Reducing Metadata Complexity for Faster Table Summarization [*]

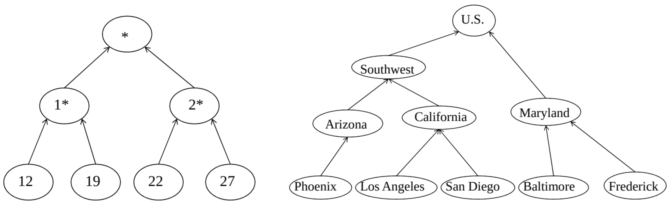
K. Selçuk Candan
Arizona State University
Tempe, AZ 85283, USA
candan@asu.edu

Mario Cataldi
Università di Torino
Torino, Italy
cataldi@di.unito.it

Maria Luisa Sapino
Università di Torino
Torino, Italy
mlsapino@di.unito.it

ABSTRACT

Since the visualization real estate puts stringent constraints on how much data can be presented to the users at once, table summarization is an essential tool in helping users quickly explore large data sets. An effective summary needs to minimize the information loss due to the reduction in details. Summarization algorithms leverage the redundancy in the data to identify value and tuple clustering strategies that represent the (almost) same amount of information with a smaller number of data representatives. It has been shown that, when available, metadata, such as value hierarchies associated to the attributes of the tables, can help greatly reduce the resulting information loss. However, table summarization, whether carried out through data analysis performed on the table from scratch or supported through already available metadata, is an expensive operation. We note that the table summarization process can be significantly sped up when the metadata used for supporting the summarization itself is pre-processed to reduce the unnecessary details. The pre-processing of the metadata, however, needs to be performed carefully to ensure that it does not add significant amounts of additional loss to the table summarization process. In this paper, we propose a *tRedux* algorithm for value hierarchy pre-processing and reduction. Experimental evaluations show that, depending on the table and taxonomy complexity, metadata summarization can provide gains in table summarization time that can range (in absolute values) from seconds to 10s-of-1000s of seconds. Consequently, while resulting in only an extra $\sim$ 20% reduction in table quality, *tRedux* can provide $\sim 2\times$ speedups in table summarization time. Experiments also show that *tRedux* has a better performance than alternative metadata reduction strategies in supporting table summarization; and, as the taxonomy complexity increases, the absolute gains of *tRedux* also increase.



(a) Hierarchy for *Age* (b) Hierarchy for *Location*

Figure 1: Value hierarchy for attribute *Age* (a) and *Location* (b); directed edges denote the clustering/summarization direction (taken from [1])

Categories and Subject Descriptors

H.2.4 [**Information Systems**]: Database Management—*systems, database applications*; H.3.3 [**Information Systems**]: Information Storage and Retrieval—*information search and retrieval*

General Terms

Algorithms, Experimentation

Keywords

Table Summarization, Metadata Complexity, Taxonomy reduction

1. INTRODUCTION

Table summarization is an important tool in helping users quickly explore large data sets. An effective summary needs to minimize the information loss due to the reduction in details; in particular each tuple in the original table needs to be represented in the summary with a sufficiently similar tuple. Moreover, each tuple in the summary must be sufficiently different from other summary tuples to ensure that the summary real-estate is not wasted. Summarization algorithms leverage the underlying redundancy (such as approximate functional dependencies and other patterns) in the data to identify value and tuple clustering strategies that represent the (almost) same information with a smaller number of data representatives.

When available, metadata, such as value hierarchies (like the ones shown in Figure 1) can help greatly reduce the resulting information loss. Value hierarchies have been commonly used for user-driven data analysis (e.g. OLAP [2]) and exploration *within* large data sets. In [1], we have shown that value hierarchies associated to the attributes of the tables can also be used to support table summarization. Con-

[*] Partially supported by NSF Grant "Archaeological Data Integration for the Study of Long-Term Human and Social Dynamics" (0624341)

EDBT 2010, March 22–26, 2010, Lausanne, Switzerland.

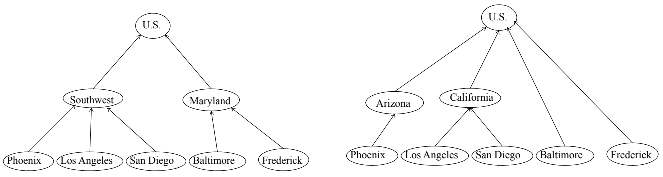
Name	Age	Location
John	12	Phoenix
Sharon	19	Los Angeles
Mary	19	San Diego
Peter	22	Baltimore
James	22	Frederick
Alice	27	Baltimore

(a) Data table

Name	**Age**	**Location**
-	1*	Southwest
-	2*	Maryland

(b) Summarized table

Table 1: (a) A database and (b) a summary on the $\langle Age, Location\rangle$ pair using hierarchies in Figure 1 (also taken from [1])



(a) Alternative #1 (b) Alternative #2

Figure 2: Two possible reductions of the location hierarchy in Figure 1

sider, for example, Table 1 (a) which shows a data table consisting of 6 rows. If the user is interested in summarizing this table based on the attribute pair, $\langle Age, Location\rangle$ in such a way that the summarized table can be visualized in a space that can hold at most 2 tuples, the hierarchies in Figure 1 can be used to obtain the summary in Table 1(b).

In this paper, we note that table summarization, whether carried out through data analysis performed on the table from scratch or supported through already available metadata, is an expensive operation. For example, the computational cost of the metadata supported table summarization process is exponential in the depth of the hierarchy (i.e., the number of alternative value clustering strategies) [1, 3].

The key observation driving the work in this paper is that the speed of the table summarization process can be significantly improved when the metadata used for supporting summarization itself is pre-processed to reduce its unproductive details. Metadata are often provided by domain experts and their primary usage is in data organization and interpretation. As such, they can be overly detailed and all of these details may not be relevant in the summary of the data. The pre-processing of the metadata to eliminate details not relevant for obtaining a table summary, however, needs to be performed carefully to ensure that it does not add significant amounts of additional loss to the table summarization process. For example, while the reduced hierarchy in Figure 2(a) would still give the table summary in in Table 1(b), the alternative in Figure 2(b) would cause further loss in the table summary. Thus, intuitively, in this context a "good" reduction of the given metadata is the one that leads to high-quality table summaries.

Based on these observations, we propose a novel hierarchy/taxonomy reduction method, *tRedux*, which reduces the complexities (defined in terms of the hierarchy height and density) of the given value hierarchies while preserving their effectiveness in table summarization. In the next section, we provide an overview of the relevant literature. Section 3 formalizes the table summarization problem and introduces the *quality* measures for the table summaries. In Sections 4 and 5, we first introduce our *tRedux* algorithm for taxonomy reduction and, then, describe the use of *tRedux* within the table summarization process.

Evaluations presented in Section 6 show that, depending on the table and taxonomy complexity, metadata summarization can provide gains in table summarization time that can range (in absolute values) from seconds to 10s-of-1000s of seconds. Consequently, while resulting in only an extra $\sim 20\%$ reduction in table quality, *tRedux* can provide $\sim 2\times$ speedups in table summarization time. Experiments also show that *tRedux* has a better performance than alternative metadata reduction strategies in supporting table summarization; and, as the taxonomy complexity increases, the absolute gains of *tRedux* also increase.

