numerical follows similarly. To compute $\text{CORR}(X,Y)$, we need to compute $H(X)$ and $H(X|Y)$. Computing $H(X)$ is straightforward. In the following, we focus on $H(X|Y)$.

Due to the low support characteristic of real-valued data, each specific realization y of Y may be unique. Hence, each tuple (x,y) in the joint distribution of X and Y is likely unique [17, 28]. According to Definition 5, we have

$$H(X|Y) = \int H(X|y)p(y)$$

with $H(X|y) = \lim_{\epsilon \to 0^+} H(X|y-\epsilon \leq Y \leq y+\epsilon)$.

Since the total number of records N in the joint distribution of X and Y is finite, the number of values in $[y-\epsilon, y+\epsilon]$ approaches 1

as $\epsilon \to 0^+$. As a consequence, we likely only have one tuple (x, y) to compute $H(X|y)$, i.e., $H(X|y)$ vanishes. Hence, $H(X|Y)$ becomes 0! This problem, named the empty space issue [17], also happens when X is either real-valued or discrete.

EXAMPLE 3. *Suppose that the joint distribution of X (categorical) and Y (real-valued) is as follows:*

X	orange	red	orange	red	orange	red
Y	1.00	1.01	2.00	2.01	4.00	4.01

Sticking to the exact formula of $H(X|Y)$, we have $H(X|Y) = \frac{1}{6}H(X|Y = 1.00) + \frac{1}{6}H(X|Y = 1.01) + \frac{1}{6}H(X|Y = 2.00) + \frac{1}{6}H(X|Y = 2.01) + \frac{1}{6}H(X|Y = 4.00) + \frac{1}{6}H(X|Y = 4.01) = 0$. Thus, $\textsc{Corr}(X, Y) = H(X) - H(X|Y) = 1 - 0 = 1$. *This result is not accurate since it is due to small mismatches in the values of Y—a scenario which is common for real-valued columns. A more correct computation of $H(X|Y)$ is given in Example 4.*

To overcome the issue, we propose to construct a histogram for Y. This is to increase the likelihood that we have enough points for meaningful computation. Here, any histogram construction method can be used, e.g., the equal-frequency method. One can also apply more sophisticated methods such as the one we are going to propose here. In a nutshell, we search for the histogram of Y that minimizes $H(X|Y)$, i.e., maximizes $\textsc{Corr}(X, Y)$, by dynamic programming. The search can be done efficiently by considering a reduced set of cut points while still guaranteeing high quality. For more details on how to achieve this, see [28]. Using this method, we can avoid setting a fixed number of histogram bins. Hence, we achieve a histogram that closely reflects the true distribution of Y, i.e., under- and over-fitting are avoided. The detail of our histogram search is available on our website.[2]

By means of histogram construction, we can overcome the empty space issue and obtain a reliable estimation of $H(X|Y)$.

EXAMPLE 4. *Continuing Example 3. To simplify our illustration, we here assume that the equal-frequency technique is applied where the number of bins is 3. Note, however, that in reality our method does not require fixing the number of bins in advance.* DECOREL *produces following bins for Y: $b_1 = [1.00, 2.00)$, $b_2 = [2.00, 4.00)$, and $b_3 = [4.00, 4.01]$. We have $H(X|Y) = \frac{1}{3}H(X|b_1) + \frac{1}{3}H(X|b_2) + \frac{1}{3}H(X|b_3) = 1$. Thus,* $\textsc{Corr}(X, Y) = H(X) - H(X|Y) = 1 - 1 = 0$. *We note that this result makes sense since X is randomly distributed in any bin of Y, i.e., knowing the value (bin) of Y tells us nothing about X.*

Note that when forming the histogram for Y, we place its NULL values (caused by the left outer join) into the same histogram bin. We expect the values of X in this bin to be very disparate. In other words, the NULL bin is expected to have high entropy/*CE* with respect to X. Hence, $\textsc{Corr}(X, Y)$ gets smaller. This helps us to achieve our goal of punishing unmatched values between the two columns.

Case 2: Y is discrete

The usual way would be to compute $\textsc{Corr}(X, Y)$ as when Y is categorical. However, this approach may be prohibitive if Y has a large number of distinct values, for example, Y is an auto-generated primary key. To boost efficiency, similarly to Case 1, we also construct a histogram for Y. The rests thus are similar to Case 1.

[2] http://www.ipd.kit.edu/~nguyenh/ssdbm14-appendix.pdf

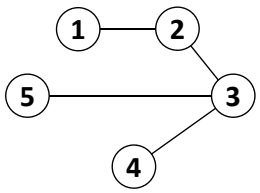


Figure 1: Example of an independence graph for a group $g = \{C_1, C_2, C_3, C_4, C_5\}$.

Case 3: Y is categorical

Since categorical columns have straightforward distributions and thus information theoretic calculations, we omit further details of how to compute $\textsc{Corr}(X, Y)$.

5. FROM PAIRWISE TO MUTUAL CORRELATION

Since our goal is to detect groups of any size, we need to compute mutual correlations efficiently. In this section, we explain how to find mutual correlations through pairwise correlations. Our results are based on the theory of independence graphs [32].

5.1 Independence Graph

Consider a group $g = \{C_i\}_{i=1}^d$. Using CORR, we compute its pairwise correlation scores. Then, following [22, 31], we construct an independence graph $\mathcal{G}$ for g. In short, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V} = g = \{C_i\}_{i=1}^d$ (i.e., each column is a node) is undirected, acyclic, connected, and $(C_i, C_j) \notin \mathcal{E} \Leftrightarrow C_i \perp C_j \mid \mathcal{V} \setminus \{C_i, C_j\}$. That is, two columns C_i and C_j not connected by an edge are regarded as conditionally independent given all other columns $\mathcal{V} \setminus \{C_i, C_j\}$.

EXAMPLE 5. *Figure 1 depicts a possible independence graph for a group $g = \{C_1, C_2, C_3, C_4, C_5\}$. One can see that this graph is undirected, acyclic, and connected. Further, since there is no edge connecting C_1 and C_5, they are considered to be conditionally independent given columns C_2, C_3, and C_4. Therefore, $C_1 \perp C_5 \mid \{C_2, C_3, C_4\}$.*

We have the following result for the total correlation:

THEOREM 3. *$T(C_1, \ldots, C_d)$ is equal to* $\sum_{(C_i, C_j) \in \mathcal{E}} I(C_i, C_j)$.

PROOF. From [22], $p(C_1, \ldots, C_d)$ is equal to:

$$\frac{\prod_{(C_i, C_j) \in \mathcal{E}} p(C_i, C_j)}{\prod_{C \in \mathcal{V}} p(C)^{deg(C)-1}}$$

where $deg(C)$ denotes the degree of C in $\mathcal{G}$.