2. RELATED WORK

2.1 Table Summarization

[4, 5] present a table summarization system, *SaintEtiQ*, which computes and incrementally maintains a hierarchically arranged set of summaries of the input table. *SaintEtiQ* uses background knowledge (i.e., metadata) to support these summaries. [6] also performs data summarization, but it relies on frequent patterns in the relational dataset. TabSum [7] creates and maintains table summaries through row and column reductions. To reduce the number of rows, the algorithm first partitions the original table into groups based on one or more attribute values of the table, and then collapses each group of rows into a single row relying on the available metadata, such as the concept hierarchy. For column reduction, it simplifies the value representation and/or merges multiple columns.

Data compression techniques, like Huffman or Lempel-Ziv, can also be used to reduce the size of the table. For example, [8] presents a database compression technique based on vector quantization. Buchsbaum *et al.* [9] develop algorithms to compress massive tables through a partition-training paradigm, but the compressed tables are not human readable. Histograms can also be exploited to summarize information into a compressed structure. Following this idea, Buccafurri *et al.* [10] introduce a quad-tree based partition schema for summarizing two-dimensional data. Leveraging the quad-tree structure, [11] proposes approaches to processing OLAP operations over the summary. In fact, any multidimensional clustering algorithm can be used to summarize a table. Such methods, however, do not take into account specific domain knowledge (e.g. "what are acceptable summarizations, how do they rank?") that hierarchies would provide.

The concept of imprecision in OLAP dimensions is discussed in [12]. In that framework, a fact (e.g., a tuple in the table) with imprecise data is associated with dimension values of coarser granularities, resulting in the dimensional imprecision. In [13], we supported OLAP operations over imperfectly integrated taxonomies. We proposed a reclassification strategy which eliminates conflicts by introducing minimal imprecision. This approach is complementary to the work presented here: the obtained navigable taxonomy can be taken as the input for summarization.

The table summarization task is also related to k-anonymization problem, introduced as a technique against linkage attacks on private data [3]. The k-anonymization approach eliminates the possibility of such attacks by ensuring that, in the disseminated table, each value combination of attributes is matched to k others. To achieve this, k-anonymization techniques rely on a-priori knowledge about acceptable value generalizations. Cell generalization schemes [14] treat each cell in the data table independently. Thus, different cells for the same attribute may be generalized in a different way. This provides significant flexibility in anonymization, while the problem remains extremely hard (NP-hard [15]) and only approximation algorithms are applicable under realistic usage scenarios [14]. Attribute generalization schemes [3, 16] treat all values for a given attribute collectively; i.e., all values are generalized using the same unique domain generalization strategy. While the problem remains NP-hard (in the number of attributes), this approach saves significant amount of time in processing and may eliminate the need for using approximation solutions, since it does not need to consider the individual values. Most of these schemes, such as Samarati's original algorithm [3], rely on the fact that, for a given attribute, applicable generalizations are in total order and that each generalization step in this total order has the same cost. [3] leverages this to develop a binary search scheme to achieve savings in time. [16] relies on the same observation to develop an algorithm which achieves attribute-based k-anonymization one attribute at a time, while pruning unproductive generalization strategies.

2.2 Metadata Reduction

Ontology summarization has been used to support various reasoning tasks. Fokouel *et al.* [17] focus on the problem of summarizing OWL ontologies to reduce the cost of reasoning with ontologies. Often, metadata (such as RDF collections) can be seen as graphs. [18] analyzes the metadata graph to measure centrality of the nodes (e.g. concepts) in the graph. In this approach, the centrality of a given node reflects its degrees of salience; thus the most salient nodes are maintained in the summary. In contrast, clustering based approaches discover groups of nodes such that the intra-group similarity is maximized and the inter-cluster similarity is minimized [19, 20]. [21] presented a novel hierarchical clustering algorithm, merging two clusters only if the inter-connectivity and closeness between two clusters are high relative to the internal inter-connectivity of the clusters and closeness of items within the clusters. [22] uses an agglomerative clustering algorithm that merges the two clusters with the greatest number of common neighbors. More recently, [23] and [24] use random walks based cluster analysis techniques.

Since many metadata types, such as taxonomies are hierarchical, researchers also experimented with tree summarization algorithms [25, 26]. Davood *et al.* [26] observed that summaries of XML trees did a much better job in document clustering tasks than by using edit distance values, e.g. [27], on the original trees. DataGuides [25] was one of the first approaches which attempted to construct structural summaries of hierarchical structures to support efficient query processing. Though this and similar methods work fine for tree based structures, the constructed summaries are not trees, but graphs. Various other summarization algorithms, such as [26, 28], focus on creating summaries suitable for efficient similarity-search in tree-structured data. Since, once again, the goal of these algorithms is not to obtain a smaller tree representing the larger one provided as input, but to find a representation that will speed up query processing, the resulting summaries are in the form of strings, hash sequences, and concept/label vectors. Our goal, in this paper, however is to reduce the size of the input taxonomy tree to support table summarization process, therefore these and similar algorithms are not applicable.

3. TABLE SUMMARIZATION

Summarization of large data tables is required in many scenarios where it is hard to display complete data sets. The summarization process takes as input a database table and returns a reduced version of it. The result provides tuples with less precision (knowledge relaxation) than the original, but still informative of the content of the database. This reduced form can then be presented to the user for exploration or be used as input for advanced data mining processes.

3.1 Value Clustering Hierarchies

A value clustering hierarchy[1] is a tree $H(V,E)$ where V encodes values and clustering-identifiers (e.g., high-level concepts or cluster labels, such as "1*" in Figure 1(a), and E contains acceptable value clustering relationships.

Definition 3.1 (Value Hierarchy). *A value clustering hierarchy, H, is a tree $H(V,E)$:*

- *$v = (id\!:\!value) \in V$ where $v.id$ is the node id in the tree and $v.value$ is either a value in the database or a value clustering encoded by the node.*
- *$e = v_i \rightarrow v_j \in E$ is a directed edge denoting that the value encoded by the node v_j can be clustered under the value encoded by the node v_i.*

Those nodes in V which correspond to the attribute values that are originally in the database do not have any outgoing edges in E; i.e., they form the leaves of the hierarchy. Given an attribute value in the data table T and a value hierarchy corresponding to that attribute, we can define alternative clusterings as paths on the corresponding hierarchy.

Definition 3.2 (Value Clustering). *Given a value clustering hierarchy H, a tree node v_i is a clustering of a tree node v_j, denoted by $v_j \preceq v_i$, if $\exists path\ p = v_i \rightsquigarrow v_j$ in H. We also say that v_i covers v_j.*

3.2 Tuple-Clustering and Table Summary

Let us consider a data table, T, and a set, SA, of summarization attributes. Roughly speaking, our purpose is to find another relation T' which clusters the values in T such that T' summarizes T with respect to the summarization-attributes. Based on the above, in the following, we formalize the concept of tuple summarization.