W.l.o.g., we assume that $\{C_i\}_{i=1}^d$ are all real-valued. Then we

have that $T(C_1, \ldots, C_d)$ is equal to:

$$\int p(\{c_i\}_{i=1}^d) \log \frac{p(\{c_i\}_{i=1}^d)}{\prod_{i=1}^d p(c_i)} dc_1 \cdots dc_d$$
$$= \int p(\{c_i\}_{i=1}^d) \log \frac{\prod_{(C_i,C_j)\in\mathcal{E}} p(c_i, c_j)}{\prod_{C\in\mathcal{V}} p(c)^{deg(C)}} dc_1 \cdots dc_d$$
$$= \sum_{(C_i,C_j)\in\mathcal{E}} \int p(\{c_i\}_{i=1}^d) \log \frac{p(c_i, c_j)}{p(c_i)p(c_j)} dc_1 \cdots dc_d$$
$$= \sum_{(C_i,C_j)\in\mathcal{E}} \int p(c_i, c_j) \log \frac{p(c_i, c_j)}{p(c_i)p(c_j)} dc_i dc_j$$
$$= \sum_{(C_i,C_j)\in\mathcal{E}} I(C_i, C_j)$$

□

EXAMPLE 6. *Continuing Example 5. Following Theorem 3:* $T(C_1, C_2, C_3, C_4, C_5) = I(C_1, C_2) + I(C_2, C_3) + I(C_3, C_4) + I(C_3, C_5)$.

Theorem 3 shows a decomposition of total correlation into a sum of mutual information terms. In other words, it tells us that we can find mutual correlations by means of pairwise correlations.

We note that being able to estimate mutual correlations by pairwise correlations is insufficient. In particular, directly adopting the result of Theorem 3, a naive solution would be as follows: For each group $g = \{C_i\}_{i=1}^d$, one measures the mutual information score of each column pair. Then, one constructs the maximum spanning tree to obtain the independence graph of g [22, 31]. One estimates the total correlation score of g using Theorem 3. Finally, one picks groups with largest scores. While this solution is straightforward, it suffers from two issues. First, as mentioned before, mutual information is not a reliable correlation measure for real-valued data. Second, the solution requires to examine each and every candidate group. However, the number of groups is still exponential in the number of columns. We address these issues next.

5.2 Our Approach

We propose to mine groups g where each member column is correlated with *most* of the columns in g. We name such a group an approximate group. Our intuition behind this design choice is that by enforcing the requirement of almost perfect pairwise correlation among columns of g, the edge weights of its graph $\mathcal{G}$ (nodes are its columns and edge weights are pairwise correlation scores) are large. Hence, the sum of edge weights of its maximum spanning tree is likely large, which, according to our analysis in Section 5.1, signifies a large total correlation score, i.e., mutual correlation.

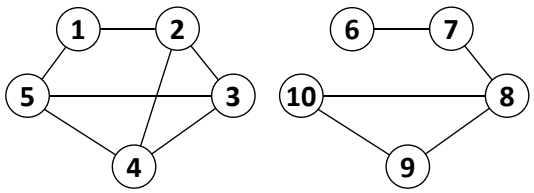
We support this observation by another result of ours as follows. W.l.o.g., consider a group $g = \{C_i\}_{i=1}^d$ where every C_i is correlated with every other C_j. We define C_i and C_j to be correlated iff $\text{CORR}(C_i, C_j)$ and $\text{CORR}(C_j, C_i)$ are large. We discuss how to decide if a correlation score produced by CORR is large in Section 6. We have:

CLAIM 1. *$\{C_i\}_{i=1}^d$ are likely mutually correlated under different correlation measures.*

Claim 1 essentially states that if every two columns of a group are correlated, then all of its columns are likely mutually correlated. To verify this, we need the following result:

THEOREM 4. *It holds that:*

$$T(C_1, \ldots, C_d) \geq \sum_{i=1}^{d-1} H(C_{i+1}) - H(C_{i+1}|C_i) \quad .$$



(a) g_1: An approximate group (b) g_2: *Not* an approximate group

Figure 2: Example of approximate groups ($\delta = 0.5$).

PROOF. From Definition 2, we have

$$T(C_1, \ldots, C_d) = \sum_{i=1}^{d-1} H(C_{i+1}) - H(C_{i+1}|C_1, \ldots, C_i) \quad .$$

Since conditioning reduces entropy [6], we have

$$H(C_{i+1}|C_1, \ldots, C_i) \leq H(C_{i+1}|C_i) \quad .$$

Thus, we arrive at the result. □

We now explain our claim. In particular, if C_i and C_{i+1} are correlated, $p(C_i, C_{i+1})$ deviates from $p(C_i)p(C_{i+1})$ (see Lemma 1). Thus, $H(C_{i+1}) - H(C_{i+1}|C_i)$, which equals to the Kullback-Leibler divergence of $p(C_i, C_{i+1})$ and $p(C_i)p(C_{i+1})$ [6], is high. Following Theorem 4, we conclude that $T(C_1, \ldots, C_d)$ is high. This also means that the difference between $p(C_1, \ldots, C_d)$ and $p(C_1) \cdots p(C_d)$ is high with respect to the Kullback-Leibler divergence. Hence, $\{C_i\}_{i=1}^d$ are mutually correlated, not just pairwise correlated. Since many other correlation measures define mutual correlation based on the difference between the joint distribution and the product of marginal distributions [27], $\{C_i\}_{i=1}^d$ are also likely mutually correlated under those correlation measures. □

In the above, we have shown that for a given group, if every two of its columns are correlated, then its columns are likely mutually correlated. In other words, this group is likely a group of correlated columns. However, real-world data tends to contain noise making perfect pairwise correlation between columns of any group not always happen. In fact, our preliminary empirical analysis points out that sticking to the requirement of perfect pairwise correlation, we would end up only with small groups, e.g., those containing two to three columns. To address this issue, we go for a fault-tolerant solution. More specifically, we focus on groups g where each member column is correlated with *most* of the columns in g. We will show in Section 6 that such groups permit efficient mining. In addition, they yield very high quality in our experiments. Below we provide a formal definition of approximate groups.

DEFINITION 6. *(*APPROXIMATE GROUP*)* $g = \{C_i\}_{i=1}^d \subset \mathcal{C}$ *with $d \geq 2$ is an approximate group of correlated columns iff (a) C_i is correlated to at least $\lceil \delta \cdot (d-1) \rceil$ columns in g ($0 < \delta \leq 1$), and (b) no proper superset of g is an approximate group of correlated columns.*

In Definition 6, we enforce the maximality requirement of approximate groups in order to eliminate redundancy. We note that DECOREL is not constrained to this notion of approximate groups. Depending on the application scenario, one could go for a tighter notion, e.g., perfect pairwise correlation by setting $\delta = 1$.

EXAMPLE 7. *Figures 2(a) and 2(b) depict pairwise correlations of two toy groups $g_1 = \{C_1, C_2, C_3, C_4, C_5\}$ and $g_2 =$*

$\{C_6, C_7, C_8, C_9, C_{10}\}$, *respectively. Note that they do* ***not*** *depict independence graphs. In both figures, the convention is that two nodes are connected iff the corresponding columns are correlated. For instance, in group* g_1, C_1 *and* C_2 *are correlated. Assume that* $\delta = 0.5$. *According to Definition 6,* g_1 *is an approximate group. On the other hand,* g_2 *does not meet the condition (a) of Definition 6 since* C_6 *is only correlated to one column* C_7 *(the minimum vertex degree is* $\lceil 0.5 \cdot (5-1) \rceil = 2$*). Hence,* g_2 *is not an approximate group. We note that* $\{C_1, C_2, C_3, C_5\}$, $\{C_2, C_3, C_4\}$, *and* $\{C_3, C_4, C_5\}$ *are also approximate groups. However, with the maximality requirement, these groups will not be output since they are redundant with respect to* g_1.

To solve our problem of detecting groups of correlated columns, for efficiency reasons we mine approximate groups instead. From now on, with groups we mean approximate groups.

6. GROUP DISCOVERY

To mine groups, we have to address three questions. First, since CORR produces real-valued correlation scores, how can we decide if a score is large enough to signify that two columns are correlated? Second, the search space is still potentially exponential to the total number of columns. So how can we efficiently mine groups? Third, the number of groups output may be too large for subsequent processing steps, e.g., when users want to manually inspect the groups for adjusting the query optimizer. Hence, how can we produce a succinct set of groups that facilitates post-analysis? We answer all questions in this section.

6.1 Thresholding Correlation Scores

$\text{CORR}(X, Y)$ is upper-bounded by the entropy/cumulative entropy of X, which in turn is dependent on its value domain. Further, CORR is asymmetric. Thus, we propose to threshold the correlation scores in each individual column. In particular, let $\mathcal{N}(C)$ be the set of *neighboring columns* of C, i.e., those that are most correlated to C. We identify $\mathcal{N}(C)$ as follows.

Let $\{(C_i, \text{CORR}(C, C_i))\}_{i=1}^{M}$ be the set of correlation scores between C and each other column C_i. W.l.o.g., we assume that the set is sorted in descending order w.r.t. $\text{CORR}(C, C_i)$. We define

$$ind(C) = argmax_{i \in [1, M-1]} \frac{\text{CORR}(C, C_i) + 1}{\text{CORR}(C, C_{i+1}) + 1} \quad .$$

That is, if we plot the correlation score spectrum against the index $i \in [1, M]$, $ind(C)$ is where there is the biggest jump in the correlation score ratio. We add 1 to both the numerator and the denominator to remove the impact of small correlation scores, which may cause the respective ratios of scores to be unusually large. Let the cutoff threshold be $th(C) = \text{CORR}(C, C_{ind(C)})$. We set

$$\mathcal{N}(C) = \{C_i : i \in [1, M] \wedge \text{CORR}(C, C_i) \geq th(C)\} \quad .$$

Our thresholding scheme has its intuition from eigenvalue spectrum analysis [11], which shows that using score ratios effectively separates small and zero scores from large ones in a score spectrum, without introducing any additional parameter.

EXAMPLE 8. *In Figure 3, we plot the correlation score spectrum of NATIONKEY in* TPC-H. *We can see that the scores form three distinct clusters. Intuitively, a cutoff should be placed at rank 12. Using our thresholding scheme,* DECOREL *correctly sets* $ind(NATIONKEY)$ *to 12.*

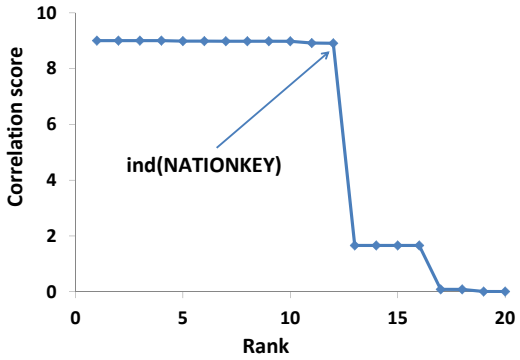


Figure 3: Correlation score spectrum of NATIONKEY in *TPC-H*. According to our method, $ind(NATIONKEY) = 12$**.**

6.2 Group Mining

We form an undirected column graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each node $C \in \mathcal{V}$ is a database column. An edge $e \in \mathcal{E}$ exists between two nodes C_i and C_j $(i \neq j)$ iff $C_i \in \mathcal{N}(C_j)$ and $C_j \in \mathcal{N}(C_i)$. The resulting $\mathcal{G}$ captures pairwise correlations of columns.

Given a subset of vertices $S \subseteq \mathcal{V}$, we define the subgraph $\mathcal{G}(S)$ induced by S as the one with vertex-set being S, and edge-set being edges of $\mathcal{E}$ whose both end-points are in S. In fact, the groups in Definition 6 correspond to quasi-cliques in $\mathcal{G}$.

DEFINITION 7. *(*QUASI-CLIQUE*) A subset* $S \subseteq \mathcal{V}$ *with at least two vertices is a* δ*-quasi-clique (or quasi-clique for short) of* $\mathcal{G}$ *iff: (a) every vertex* $v \in S$ *has a degree in* $\mathcal{G}(S)$ *of at least* $\lceil \delta \cdot (|S| - 1) \rceil$, *and (c) no proper superset of* S *is a* δ*-quasi-clique.*

Considering Definition 6 of approximate groups, the requirement of minimum vertex degree ensures that each column of the group is correlated to most of the other columns. Also, the maximality requirement of S addresses the maximality of the group.

Hence, we mine groups forming quasi-cliques in $\mathcal{G}$. If $\delta = 1$, we end up with cliques, which would yield groups of columns with stronger correlations. However, we expect $\mathcal{G}$ to be sparse. Searching for cliques, we might end up only with groups corresponding to two end-points of the same edges. On the other hand, we still need to set δ high enough to ensure the compactness of the groups. Following the proposal in [19], we set $\delta = 0.5$. By this, we ensure that the quasi-cliques, and hence groups, are connected tightly. In fact, the problem of mining all quasi-cliques is NP-hard [10].

Since $\mathcal{G}$ tends to be sparse, we address the NP-hardness complexity of the problem by practical algorithms that have good performance on real-world data. In particular, we rely on several pruning rules to efficiently explore the search space [19]. For instance, when $\delta = 0.5$, the shortest path between any two vertices of a quasi-clique contains at most two edges. We use this observation to reduce the vertices which can be used to extend a candidate quasi-clique. Another pruning is based on the upper and lower bounds of the number of vertices that can be added to a candidate quasi-clique concurrently to form a larger quasi-clique. By these pruning rules, we are able to eliminate candidate extensions of existing quasi-cliques. Our experiments show DECOREL to achieve both higher quality and better scalability than existing methods.

After mining groups, we may achieve very many groups, which hinder post-analysis. Our goal is to obtain a succinct set of groups that can be inspected manually. Therefore, we propose to merge similar groups in the subsequent step.

EXAMPLE 9. *Assume that after the group mining phase, two of the groups that we obtain are* $g_3 = \{C_1, C_2, C_3, C_4\}$ *and* $g_4 = \{C_1, C_3, C_4, C_5\}$. *Since they have many dimensions in common, i.e., they bring about similar information, it would make sense to merge them to create the larger group* $\{C_1, C_2, C_3, C_4, C_5\}$.

6.3 Group Merging

Let $\{g_i\}_{i=1}^m$ be the set of groups found. Further, let $\{C_j\}_{j=1}^l$ be the set of columns such that for each C_j, there exists g_i that contains C_j. For each C_j, we construct a binary vector $u = (u_1, \ldots, u_m)$ with $u_i = 1$ if and only if $C_j \in g_i$. Thereby, we obtain a binary data set $\mathcal{B}$ with l records and m dimensions. We merge groups by dividing dimensions of $\mathcal{B}$ into clusters such that each cluster contains similar dimensions. Each such cluster constitutes one final group. We aim at achieving the task without having to define any distance function among the dimensions of $\mathcal{B}$. Thus, we apply the merge algorithm proposed in [20] which uses the Minimum Description Length (MDL) principle.

Given a set of models $\mathcal{M}$, MDL identifies the best model $M \in \mathcal{M}$ as the one that minimizes $L(\mathcal{B}, M) = L(M) + L(\mathcal{B} \mid M)$, in which $L(M)$ is the length in bits of the description of the model M, and $L(\mathcal{B} \mid M)$ is the length of the description of the data $\mathcal{B}$ encoded by M. That is, MDL helps select a model that yields the best balance between goodness of fit and model complexity.