Definition 3.3 (Tuple-Clustering). *Let t be a tuple on attributes $SA = \{Q_1, \cdots, Q_q\}$. t' is said to be a clustering of the tuple, t, (on attributes SA) iff $\forall i \in [1,q]$*

- *$t'[Q_i] = t[Q_i]$, or*
- *$\exists path\ p_i = t'[Q_i] \rightsquigarrow t[Q_i]$ in the corresponding value hierarchy H_i.*

[1] We use the terms "value hierarchy" and "value clustering hierarchy" interchangeably

In this paper, we use $t \preceq t'$ as shorthand.

Given this definition of tuple-clustering, we can define the summary of a table as a one-to-one and onto mapping which clusters the tuples of the original table.

Definition 3.4 (Table Summary). *Given two data tables T and T' (with the same schema), and the summarization-attribute set SA, T' is said to be a summary of T on attributes in SA ($T[SA] \preceq T'[SA]$ for short) iff there is a one-to-one and onto mapping, μ, from the tuples in $T[SA]$ to $T'[SA]$, such that*

- $\forall t \in T[SA],\ t \preceq \mu(t)$

Here, $T[SA]$ and $T'[SA]$ are projections of the data tables T and T' on summarization-attributes.

3.3 Table Summarization Process

As described in Section 2.2, there are various metadata supported table summarization algorithms [1, 3, 16]. Without loss of generality, in this work, we use Samarati's k-anonymization algorithm [3] as the back-end table summarizer. In [3], each unique tuple gets clustered with at least $k-1$ other similar tuples to ensure that no single tuple can be uniquely identified. The algorithm uses attribute value hierarchies to ensure that the amount of loss (i.e., value generalizations using the value hierarchies) is minimized. For each attribute, the algorithm takes a *value clustering hierarchy* (a taxonomy) which describes the generalization/specialization relationship between the possible values. For example, consider a table with a "*Location*" attribute. The hierarchy represents all the relevant values in the corresponding domain as the leaves of a tree (Figure 1(b)). The internal nodes in the value hierarchy will correspond to appropriate (geographic or political) clusterings of countries. Thus, in a summary, an internal node of the hierarchy can be used to cluster all the leaves below it using a more general label. If in the summary a leaf value is used, this gives zero generalization ($g = 0$); if, on the other hand, a leaf at depth d is replaced with an internal node at depth d', this causes $g = d - d'$ steps of generalization; of course, by picking clusters closer to the root, the algorithm will be able to summarize more easily. On the other hand, more general cluster labels also cause higher degree of knowledge relaxation. In Section 3.4, we will refer to the knowledge relaxation due to the use of generalizing clusters as *dilution*.

Among all possible clusterings that put each tuple with $k-1$ other similar ones, [3] aims to find those that require *minimal generalizations*; i.e., the amount of distortion in the data needed to achieve the clustering is as small as possible. Intuitively, if there is a generalization at depth d that puts all tuples into clusters of size k, then there will be generalizations of level $d' \leq d$ that also cluster all tuples into clusters of size at least k, but will have more loss; conversely, if one can establish that there is no generalization at level d that is a k-clustering, then it follows that there are no other clustering of level $d' > d$ that can cluster all tuples into clusters of size at least k. Relying on the fact that for a given attribute, applicable domain generalizations are in total order and that each generalization step in this total order has the same cost, [3] develops a binary search scheme to achieve savings in time[2]. It starts evaluating generalization levels from the middle-level to see if there is a corresponding k-clustering solution:

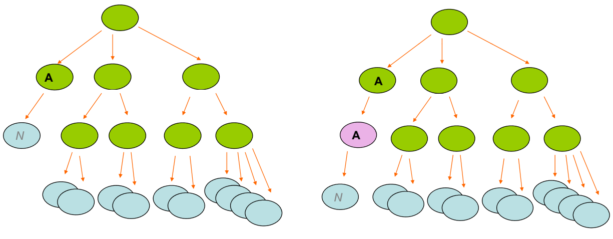


Figure 3: *Ghost* nodes in unbalanced hierarchy

- if there is, then the algorithm tries to find another solution with less generalization by jumping to the central point of the half path with lower generalization;

- if there is none, on the other hand, the algorithm tries to find a solution by jumping to the central point point of the half path with higher generalization.

The process continues in this binary search until a generalization level such that no solution with a lower generalization exists is found. Unfortunately, this and other attribute-generalization based algorithms, including [16] and [29], are all exponential in the number of attributes that need summarization[3]. When there is a single attribute to summarize, for a hierarchy, H, of depth, $depth(H)$, the algorithm considers $log(depth(H))$ alternative clustering strategies. When there are m attributes to consider, however, the maximum degree of generalization is $\sum_{1 \leq i \leq m} depth(H_i)$, where all attributes are generalized to the max, causing the greatest amount of loss. In this case, the algorithm considers $log(\sum_{1 \leq i \leq m} depth(H_i))$ alternative generalization levels on the average; moreover, for each generalization level, g, the algorithm has to consider all combinations of attribute generalizations such that $(\sum_{1 \leq i \leq m} g_i) = g$. Since in general, there can be exponentially many such combinations, the worst case time cost of the algorithm is exponential in the number of attributes.

One problem with using Samarati's algorithm [3] as the back-end table summarizer is that it assumes balanced hierarchies as input. In order to perform table summarization using unbalanced input hierarchies, we first balance the input hierarchies by introducing *ghost* nodes that fill the empty spots in the hierarchy. As shown in Figure 3, these ghost nodes act as surrogates of the closest ancestors of the leaf nodes that are out of balance: in this example, in order to put N at the same level of the other leaves, we introduce a *ghost* node between N and its parent A (Figure 3(b)). The ghost node is also labeled A as the parent. Note that, whether N is generalized to its original parent or the new (ghost) node, it will be replaced in the summarized table with A, therefore, this transformation does not result in additional loss.

3.4 Quality of a Table Summary

Information-based measures of quality leverage statistical knowledge, for example the knowledge about data frequencies, to measure information loss. One advantage of the use of value hierarchies for table summarization is that

[2] [16] relies on the same observation to develop an algorithm which achieves attribute-based k-anonymization one attribute at a time, while pruning unproductive attribute generalization strategies. [29] further assumes an attribute order and attribute-value order to develop a top-down framework with significant pruning opportunities.

[3] In fact the problem is NP-hard [1]

the degree of loss resulting from the summarization process, can be quantified and explicitly minimized using the available value hierarchies. Unlike purely numeric information loss measures, such as mean squared error, and statistical measures, such as entropy, classification, discernibility, and certainty [29, 30, 31, 32], knowledge about value hierarchies provides a mechanism to judge the significance of the distortion within the given application domain [3, 33, 34]. For example, a commonly used technique for measuring the amount of loss during the summarization process is to count the number of generalization steps required to obtain the summary [3, 29, 16]: given a generalization hierarchy, each step followed to achieve the value clustering is considered one unit of loss.

Definition 3.5 (Penalty of a Tuple Clustering). *Let t and t' be two tuples on attributes $SA = \{Q_1, \cdots, Q_q\}$, such that $t \preceq t'$. Then the cost of the corresponding clustering strategy is defined through a monotonic combination function, $\sum$, of the penalty of the clustering along each individual summarization-attribute:*

$$\Delta(t \preceq t') = \sum_{1 \leq i \leq q} \Delta_i,$$

where

- $\Delta_i = 0$, *if* $t'[Q_i] = t[Q_i]$
- $\Delta_i = \Delta(t[Q_i], t'[Q_i])$ *(i.e., the minimal number of edges that separates* $t[Q_i]$ *to* $t'[Q_i]$ *in the corresponding original hierarchy) otherwise.*

Let us consider a data table T, and a set SA of summarization attributes. Let T' be a summary of T on attributes in SA (i.e., $T[SA] \preceq T'[SA]$). In this paper, we use the following quality measures to evaluate table summaries:

- *dilution (dl)*:

  $$dl(T, T', SA) = \frac{1}{|T|} \sum_{t \in T} \Delta(t[SA], \mu(t[SA])).$$

  The smaller the degree of dilution, the smaller is the amount of loss and the higher is the quality of the summary.

- *diversity (div)*:

  $$div(T', SA) = \frac{2}{|T'|(|T'|-1)} \sum_{t_1, t_2 \in T' (t_1 \neq t_2)} \Delta(t_1[SA], t_2[SA]).$$

  The greater the diversity, the higher is the quality of the summary.

Therefore, given a table T and the set of summarization attributes, SA, the goal of the summarization algorithm is to find a summary such that the degree of dilution is minimized, yet the diversity is maximized.

4. TAXONOMY REDUCTION

As we have seen in Section 3.3, the table summarization process can be prohibitively costly, especially when the number of relevant attributes is large. Our key observation in this paper is that *it might be possible to reduce the cost of the table summarization algorithm significantly by reducing the sizes of the input hierarchies.* The hierarchy reduction, however, needs to be performed in such a way that it does not add significant amounts of additional loss to the table summarization process. In other words, the hierarchy reduction process should eliminate the details in the metadata that are not likely to be used in the table summary anyhow (remember the example in Figure 2).

In this section, we propose a *tRedux* algorithm for value hierarchy reduction. Our proposed approach to taxonomy preprocessing and reduction consists of three steps:

I: create a graph representing the structural distances between the nodes in the taxonomy as well as the distribution of nodes labels in the database;

II: partition the resulting graph into disjoint sub-graphs based on connectivity analysis; and

III: finally, select a representative label for each partition and reconstruct a taxonomy tree.

In Section 6, we show that taxonomies reduced using this algorithm help preserve qualities of table summaries, while providing significant gains in table summarization time. In the rest of this section, we describe each of the above steps.

4.1 Step I: Constructing the Node Structural-Similarity/Occurrence Graph

Naturally, the most effective way to ensure the quality of the table summarization process is to cluster those taxonomy nodes whose labels would be judged to be similar by human users of the system. Simultaneously, the cluster should also represent the joint-distribution of the node labels in the database. Therefore, the first step of the process is to create a graph that represent both the structural similarities of the nodes and label co-occurrence in the database.

More formally, let us consider a data table T and a set SA of summarization attributes. Let $H_i(V_i, E_i)$ be a value hierarchy corresponding to the attribute $Q_i \in SA$. In the next step, the algorithm constructs a weighted directed graph, $G_i(V_i, E'_i, w)$, where the set of vertices, $V_i = \{v_1, \ldots, v_n\}$, corresponds to the concepts in the input taxonomy. The weights ($w : E \rightarrow R^+$) associated to the edges in E represent both

- similarities between pairs of concepts in the taxonomy, and
- occurrences of data values in the database.

In structure-based methods, the similarity between two nodes in a taxonomy is measured by the distance between them [35] or the sum of the edge weights along the shortest path connecting the words [36]. CP/CV, presented in [37] first associates a node vector to each node in the taxonomy (the vector represents the relationship of this with the rest of the nodes in the taxonomy) and, then, compares the vectors to quantify the how structurally similar the two nodes are. Comparisons against other approaches on available human-generated benchmark data [38, 39] showed that CP/CV improves the correlation of the resulting similarity judgments to human common sense. Thus, without loss of generality, we use the node similarity measure presented in [37] to quantify the structural similarities among the taxonomy nodes. More specifically, given the data table T, for each value hierarchy, $H_i(V_i, E_i)$, we construct a complete directed graph,

$G_i(V_i, E'_i, w)$: for each pair v_a to v_b of nodes in the taxonomy H_i, the edge between the corresponding nodes in G_i has the following weight:

$$w(\langle v_a, v_b \rangle) = \sum_{t \in T} cv(v_a)[t.Q_i] \times cv(v_b)[t.Q_i],$$

where $t.Q_i$ is the value of tuple t for attribute Q_i and $cv(x)[y]$ gives the CP/CV value for node x along the vector dimension corresponding to the node y. Intuitively, the weight $w(\langle v_a, v_b \rangle)$ measures the aggregate similarity between the taxonomy nodes v_a and v_b in the value hierarchy for all the values in the corresponding attribute in the database. Thus, the resulting graph $G_i(V_i, E'_i, w)$ represents the structural relationships in $H_i(V_i, E_i)$ as well as the distribution of the data in the corresponding summarization attribute, Q_i, in the database: the weight of an edge is high if the concepts are structurally related in the value hierarchy, H_i, and there is plenty of tuples in the corresponding attribute that are highly related to these concepts.

Lastly, this graph $G_i(V_i, E'_i, w)$ is thinned by applying a locally adaptive edge thinning algorithm. For each v_a in V, we consider the set, $out(v_a)$, of all outgoing edges:

1. we first sort the edges in $out(v_a)$ in decreasing order of weights;
2. next, we compute the *maximum drop* in consecutive weights; and identify the corresponding *max-drop* point in the sorted list of edges;
3. we, then, compute the *average drop* (between consecutive entities) for all those edges that are ranked before the identified *max-drop* drop point.
4. the first weight drop which is higher than the computed average drop is referred to as the *critical-drop*. All the edges in $out(v_a)$ beyond this *critical-drop* point are eliminated from E'_i.

This final thinning process ensures that only those edges that represent strongest relationships are maintained (note that, since the graph is directed and the thinning process is asymmetrical, it is possible that the E'_i will contain a link from v_a to v_b, but not vice versa).

4.2 Step II: Balanced Taxonomy Partitioning

In the next step, the resulting weighted graph $G_i(V_i, E'_i, w)$ is partitioned based on its connectivity and the weights. In theory, any existing graph partitioning algorithm (e.g. [19, 20, 40, 24, 41]) can be used in this stage. Many of these (including METIS [41], which we evaluate in the experiments section), however, require advance knowledge about the number of clusters. Thus, in practice, since the user will not be likely to have a target taxonomy size for table summarization, a summarization algorithm which can partition the input graph based on its inherent structure, without requiring an input number of clusters, may be more suitable.