In our problem, each model M corresponds to a merge of groups, i.e., a clustering of attributes of $\mathcal{B}$. Our goal is then to discover the clustering that minimizes the total encoding cost $L(\mathcal{B}, M)$. However, since the search space is $O(2^m)$ and unstructured, we utilize a heuristic algorithm. In particular, we start with each dimension of $\mathcal{B}$ forming its own cluster. Then, step-by-step we pick two clusters whose merge leads to the largest reduction in the total encoding cost, and merge them. This practice ensures the two most similar clusters to be merged at each step [20]. The algorithm terminates when either there are no more clusters to merge, or when the current step does not reduce the total encoding cost any more.

Our merge of groups guarantees completeness and minimizes redundancy. That is, our group merging guarantees that its output groups capture all the groups produced by the graph mining step. This is because MDL produces a *lossless compression*. Therefore, the original set of groups are compressed while ensuring no information loss. In addition, our algorithm selects the clustering of groups that minimizes the overall compression cost. Thus, if a clustering contains two very similar groups, our algorithm would not pick it. This is because the merge of two groups can result in a better clustering with a lower encoding cost. Hence, redundancy is minimized, i.e., DECOREL manages to provide a succinct output that facilitates manual inspection and post-analysis. Not only that, by group merging, DECOREL discovers correlated columns that would go undetected otherwise, e.g., CO_2 concentration and amount of drinking water on *Climate* database (see Section 8.2).

We highlight that this merge step only merges groups with high overlap. This means groups with low overlap remain separate. Thus, DECOREL still detects overlapping groups representing different semantics of columns. We ensure this by detecting overlapping groups of correlated columns by first mining quasi-cliques in the column graph $\mathcal{G}$, and then merging groups.

So far we have described all steps of DECOREL. Due to space issues, we omit the detailed analysis of its overall time complexity. A similar analysis, however, can be found in [23].

7. THEORETICAL COMPARISON

In this section, we theoretically compare DECOREL against the state of the art method for grouping relational columns [34]. This method, named ECLUS, uses a distribution test based on Earth Mover's Distance (EMD) [18] to assess relationships of columns. Our purpose here is to show that DECOREL is more general than ECLUS. Hence, we are able to discover column relationships that ECLUS misses. We now review the background of ECLUS before presenting our analysis.

7.1 Review: Distribution Test in ECLUS

ECLUS uses EMD to quantify the (dis-)similarity between the marginal distributions of any two columns, e.g., C_1 and C_2. The lower the distance, the more similar C_1 and C_2 are.

Let $|C|$ be the number of distinct values of column C. To compute EMD, ECLUS first forms pmfs on the sets of values of C_1 and C_2 by assigning a probability mass of $1/|C_1|$ and $1/|C_2|$ to each value of C_1 and C_2, respectively. Then it sorts the union of the values of C_1 and C_2 (in lexicographical order for strings, and in numerical order for numerical values) and computes the EMD of two distributions of the *ranks*. Suppose that the distribution of C_1 is $p = \{(x_1, p_1), \ldots, (x_{|C_1|}, p_{|C_1|})\}$ and the distribution of C_2 is $q = \{(y_1, q_1), \ldots, (y_{|C_2|}, q_{|C_2|})\}$. Here, x_i and y_j are ranks, and p_i and p_j are their masses, respectively. ECLUS then instantiates $EMD(C_1, C_2)$ to $EMD(p, q)$.

For efficiency purposes, the ranks may be further discretized into quantile histograms, and the EMD is applied on two such histograms accordingly. Zhang et al. [34] show that the EMD test can discover various column relationships: (a) foreign key and primary key constraints, (b) two foreign keys referring to the same primary key, (c) a column in a view and its correspondence in the base table, (d) two columns in two different views but originating from the same column in the base table, and (e) two columns without any explicit association but semantically related through a third column.

However, their method only discovers groups of columns of the same data type and of overlapping value domains. This is too restrictive for real-world databases.

7.2 Correlation Test is More General

Our claim is that two columns passing the EMD test, i.e., having a low EMD value, are very likely correlated. On the other hand, having have a high EMD value does not imply that they are uncorrelated. In other words, we have:

CLAIM 2. *Our correlation test is more general than EMD test.*

To show this, we utilize the statistical intuition of EMD. In fact, EMD can be regarded as the Wasserstein metric [25]. Applying the Wasserstein metric to columns C_1 and C_2, their EMD becomes:

$$\min_F \{E_F(|X - Y|) : X \sim p, Y \sim q, (X, Y) \sim F\} \quad .$$

That is, $EMD(C_1, C_2)$ equals to the minimum of the expected difference between their ranks (X and Y, respectively), taken over all possible joint probability distributions F of their ranks such that the marginal distributions of F are p and q. Assume that $|C_1| = |C_2| = n$ and $\{x_i\}_{i=1}^n$ and $\{y_i\}_{i=1}^n$ are sorted in ascending order. It holds that [18]:

$$EMD(C_1, C_2) = \frac{1}{n} \sum_{i=1}^{n} |x_i - y_i| \quad .$$

Thus, $EMD(C_1, C_2) = 0$ iff $x_i = y_i$ for every $i \in [n]$. This implies a perfect 1-to-1 mapping between X and Y, and hence, between the values of C_1 and C_2. Thus, both $\text{CORR}(C_1, C_2)$ and $\text{CORR}(C_2, C_1)$ are likely large, i.e., C_1 and C_2 are correlated.

This observation also holds for general cases when $|C_1| \neq |C_2|$. That is, the lower the EMD score, the less cost of transforming p

to q, the easier it is to define a 1-to-1 mapping between X and Y. Thus, the more likely it is that C_1 and C_2 are correlated.

All in all, a low EMD score tends to correspond to a high correlation score. Thus, if two columns have a low correlation score, they tend to have a high EMD score, i.e., they are unrelated under the EMD test. However, a high EMD does not say anything about the correlation. This is because EMD does not assess joint distributions while correlation analysis is involved in both marginal and joint distributions (see Section 3). As a result, even if two columns have similar marginal distributions, they may be uncorrelated [24]. For instance, consider two numerical columns whose joint distribution contains uniformly distributed data, i.e., they are not correlated. If we project the data onto each of the two columns, we obtain two similar marginal (uniform) distributions. On the other hand, two columns can be correlated while having very different marginal distributions. Thus, by performing correlation analysis, we can detect not only the relationships that an EMD test can find but also the relationships that such a test cannot cover. □

Hence, when the basic schema information is available, ECLUS is restrictive and misses correlations among columns with not only different value domains, but also different data types.

EXAMPLE 10. *Our experiments reveal that on* TPC-H*,* ECLUS *cannot discover the correlation between the order date and the corresponding order status. This is because the two columns have very different value domains. On* Energy*,* ECLUS *also misses the correlation between the energy consumption indicators of each building (real-valued) and its location (categorical).* DECOREL *nevertheless handles these scenarios.*

In sum, DECOREL assesses the relationships of columns based on their joint distributions and marginal distributions. Thus, it does not require columns to have similar values or data types. This is an important contribution of ours to the area of schema extraction.