Consequently, without loss of generality, we rely on a random walk-based graph partitioning algorithm that does not require an advance knowledge of the number of resulting clusters. Intuitively, two vertices in the same cluster should be more connected to each other than two vertices in other clusters (and that two vertices in the same cluster should be quickly reachable from each other through a random walk)[42]. In particular, given the graph $G_i(V_i, E'_i, w)$ constructed in the previous step we construct a random walk graph by associating the following transition probability to edges: let e be an edge from vertex v_a to vertex v_b; then, the corresponding probability of transition is

$$p_{a,b} = \frac{w(\langle v_a, v_b \rangle)}{\sum_{e_k \in out(v_a)} w(e_k)}.$$

The resulting node-to-node transition probability matrix is then used, as in [42], to partition the nodes based on their proximities.

Existing random-walk based clustering algorithms, such as [42], consider only the connectivity and the weights and does not seek to return partitions balanced in terms of the number of vertices. In other words, the number and the sizes of the clusters is strictly dependent on the weights obtained in Step II. For data summarization, however, we may not want a summarized taxonomy, where some summary nodes are precise (and represent only a few nodes in the original value hierarchy), whereas others are vague (and represent large numbers of nodes). An equally distributed set of clusters, on the other hand, would permit to generate a more informative and representative summary of the initial taxonomy, as each new entry in the reduced taxonomy represents (approximately) the same number of original nodes.

Therefore, we follow the initial partitioning step with a re-balancing step. Let $H_i(V_i, E_i)$ be a value hierarchy and $\mathcal{P}_i = \{P_{i,1}, \ldots, P_{i,m}\}$ be the set of partitions obtained through the random walk process. In order to promote balance in partitions, we introduce a *tolerance value*, $\tau = \theta\frac{|V_i|}{m}$, that sets the maximum number of concepts that could be represented by any partition. If a cluster, $P_{i,j}$, contains too large a number of concepts, then a set, $\mathcal{X}_i$, of extra vertices are picked and moved to other partitions. This set of vertices are selected in such a way that the cost, $cost(\mathcal{X}_i)$, of displacement of the set of extra vertices among partitions. The term $cost(\mathcal{X}_i)$ is

$$\sum_{v_a \in \mathcal{X}_i} \left(\sum_{e_j \in (edges(v_a) \cap \mathcal{P}_i)} w(e_j) - \sum_{e_j \in (edges(v_a) \cap dest(v_a))} w(e_j) \right),$$

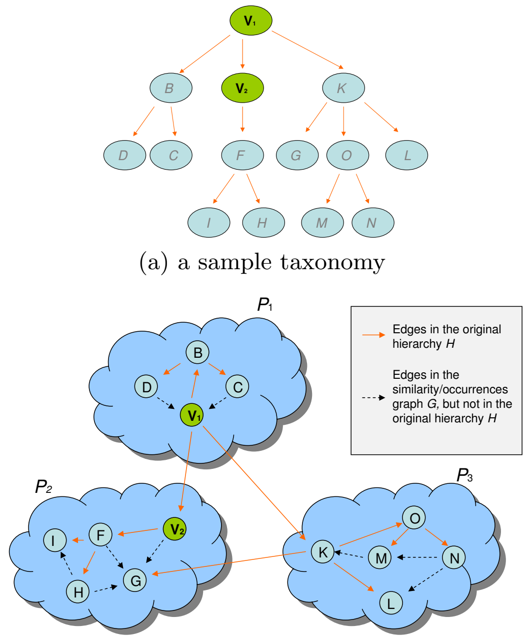
where $edges(v_a)$ is the set of all incoming and outgoing edges to v_a and $dest(v_a)$ is the partition, other than P_i, with the highest weighted connectivity to v_a. The vertices in $\mathcal{X}_i$ and their destinations are selected through a K-means like iterative improvement process. In Section 6.2 we will deeply study the use of θ.

4.3 Step III: Taxonomy Re-construction

In order to construct the reduced taxonomy, we need to reattach the partitions, obtained in the previous step, in the form of a tree structure. Furthermore, for each partition, we need to pick a *label* describing the concepts in the partition.

4.3.1 Partition Labeling

Considering a hierarchy node, its label is important because it is what will be presented to the user in the resulting table summary. Thus, considering our partitions, we have to carefully select the appropriate labels in order to be sufficiently representative of the cluster. Let $P_{i,j}$ be a partition in $\mathcal{P}_i$. In order to pick a label for $P_{i,j}$, we consider the relationships of the vertices in $P_{i,j}$ in the original hierarchy H_i. If there is a vertex, $v_a \in P_{i,j}$ that dominates all the other



V₁
B
V₂
K
D
C
F
G
O
L
I
H
M
N
P₁
P₂
P₃
V₁
V₂
Edges in the original hierarchy H
Edges in the similarity/occurrences graph G, but not in the original hierarchy H

(a) a sample taxonomy

(b) partitioning example

Figure 4: A sample taxonomy and its partitioning

vertices in the partition (i.e., $\forall v_b \in P_{i,j} \;\; v_b \preceq v_a$), then v_a is selected as the label. If there is no such single vertex, then the minimal set, D_j, of vertices covering the partition $P_{i,j}$ (based on H_i) is found and the set, D_j, is used as the partition label.

4.3.2 Partition Linking

The reduced taxonomy, H_i' should preserve the original structure of H_i as much as possible:

- The root of H_i' is the partition $P_{i,j}$ which contains the root vertex of H_i.

- Let us consider a pair, $P_{i,j}$ and $P_{i,k}$, of partitions in $\mathcal{P}_i$. Let $E_{j,k}$ be the set of edges in H_i that go from the vertices in $P_{i,j}$ to vertices in $P_{i,k}$. Similarly, let $E_{k,j}$ be the set of edges in H_i that go from the vertices in $P_{i,k}$ to vertices in $P_{i,j}$.

  If in H_i', $P_{i,j}$ is an ancestor of $P_{i,k}$, then the broken set of edges in $E_{k,j}$ will result in structural constraints that are violated. If $P_{i,k}$ is an ancestor of $P_{i,j}$, then broken edges in $E_{j,k}$ will result in structural constraints that are violated. If neither is an ancestor of the other, on the other hand, the edges in $E_{k,j} \cup E_{k,j}$ will determine the constraints that are violated.

  Let $e = \langle v_a, v_b \rangle$ be an edge from partition $P_{i,j}$ to $P_{i,k}$. If e is broken, then its cost ($cost(e)$) is the number of descendants of v_b in the original hierarchy H also contained in $P_{i,k}$ plus one (for v_b). For example, in Figure 4, if the edge between V_1 and K is broken, then the cost of this edge is equal to $1 + |\{O, N, L, M\}| = 5$.

  Thus, the taxonomy H_i', minimizing the errors due to structural constraint violations can be constructed by

  1. creating a complete weighted directed graph, $G_P(V_P, E_P, w_P)$, of partitions, where
     – $V_P = \mathcal{P}_i$,
     – E_P is the set of edges between all pairs of partitions, and
     – $w_P(\langle P_{i,j}, P_{i,k} \rangle) = \sum_{e \in E_{k,j}} cost(e)$; and
  2. finding a *maximum spanning tree* of G_P rooted at the partition $P_{i,j}$ which contains the root of H_i.

At this point, the original taxonomy has been partitioned and a reduced taxonomy, which can be used in the table summarization process, has been reconstructed.

5. TABLE SUMMARIZATION USING REDUCED VALUE HIERARCHIES

Let us consider a data table T and a set $SA = \{Q_1, \cdots, Q_q\}$ of summarization attributes. Let $H_1, \ldots H_q$ be the corresponding value hierarchies. In Section 3.2, Definition 3.4 introduces the summary, T', of T based on the given value hierarchies. Let $H_1', \ldots H_q'$ be a set of reduced value hierarchies, corresponding to the summarization attributes.

Remember from Section 3.4 Definition 3.5 that if t and t' are two tuples in T on attributes $SA = \{Q_1, \cdots, Q_q\}$, such that $t \preceq t'$, then the penalty of clustering is defined as

$$\Delta(t \preceq t') = \sum_{1 \leq i \leq q} \Delta_i,$$

where

- $\Delta_i = 0$, if $t'[Q_i] = t[Q_i]$

- $\Delta_i = \Delta(t[Q_i], t'[Q_i])$ (i.e., the penalty of replacing $t[Q_i]$ with $t'[Q_i]$) otherwise.

In the original definition, both $t[Q_i]$ and $t'[Q_i]$ are from the same hierarchy, H_i; when using reduced hierarchies for table summarization, however, while $t[Q_i]$ comes from H_i, $t'[Q_i]$ comes from the reduced hierarchy H_i'. Therefore, the definition of the $\Delta(t[Q_i], t'[Q_i])$ needs to be extended. Based on the partition labeling strategy for reduced taxonomies, discussed in Section 4.3.1, there are two cases to consider:

- *Case I: $t'[Q_i] \in H_i'$ is also in H_i.* If this is the case, the task is simpler and $\Delta(t[Q_i], t'[Q_i])$ can simply be computed on the original hierarchy.

- *Case II: $t'[Q_i] \in H_i'$ is not in H_i.* If this is the case, $t'[Q_i]$ must correspond to a set of values in H_i. Therefore, $\Delta(t[Q_i], t'[Q_i])$ needs to be computed using a monotonic combination function, $\sum$, of the replacement costs of the values in the original hierarchy:

  $$\Delta(t[Q_i], t'[Q_i]) = \sum_{v \in t'[Q_i]} \Delta(t[Q_i], v).$$

This monotonic combination function can be optimistic (*min*), pessimistic (*max* or *sum*), or agnostic (*avg*).

6. EVALUATION OF METADATA REDUCTION BASED TABLE SUMMARIZATION

Metadata supported table summarization needs three inputs: (a) a table T to summarize with q attributes; (b) a set of domain hierarchies D_i (for each attribute which we want to summarize), and (c) a parameter k that determines the minimum size of tuple clusters in the summary. We experimented with different data sets, taxonomies, and k values. We considered two different datasets:

- Real dataset: we used the *Census Income* dataset (also known as *Adult* dataset), extracted from the 1994 Census database [43]. This data set contains ∼30K tuples and includes 16 attributes.

- Synthetic dataset: we constructed sub-sets of tuples with different properties in order to evaluate *tRedux* under different conditions (see Section 6.2).

For both data sets, we varied the number of tuples in the data set. For all experiments described in Section 6.1, we considered 7 different subsets of tuples (from ∼ 100 to ∼ 800) for the Adult dataset and 6 different subsets (from ∼ 100 to ∼ 1000) for the synthetic dataset.

We also varied the *tuple count variance* (t-var), which is defined as the variance in the number of occurrences (in the input table) of the leaf values of the hierarchy; this value was varied between 0 (i.e., uniform distribution) and ∼ 11 for Adult dataset and between 0 and ∼ 15 for Synthetic dataset.

We also experimented with different numbers (1,2 and 3) of attributes in the summary. For each case, we considered different summarization requirements, varying k in the set $\{5, 10, 20, 30\}$. In addition to the real and synthetic data, we also experimented with real and synthetic domain hierarchies. The synthetic domain hierarchies we used for the experiments also varied in structure (size and height). We provide more details in Section 6.2.

Finally, we have also experimented with different partition balance tolerance values when creating the reduced taxonomies (see Section 4.3.2). We varied the tolerance value, θ in the set $\{1, 1.5, 2, 3, 4\}$ ($\theta < 1$ is not meaningful, $\theta = 1$ means balance, and $\theta > 1$ is increasingly lax in terms of balance requirement – as we will see in Section 6.2, diversity and dilution is more or less constant for $\theta \geq 2$, therefore this range is sufficient for observing the impact of θ). Unless explicitly stated, the default tolerance value, $\theta = 2$, is used. For all the experiments we used an Intel Core 2CPU @2,16GHz with 1GHz Ram.

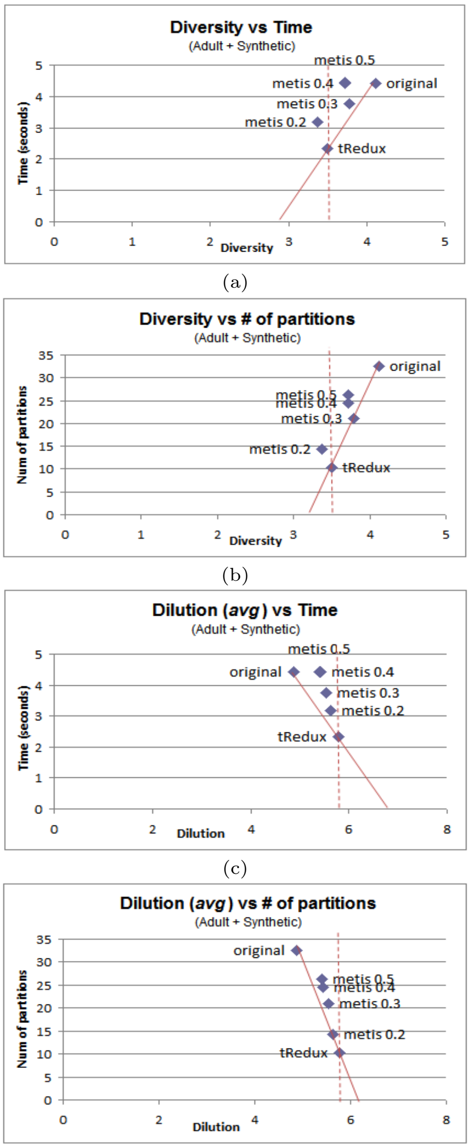
6.1 Loss in Diversity and Dilution due to Reduced Metadata

Before we analyze the behavior of the *tRedux*-based table summarization under different system parameters, we first compare dilution and diversity behaviors of various alternative metadata driven table summarization approaches. In particular, we compare the following alternative schemes:

- table summarization using the *original* hierarchies; in this scheme the input hierarchies are not reduced.

- table summarization using hierarchies reduced by applying *tRedux*.