8. EXPERIMENTS

In this section, we report our empirical study on the performance of DECOREL. Our objectives are to assess (a) the quality of groups produced by DECOREL, (b) the scalability of DECOREL with the database size as well as the number of columns, and (c) the succinctness of its output. To compute $\text{CORR}(X, Y)$ when Y is numerical, we search for the histogram of Y that maximizes the score $\text{CORR}(X, Y)$ (see Section 4.3.2).

We use both synthetic and real-world databases. In particular, we generate a synthetic database *SYNTH* containing several relations and columns of different data types (real-valued, discrete, and categorical). We model it according to the *TPC-H* benchmark. However, we additionally embed several correlations, ranging from simple linear to complex non-linear correlations. We use *SYNTH* to quantitatively assess the quality of the correlations detected by DECOREL. Further, we use the synthetic *TPC-H* benchmark database itself with scale factor 1 (i.e., about 1GB data).

For real-world data, we use the *Energy* database, which consists of several relations, e.g., Institution, Building, Consumption. It records the energy consumption patterns of the KIT university campus. As mentioned in Section 1, *Energy* contains categorical, discrete, and real-valued columns. Thus, it is a good real-world testbed for our evaluation. Another real-world database is *Climate*, which contains indoor climate and energy consumption indicators of a building in Frankfurt, Germany. *Climate* contains real-valued columns only. In addition, we include *Census*, a publicly available real-world data set from the UCI Machine Learning Repository. Table 2 summarizes the characteristics of all databases used.

Data	Tables	Rows	Columns	Data Types
SYNTH	8	50,000	40	r, d, c
SYNTH2	8	50,000	40	r, d
TPC-H	8	8,661,245	61	r, d, c
Energy	6	1,486,718	63	r, d, c
Climate	1	35,601	251	r
Census	1	95,130	41	d, c

Table 2: Database characteristics. With data types: 'r' real-valued, 'd' discrete, and 'c' categorical

We compare DECOREL to three state of the art techniques that also group columns: First, ECLUS groups columns using EMD [34]. Second, 4S groups columns (numerical only) using a quadratic measure of dependency [23]. Finally, CORDS groups columns into *pairs* using the χ^2 test [9]. Since CORDS is not designed for combining those pairs into larger groups of columns, we apply our group mining algorithm on its output pairs.

8.1 Quantitative Assessment of Groups

Assessment based on Precision-Recall: We only use databases where we have prior knowledge on its correlations. In particular, we use *SYNTH* with categorical and numerical (discrete and real-valued) columns. We also generate its variant, named *SYNTH2*, with only numerical (discrete and real-valued) columns. In addition, we use *Climate* where correlations are pre-identified by domain experts. *Climate* contains real-valued columns only. Our goal is to assess DECOREL under different settings. Note that in all three databases, the known correlations are involved in columns of different data types and/or different value domains. Further, the ground truth of each database contains overlapping groups of correlated columns. For instance, two overlapping groups of *Climate* are: a group containing indoor temperatures of rooms located in the same section of the building, and a group containing indoor climate and energy indicators (e.g., temperature, CO_2 concentration, heating consumption) of such a room.

The results are in Table 3. We can see that DECOREL performs very well, outperforming all of its competitors. In contrast, ECLUS has low accuracy since it clusters columns of different data types and non-overlapping value domains into separate groups although these columns are correlated. Moreover, ECLUS produces disjoint groups and hence, breaks overlapping groups of correlated columns. For example, DECOREL discovers two overlapping groups of *Climate* mentioned above while ECLUS discovers only the first group.

CORDS in turn uses the χ^2 test which fits better to categorical and discrete data. As a consequence, its performance deteriorates on *Climate* which contains real-valued columns only.

4S is inapplicable to *SYNTH* since 4S is not designed to handle categorical columns. On the other two data sets, 4S approximates correlation scores using AMS Sketch [2]. To improve accuracy, the number of sketches should be large. However, 4S keeps this number small for efficiency reasons and trades quality for efficiency.

Overall, compared to all of its competitors, we see that DECOREL best detects overlapping groups of correlated columns with heterogeneous data types and value domains.

Assessment based on adjustment factor: Here, we use the adjustment factor to assess the results of DECOREL on *all* databases. We define the adjustment factor of a group $g = \{C_i\}_{i=1}^{d}$ as

$$af(g) = \frac{\prod_{i=1}^{d} |C_i|}{|C_1, \ldots, C_d|}$$

		DECOREL	ECLUS	CORDS	4S
SYNTH	Prec.	1.00	0.56	0.75	-
	Rec.	1.00	0.66	0.72	-
SYNTH2	Prec.	1.00	0.54	0.73	1.0
	Rec.	1.00	0.67	0.74	0.99
Climate	Prec.	0.91	0.72	0.68	0.85
	Rec.	0.93	0.74	0.67	0.87

Table 3: Precision and Recall on synthetic data and *Climate*.

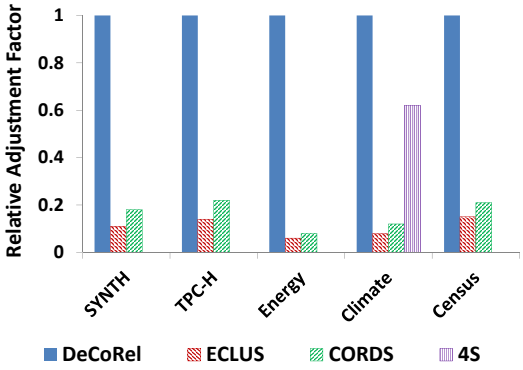


Figure 4: Relative adjustment factor compared to DECOREL.

where $|C_i|$ is the number of distinct values in C_i and similarly for $|C_1, \ldots, C_d|$. Note that if the columns belong to multiple relations, $|C_1, \ldots, C_d|$ is computed by applying full outer join of the involved relations, as done in [33].

Intuitively, the larger the adjustment factor of a group, the more correlated its columns are. We define the adjustment factor of a method producing groups $\{g_j\}_{j=1}^{n}$ as $\frac{1}{n}\sum_{j=1}^{n} af(g_j)$, i.e., the average of adjustment factors of its groups. Again, the larger the adjustment factor of a method, the better the method is in finding groups of correlated columns. In fact, a similar notion of adjustment factor was used in [9] to rank pairs of correlated columns. Further, it has been suggested that the adjustment factor has an impact on the selectivity estimates of optimizers [9, 31]. Thus, one can use the adjustment factor as an implicit measure for query optimizers.

The relative adjustment factors of all methods in comparison to DECOREL are in Figure 4. Recall that 4S is only applicable to the numerical *Climate* database. The results show DECOREL to achieve the best results, outperforming its competitors up to an order of magnitude. This suggests that DECOREL better discovers groups of correlated columns where the joint distributions deviate more profoundly from the product of the marginal distributions.

As mentioned above, the adjustment factor has an impact on selectivity estimates of optimizers. Thus, by being able to discover dependable groups of correlated columns, one could use DECOREL to reliably identify important column correlations for improving query optimizers. This is one of several possible applications of the groups detected by DECOREL.

8.2 Qualitative Assessment of Groups

We now explore in detail the groups discovered by DECOREL to gain more insight into its performance. We limit our discussion to *TPC-H* and real-world databases where we do not have full knowledge of the hidden correlations.