- table summarization using hierarchies reduced by applying (instead of *tRedux*) k-METIS clustering [41] over the concept similarity graph described in Section 4.1: the k-METIS algorithm guarantees that all partitions will be approximately equally distributed; but requires the number of partitions as input. In these experiments, we vary the number of partitions as 20%, 30%, 40% and 50% of the number of nodes in the input hierarchy (METIS-0.2, -0.3, -0.4 and -0.5 in the charts).



(a)

(b)

(c)

(d)

Figure 5: Comparisons among *tRedux*, original, and k-METIS approaches: (a) diversity-vs-time, (b) diversity-vs.-number of partitions, dilution-vs-time, (c) dilution-vs-number of partitions

The experiments reported in this subsection are high-level averages of all experiments carried out with varying system parameters. As we mentioned above, we varied values of k, the number of tuples, the hierarchy size. Then, for each alternative algorithm, we computed average diversity, average dilution, and average execution time and plotted them against each other to observe the general, high-level trends without focusing on the impacts of the specific system parameters. In Figure 5, the first scheme, "*original*", does not use taxonomy reduction, while the other schemes, "*tRedux*" and "k-METIS", are both instances of taxonomy reduction based table summarization approach proposed in this paper.

Diversity vs. Time Figure 5(a) shows the amount of diversity maintained by alternative schemes against the amount of time required by the table summarization algorithm. As can be seen in this figure, table summarization using the original hierarchies provides the highest diversity; but also takes the greatest amount of time. METIS algorithms with 40% and 50% hierarchy nodes cause drops in the diversity, without any significant temporal gain. METIS with 20% and 30% nodes result in some gains in time; but the highest gain in time occurs when using *tRedux* for summaries. Most importantly though, the diversity-vs-time behavior (highlighted by the slopes of the line segments that connect the point corresponding to *original* summaries with the points corresponding to the algorithms), is the best for *tRedux*. Overall, *tRedux* provides a ∼ 50% gain in execution time, with only a ∼ 15% reduction in diversity.

Diversity vs. # of Nodes in the Reduced Hierarchy Figure 5(b) shows the diversity maintained by alternative schemes against the number of nodes (partitions) in the reduced hierarchy. As expected, there is a correlation with the number of nodes in the hierarchy with the overall diversity. However, as can be seen comparing METIS with 20% of nodes and *tRedux* results, *tRedux* is able to maintain a similar amount of diversity with smaller number of nodes in the hierarchy.

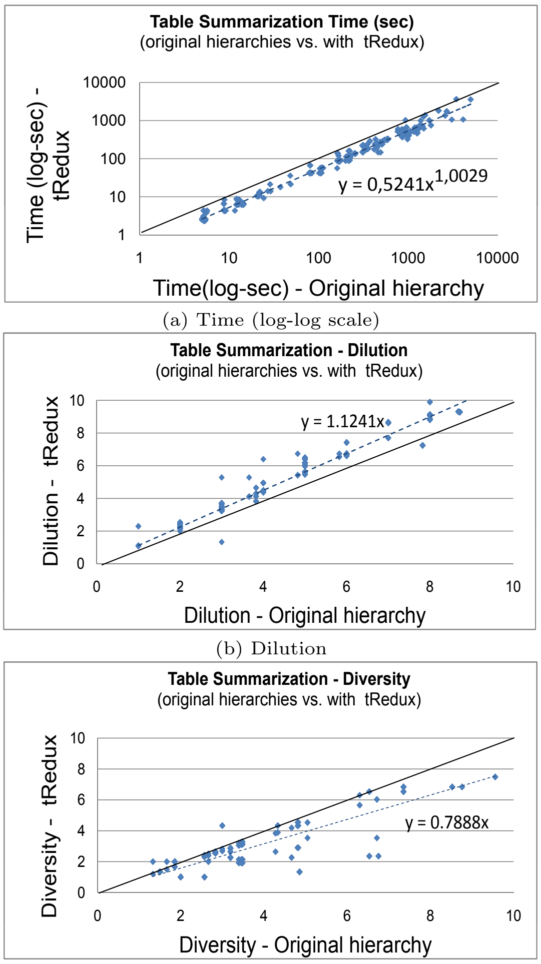
Dilution vs. Time Figure 5(c) shows dilution[4] against table summarization time: the highest absolute and relative (to dilution) time gains are achieved by the *tRedux*.

Dilution vs. # of Nodes in the Reduced Hierarchy Figure 5(d) shows the dilution[5] caused by alternative schemes against the number of nodes in the reduced hierarchy. As can be seen here, as expected, the smaller the number of nodes in the hierarchy, the higher the resulting dilution is. On the other hand, among the different metadata reduction schemes, *tRedux* has the best relative dilution behavior: a 66% drop in the number of nodes in the hierarchy results in only a less than 20% increase in dilution.

Summary The results in this section shows that *tRedux* is able to reduce the taxonomy (based on its inherent structure, without requiring the size of the output taxonomy as an input) in a way that provides the best diversity-time and dilution-time trade-off. Algorithms, like k-METIS can be used as the base graph partitioner if the user would like to reduce the sizes of the input taxonomies beyond what is structurally recommendable (albeit at the cost of further information loss).

6.2 Dissecting tRedux

In the previous subsection, we have looked at the *high-level* behavior of the various algorithms and seen that metadata reduction based table summarization can provide significant time gains, while resulting in relatively small increase in dilution and drop in diversity. We have also seen that among alternative ways to taxonomy reduction, *tRedux* has the best dilution-time and diversity-time behaviors. In this subsection, we look at the *tRedux* algorithm in greater detail and study how different problem parameters affect dilution, diversity and time behaviors of *tRedux*. In particular, we vary (a) the imbalance tolerance value, θ, (b) the number of tuples in the input table, (c) the value distributions in the data, (d) the sizes of the hierarchies, and (e) the heights of the hierarchies, and compare the *tRedux*-supported summaries with summaries using original hierarchies. We mostly experiment with synthetic data where we can freely change various parameters and observe the behaviour of tRedux, but we also include results with the Adults data set.



(a) Time (log-log scale)

(b) Dilution

(c) Diversity

Figure 6: Table summarization total results with and without *tRedux*

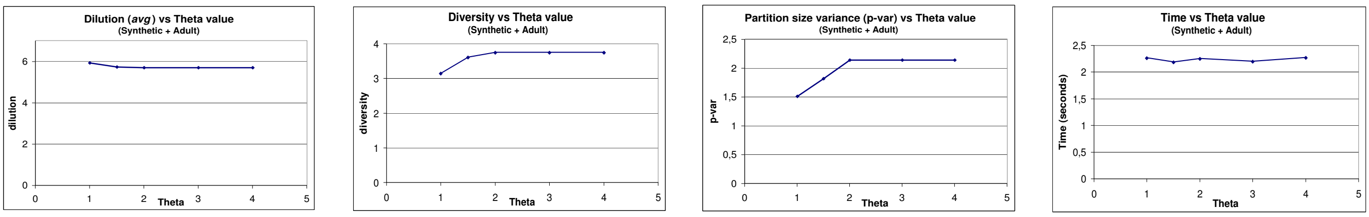
Overview First, Figures 6(a),(b) and (c) bring together all experiment instances (independent of their parameters) into three tables which plot performance measures (time, dilution and diversity) for table summarization with the original hierarchy against table summarization with *tRedux*. As the trend line in Figure 6(a) shows, on the average the summarization times with *tRedux* is just ∼ 50% of the summarization times needed with the original summary (i.e., summarization is 2× as fast when using *tRedux*) and this behavior is highly consistent. Moreover, the average loss in terms of dilution (Figure 6(b)) is only ∼ 12% higher when using a summarized taxonomy, while the average loss in terms of diversity is ∼ 21% (Figure 6(c)).