***TPC-H*:** DECOREL correctly identifies all correlations which are involved in declared foreign key and primary key constraints, regardless of column data types, and place the respective columns into the same group. More than that, DECOREL is able to group columns with no explicit relationships but semantically correlated. For instance, DECOREL puts ORDERDATE and ORDERSTATUS into the same group. In fact, [34] has pointed out that the order date and the corresponding order status are correlated. Since these two columns have totally different value domains, ECLUS cannot recognize their relationship. CORDS in turn tends to place correlated columns where at least one column is real-valued into separate groups. Intuitively, there is a correlation between the total price of an order and the respective customer name. CORDS however fails to detect it. 4S in turn is inapplicable to *TPC-H* since it only handles numerical columns. In contrast, DECOREL successfully groups correlated columns without being constrained to their data types as well as value domains.

***Climate*:** One of the groups identified by DECOREL reflects a correlation among the air temperature supplied to the heating system, the temperature of the heating boiler, and the amount of heating it produces. While this relation is rather intuitive and expected, ECLUS does not detect it. This could be because the three measures have very different ranges of values: from 10 to 45 for the supplied air temperature, from 25 to 350 for the temperature of the heating boiler, and from 0 to 1100 for the amount of heating produced. ECLUS, by its design, places the three columns into different groups and hence, leaves their relationship undetected. CORDS also does not find this correlation, partly because CORDS uses a correlation measure that is more suited to categorical and discrete columns. In addition, CORDS computes correlation scores using only a sample randomly drawn from the data. We observe that the pairwise joint distributions of the three measures are rather skewed. Thus, a simple random sampling as employed by CORDS likely misses the bigger picture [16]. Another correlation reflected in the groups discovered by DECOREL but not by any other method, is among the electricity consumed, the amount of drinking water, and the amount of heating. ECLUS misses this correlation, again, because the columns involved have different value ranges. CORDS also cannot identify this correlation due to its correlation measure. 4S in turn does not detect it because of inaccuracy caused by sketching. Besides correlations that are known, DECOREL is capable of discovering interesting new correlations; one of which is among room temperature, CO_2 concentration, and amount of drinking water. Domain experts we collaborate with have not been aware of this relationship initially. They however identify it to be of their interest.

***Energy*:** DECOREL groups columns of energy consumption of buildings with columns related to time (e.g., date-time, semester season, holiday/working day). This is expected as, e.g., more energy is consumed on working days than on holidays. DECOREL also groups consumption columns with the characteristics of buildings, e.g., total area, total number of staff members, and location. Again, this grouping is intuitively accurate: the larger area and the more staff members a building has, the more energy it consumes; buildings located in the campus section where, say, large-scale physics experiments are usually carried out consume more energy than buildings in other locations. However, since these columns have different value domains and even data types, ECLUS puts them into different groups. The quality of groups detected by CORDS degrades greatly. This could stem from the fact that the consumption indicators are all real-valued. Hence, the statistical power of χ^2 test, applied to these columns, becomes weak.

***Census*:** We present some groups DECOREL produces, which are undetected by other methods. We believe them to be interesting

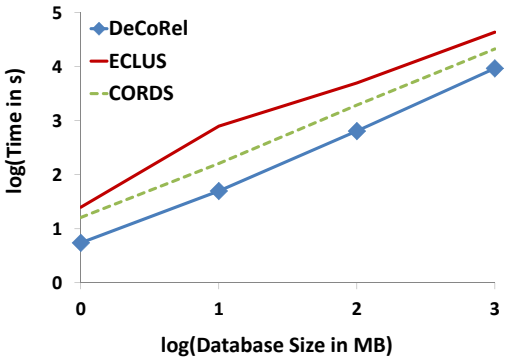


Figure 5: Scalability to database size using *TPC-H*.

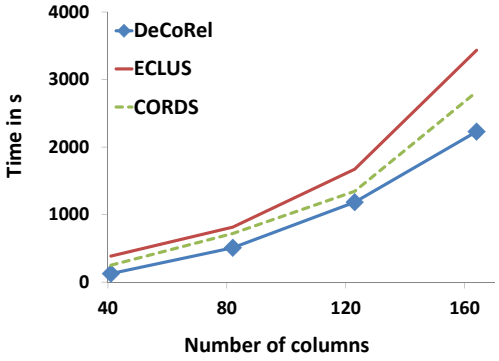


Figure 6: Scalability to the number of columns using *Census*.

from a data analysis point of view. Their details are as follows:

- weeks worked in year, education, reason for unemployment, hispanic origin, member of a labor union, wage per hour
- weeks worked in year, education, citizenship, country of birth father, country of birth mother, capital gains, income level
- marital status, age, education, dividends from stocks, detailed occupation recode

We note that some of the correlations in the groups above intuitively make sense, e.g., education and wage per hour. In addition, some of the remaining correlations, e.g., marital status, age, and education, agree with a previous study [21].

Overall, we see DECOREL to be capable of detecting column relationships regardless of their underlying data types as well as value domains. The relationships covered by DECOREL range from known ones (e.g., declared foreign key and primary key constraints) to novel ones. The latter can be exploited for, e.g., improving query optimizers and advanced data analysis.

8.3 Scalability

We assess scalability of DECOREL to the database size using the *TPC-H* benchmark and the number of columns using *Census*. For the former, we vary the scale factor and hence, effectively scale the number of tuples. For the latter, we scale the number of columns by appending the identical *Census* database for a number of times. The results are in Figures 5 and 6. For readability, we use a logarithmic scale on *both* axes for Figure 5.

We see that DECOREL scales linearly to the database size, i.e., the number of tuples. It has quadratic scalability to the number of columns. Moreover, DECOREL achieves better scalability than ECLUS in both tests. DECOREL also scales better than CORDS even though CORDS only works on a sample of the data. This

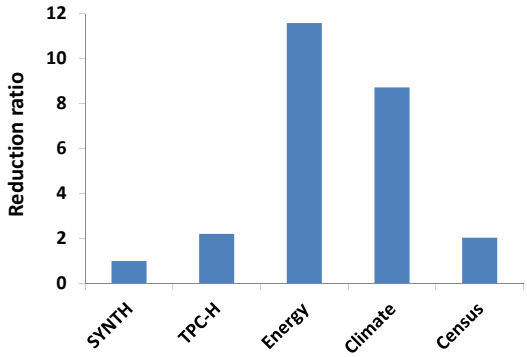


Figure 7: Reduction ratio of our group merging.

highlights the benefits of our efficient computation of pairwise correlations and our efficient mining of correlated groups. We are unable to display the runtimes of 4S since 4S is inapplicable to both *TPC-H* and *Census* due to their categorical columns. However, for numerical data such as *Climate*, we have observed that DECOREL shows runtimes similar to 4S. Thus, we can perform both categorical and numerical data assessment in a very scalable manner.

Overall, DECOREL is able to tackle the exponential search space of groups in polynomial time and hence, enables correlation analysis for large databases.

8.4 Succinctness of Output

We study the benefits of our MDL group merging. Our performance metric is the reduction ratio, i.e., $\frac{m}{m'}$ where m and m' are the number of groups before and after merging. The results are in Figure 7. We see that the group merging phase achieves up to an order of magnitude reduction ratio. This verifies our claim that DECOREL produces a succinct set of overlapping groups while still guaranteeing their quality (see Sections 8.1 and 8.2). By providing end users with a succinct set of groups, in practice DECOREL facilitates manual inspection and post-analysis which benefit advanced applications based on the knowledge derived from the groups.

9. CONCLUSIONS

Discovering overlapping groups of correlated columns in relational databases is a very important task with wide applications in many areas. However, it is very challenging due to (a) the lack of a correlation measure that is applicable to different data types (namely real-valued, discrete, and categorical), (b) the need to address mutual/multi-way correlations, (c) the requirement to allow for overlapping groupings of columns, (d) the search space of overlapping groups that is exponential in the number of columns, and (e) the redundant output that hinders post-analysis.

In this paper, we have proposed DECOREL, our solution towards addressing all the challenges. In particular, we show the feasibility of estimating mutual correlations by pairwise correlations. It lets us transform the search space into a column graph based on pairwise correlations. We compute pairwise correlations using CORR, our new correlation measure that works with both cumulative distribution functions and probability mass functions. This makes CORR applicable to different data types. Subsequently, we employ an efficient search algorithm that mines the column graph to find overlapping groups of correlated columns. To ensure succinctness of the groups output, we present an MDL-based method that achieves up to an order of magnitude size reduction ratio in practice. Lastly, we show that our method is more general than a state of the art

technique in schema extraction. Experiments on both synthetic and real-world databases show DECOREL to discover groups of higher quality than existing methods. Moreover, DECOREL scales better than its competitors with both the database size and the number of columns. This suggests that DECOREL is a very promising tool for large-scale correlation analysis on real-world databases.

In future work, we plan to study DECOREL with enterprise-scale databases, as well as extend it to handle databases with missing values. We also plan to apply the groups discovered by DECOREL in specific applications, such as query optimization. Another interesting direction is to exploit the asymmetric property of CORR for directed column relationship discovery.

Acknowledgments This work is supported by the German Research Foundation (DFG) within GRK 1194, by the Young Investigator Group program of KIT as part of the German Excellence Initiative, and by a Post-Doctoral Fellowship of the Research Foundation – Flanders (FWO).

10. REFERENCES

[1] B. Ahmadi, M. Hadjieleftheriou, T. Seidl, D. Srivastava, and S. Venkatasubramanian. Type-based categorization of relational attributes. In *EDBT*, pages 84–95, 2009.

[2] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC*, pages 20–29, 1996.

[3] P. Andritsos, R. J. Miller, and P. Tsaparas. Information-theoretic tools for mining database structure from large data sets. In *SIGMOD*, pages 731–742, 2004.

[4] P. Brown and P. J. Haas. BHUNT: Automatic discovery of fuzzy algebraic constraints in relational data. In *VLDB*, pages 668–679, 2003.

[5] C. H. Cheng, A. W.-C. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *KDD*, pages 84–93, 1999.

[6] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience New York, 2006.

[7] P. D. Grünwald. *The Minimum Description Length Principle*. MIT Press, 2007.

[8] A. Heise, J.-A. Quiané-Ruiz, Z. Abedjan, A. Jentzsch, and F. Naumann. Scalable discovery of unique column combinations. *PVLDB*, 7(4):301–312, 2013.

[9] I. F. Ilyas, V. Markl, P. J. Haas, P. Brown, and A. Aboulnaga. CORDS: Automatic discovery of correlations and soft functional dependencies. In *SIGMOD*, pages 647–658, 2004.

[10] D. Jiang and J. Pei. Mining frequent cross-graph quasi-cliques. *TKDD*, 2(4), 2009.

[11] X. Jiang, B. Mandal, and A. C. Kot. Complete discriminant evaluation and feature extraction in kernel space for face recognition. *Mach. Vis. Appl.*, 20(1):35–46, 2009.

[12] A. Jindal, E. Palatinus, V. Pavlov, and J. Dittrich. A comparison of knives for bread slicing. *PVLDB*, 6(6):361–372, 2013.

[13] J. Kang and J. F. Naughton. On schema matching with opaque column names and data values. In *SIGMOD*, pages 205–216, 2003.

[14] F. Keller, E. Müller, and K. Böhm. HiCS: High contrast subspaces for density-based outlier ranking. In *ICDE*, pages 1037–1048, 2012.

[15] H. Kimura, G. Huo, A. Rasin, S. Madden, and S. B. Zdonik. CORADD: Correlation aware database designer for materialized views and indexes. *PVLDB*, 3(1):1103–1113, 2010.

[16] G. Kollios, D. Gunopulos, N. Koudas, and S. Berchtold. Efficient biased sampling for approximate clustering and outlier detection in large data sets. *IEEE Trans. Knowl. Data Eng.*, 15(5):1170–1187, 2003.

[17] J. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer, New York, 2007.

[18] E. Levina and P. J. Bickel. The earth mover's distance is the mallows distance: Some insights from statistics. In *ICCV*, pages 251–256, 2001.

[19] G. Liu and L. Wong. Effective pruning techniques for mining quasi-cliques. In *ECML/PKDD (2)*, pages 33–49, 2008.

[20] M. Mampaey and J. Vreeken. Summarizing categorical data by clustering attributes. *Data Min. Knowl. Discov.*, 26(1):130–173, 2013.

[21] S. Mehta, S. Parthasarathy, and H. Yang. Toward unsupervised correlation preserving discretization. *IEEE Transactions on Knowledge and Data Engineering*, 17(9):1174–1185, 2005.

[22] E. Müller, I. Assent, R. Krieger, S. Günnemann, and T. Seidl. DensEst: density estimation for data mining in high dimensional spaces. In *SDM*, pages 173–184, 2009.

[23] H. V. Nguyen, E. Müller, and K. Böhm. 4S: Scalable subspace search scheme overcoming traditional apriori processing. In *BigData Conference*, pages 359–367, 2013.

[24] H. V. Nguyen, E. Müller, J. Vreeken, F. Keller, and K. Böhm. CMI: An information-theoretic contrast measure for enhancing subspace cluster and outlier detection. In *SDM*, pages 198–206, 2013.

[25] S. T. Rachev. The monge-kantorovich mass transference problem and its stochastic applications. *Theory Probab. Appl.*, 29(4):647–676, 1984.

[26] M. Rao, Y. Chen, B. C. Vemuri, and F. Wang. Cumulative residual entropy: A new measure of information. *IEEE Trans. on Information Theory*, 50(6):1220–1228, 2004.

[27] M. Rao, S. Seth, J.-W. Xu, Y. Chen, H. Tagare, and J. C. Príncipe. A test of independence based on a generalized correlation function. *Signal Processing*, 91(1):15–27, 2011.

[28] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti. Detecting novel associations in large data sets. *Science*, 334(6062):1518–1524, 2011.

[29] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. In *EDBT*, pages 168–182, 1998.

[30] Y. Sismanis, P. Brown, P. J. Haas, and B. Reinwald. GORDIAN: Efficient and scalable discovery of composite keys. In *VLDB*, pages 691–702, 2006.

[31] K. Tzoumas, A. Deshpande, and C. S. Jensen. Lightweight graphical models for selectivity estimation without independence assumptions. *PVLDB*, 4(11):852–863, 2011.

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

[33] X. Yang, C. M. Procopiuc, and D. Srivastava. Summary graphs for relational database schemas. *PVLDB*, 4(11):899–910, 2011.

[34] M. Zhang, M. Hadjieleftheriou, B. C. Ooi, C. M. Procopiuc, and D. Srivastava. Automatic discovery of attributes in relational databases. In *SIGMOD*, pages 109–120, 2011.

[35] X. Zhang, F. Pan, W. Wang, and A. B. Nobel. Mining non-redundant high order correlations in binary data. *PVLDB*, 1(1):1178–1188, 2008.

APPENDIX

A. PROOFS

PROOF OF THEOREM 1, FIRST PROPERTY. We consider the case when Y is real-valued. The proof for when Y is either discrete or categorical follows similarly.

We have $h(X|y) \geq 0$. Thus, its expectation is also non-negative. Consequently, $h(X|y)$ is zero if and only if $h(X|y) = 0$ for all $y \in dom(Y)$. This implies: $\forall y \in dom(y), \forall x \in dom(X)$,

$$P(X \leq x|y) \log P(X \leq x|y) = 0 \quad .$$

From $x \log x = 0$ if and only if $x = 0$ or $x = 1$, we arrive at $\forall y \in dom(Y), \forall x \in dom(X)$,

$$P(X \leq x|y) = 0 \vee P(X \leq x|y) = 1 \quad .$$

Let $A_y = \{x : P(X \leq x|y) = 1\}$ then $A_y \neq \emptyset$ and $\min A_y$ exists since $\lim_{x \to +\infty} P(X \leq x|y) = 1$ and $\lim_{x \to -\infty} P(X \leq x|y) = 0$. Hence, for every y, there exists a unique $x_y = \min A_y$ such that $p_{X|Y}(X = x_y|y) = 1$, i.e., X is a function of Y. □

PROOF OF THEOREM 1, SECOND PROPERTY. Again, we consider the case when Y is real-valued. The proof for when Y is either discrete or categorical follows similarly.

Given a fixed $x_0 \in dom(X)$, $P(X \leq x_0|y)$ is a number depending on y. So if we let $Z = P(X \leq x_0|Y)$, Z is then a random variable. According to the Jensen's inequality, it holds that

$$E_Z[Z \log Z] \geq E_Z(Z) \log E_Z(Z) \quad .$$

Thus:

$E_Y[P(X \leq x_0|Y) \log P(X \leq x_0|Y)]$

$\geq E_Y[P(X \leq x_0|Y)] \log E_Y[P(X \leq x_0|Y)]$

From $P(X \leq x_0|y) = \int_{-\infty}^{x_0} p_{X|Y}(x|y)dx$, we arrive at

$$E_Y[P(X \leq x_0|Y)] = \int_{dom(Y)} \int_{-\infty}^{x_0} p_{X,Y}(x, y)dxdy$$

which leads to: $E_Y[P(X \leq x_0|Y)] = P(X \leq x_0)$. Hence:

$$E_Y[P(X \leq x_0|Y) \log P(X \leq x_0|Y)] \geq P(X \leq x_0) \log P(X \leq x_0)$$

Replacing x_0 by x, integrating and negating both sides w.r.t. x, we obtain:

$$h(X|Y) = E_Y[h(X|Y)] \leq h(X) \quad .$$

Since $g(w) = w \log w$ is strictly convex, equality holds if and only if $Z = E_Z[Z]$. This implies $P(X \leq x|Y) = P(X \leq x)$, i.e., X is independent of Y. □

B. SEARCHING FOR THE OPTIMAL HISTOGRAM

We show our solution for when X is categorical. The solution for when X is numerical follows similarly.

Let g be a histogram of Y. We denote the number of bins of g as $|g|$. We write Y^g as Y discretized by g. Following [28], we restrict that $|g| < N^\epsilon$ where N is the number of tuples in the joint distribution of X and Y, and $\epsilon \in (0, 1)$. We formulate the following problem: *Find the histogram g of Y with $|g| < N^\epsilon$ that minimizes* $H(X|Y^g)$.

We prove that our optimization problem can be solved by dynamic programming. In particular, w.l.o.g., let $Y(1) \leq \ldots \leq Y(N)$ be realizations of Y. Further, let

$$Y(j, m) = \{Y(j), Y(j + 1), \ldots, Y(m)\}$$

where $j \leq m$. Slightly abusing notation, we write $Y(1, N)$ as Y. We use $H(X|\langle Y(j, m)\rangle)$ to denote $H(X)$ computed using the $(m - j + 1)$ tuples of the joint distribution corresponding to $Y(j)$ to $Y(m)$, projected onto X. For $1 \leq l \leq m \leq N$, we write

$$f(m, l) = \min_{g:|g|=l} H(X|Y^g(1, m))$$

where g is a histogram of $Y(1, m)$ with l bins, and $Y^g(1, m)$ is the discretized version of $Y(1, m)$ by g. For $1 < l \leq m \leq N$, we have

THEOREM 5. $f(m, l) = \min_{j \in [l-1, m)} A_j$ *where* $A_j = \frac{j}{m} f(j, l - 1) + \frac{m-j}{m} H(X|\langle Y(j + 1, m)\rangle)$.

PROOF. Let $g^* = \arg\min_{g:|g|=l} H(X|Y^g(1, m))$. We denote l bins that g^* generates on Y as $b_1, \ldots, b_l$. We write $|b_t|$ as the number of values of Y in b_t. Further, let $c_z = \sum_{i=1}^{z} |b_t|$. Note that each bin of Y is non-empty, i.e., $c_z \geq z$. We use $H(X|b_t)$ to denote $H(X)$ computed using the tuples of the joint distribution corresponding to the realizations of Y in b_t, projected onto X. We have: $f(m, l)$

$$\begin{aligned}
&= \sum_{t=1}^{l} \frac{|b_t|}{m} H(X|b_t) \\
&= \sum_{t=1}^{l-1} \frac{|b_t|}{m} H(X|b_t) + \frac{|b_l|}{m} H(X|b_l) \\
&= \frac{c_{l-1}}{m} \sum_{t=1}^{l-1} \frac{|b_t|}{c_{l-1}} H(X|b_t) + \frac{|b_l|}{m} H(X|b_l) \\
&= \frac{c_{l-1}}{m} f(c_{l-1}, l - 1) \\
&\quad + \frac{m - c_{l-1}}{m} H(X|\langle Y(c_{l-1} + 1, m)\rangle) \quad .
\end{aligned}$$

In the last line, $\sum_{t=1}^{l-1} \frac{|b_t|}{c_{l-1}} H(X|b_t)$ is equal to $f(c_{l-1}, l - 1)$ because otherwise, we could decrease $f(m, l)$ by choosing a different histogram of $Y(1, c_{l-1})$ into $l - 1$ bins. This in turn contradicts our definition of $f(m, l)$. Since $c_{l-1} \in [l - 1, m)$ and $f(m, l)$ is minimal over all $j \in [l - 1, m)$, we arrive at the final result. □

Theorem 5 shows that the optimal histogram of $Y(1, m)$ can be derived from that of $Y(1, j)$ with $j < m$. This allows us to design a dynamic programming algorithm to find the optimal histogram g of Y with $|g| < N^\epsilon$. To further boost efficiency, following [28], we limit the number of cut points of Y to $c \times N^\epsilon$ with $c > 1$. We do this using equal-frequency binning on Y with the number of bins equal to $(c \times N^\epsilon + 1)$. More elaborate pre-processing can be considered, yet is beyond the scope of this work. Regarding ϵ and c, the larger they are, the more candidate histograms we consider, and hence, the better the result. However, setting them too high causes computational issues. Our preliminary empirical analysis shows that $\epsilon = 0.333$ and $c = 2$ offer a good balance between quality and efficiency. Hence, we will use these values in our experiments. The total time complexity of our histogram search is $O(N^{3\epsilon}) = O(N)$.