Next we consider the impact of the individual parameters on these three measures.

Impact of the Partition Imbalance (Θ Parameter) As introduced in Section 4.2, depending on the need, the user

[4] In this setting, we are using the agnostic *avg* combination function to compute dilution –see Section 5. In these experiments, the effect of the dilution definition on the result were extremely minute; thus charts considering other functions (*min*, *max*, *sum*) are omitted for the sake of space.

[5] Again, using agnostic *avg* combination function.



(a) (b) (c) (d)

Figure 7: Dilution, diversity, partition size variance, and time for different imbalance tolerance values.

can balance the resulting partitions by using the parameter θ. In fact, when reducing input hierarchies, creating partitions with widely varying sizes might be undesirable: some partitions in the reduced hierarchy will be more precise (corresponding to only a few entries in the original hierarchy), while some others will be very vague (corresponding to a large number of values). On the other hand, requiring perfectly balanced partitions might also be counter-productive since this may result in nodes in the re-constructed hierarchy that are consisting of poorly related (non-homogeneous) concepts in the original hierarchy. Indeed, as shown in Figures 7(a) and (b), requiring strictly balanced partitions ($\theta = 1$) results in a slightly higher dilution and lower diversity. Any $\theta \geq 2$, however, provides the same performance; this is because for such large θ values there is no need to re-balance the partitions. Thus, while we foresee that in most cases *tRedux* will be used with $\theta = 2$, we also recognize that some applications may require balanced partitions and (as shown in Figure 7(c)), in these cases, θ can be used to control the balance of the partitions. Note that, since the number of partitions stays the same, θ does not affect the table summarization time (Figure 7(d)). Notice that, while diversity and dilution stay constant for $\theta >= 2$, there is no need to consider very high values of θ.

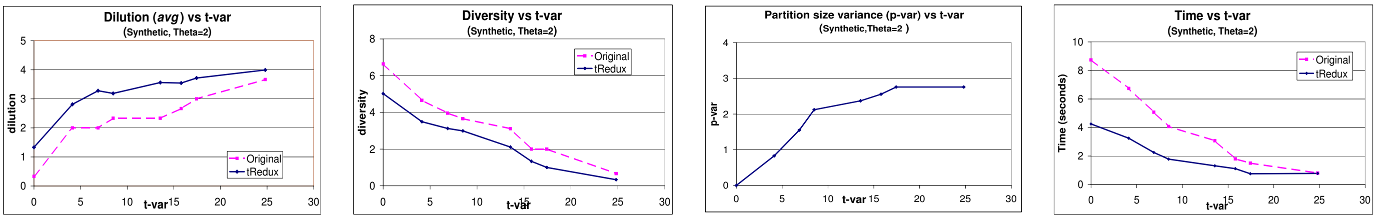
Impact of the Value Distribution in the Data Table We also experimented with different value distributions in the data table. Results for this setup are for a single attribute (with a balanced hierarchy with 127 nodes and 64 leaves) and 256 tuples in the table. In this experiment, we varied the *tuple count variance* (*t-var*) of the table between 0 and $\sim$ 25 (in the case with *t-var* = 24.80, we have almost all tuples distributed on only one leaf of the domain hierarchy and the other leaves are only represented by one tuple each). For each t-var, we analyzed 3 different randomly generated sets of tuples. Each presented result is the average of these three cases. As can be seen in Figures 8(a) and (b), large variances in the tuple distributions negatively impact the dilution and diversity for summarization. As expected, the original hierarchy provides better diversity and dilution than *t-Redux*, but is much slower Figure 8(d). As *t-var* increases, the dilution, diversity, and execution time behaviors of *t-Redux* and the original scheme approach each other. This is because an increase in the count variation also causes an increase in the partition size variation (Figure 8(c)) and when the partition variances are higher, Samarati's algorithm tends to pick nodes closer to the root instead of analyzing combinations of the internal nodes.

Impact of the Table Size To observe the effect of the table size, we considered a summarization scenario with a single attribute (having a hierarchy with 31 nodes and 16 leaves). We varied the database size from $\sim$ 50 to $\sim$ 5000000, with ×10 increments. For these experiments, we set *t-var* to 0. As these experiments show, as long as the tuples in the table are selected using the same value distribution, independent of the size of the table, the dilution and diversity stays the same. Figure 9(a),(b),(c) and (d) show the obtained results. Note that the time cost of the original scheme increases faster than the time cost of *tRedux* supported summarization as the table size increases.

Effects of the Number of Nodes in the Input Hierarchies In order to study the impact of the number of nodes in the input hierarchy, we selected 6 different hierarchies with different number of nodes (57, 115, 230, 460, 921 and 1843 nodes) but having the same height (13 levels). For each of the 6 considered cases, we analyzed 3 different random generated hierarchies and the presented results are the averages of these. For these experiments, we maintain *t-var* at 0. The results in Figure 10(a) and (b) show that the dilution and diversity behaviors of *tRedux* are not affected by the number of nodes. As shown in Figure 10(c), partition size variance *p-var* is also not affected by the changes in the number of nodes. As shown in Figure 10(d), the number of nodes affects the process in terms of execution time (because the algorithm needs to consider more nodes as candidates for the summary); the benefits of the *tRedux* scheme is more apparent for larger hierarchies.

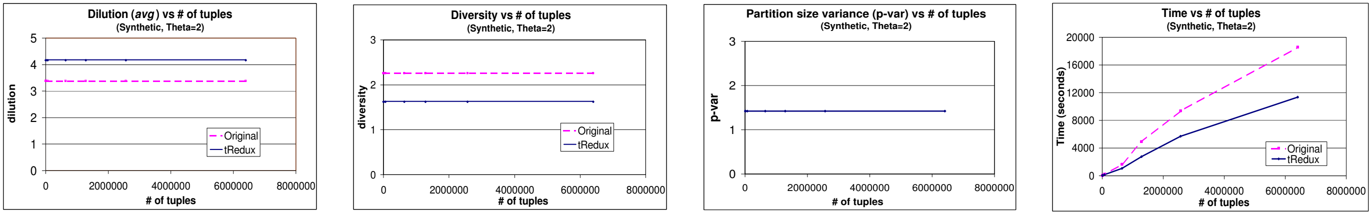
Effects of the Heights of the Input Hierarchies For these experiments, we used 8 different hierarchies with the same numbers of nodes (460 nodes of which 256 are leaves), but with different numbers of levels (8 through 17). For each of these cases, we experimented with 3 different random generated hierarchies; the presented results are averages. The table has 1024 tuples and has *t-var* of 0. As Figure 11(a) shows dilution of *t-Redux* is not affected by the height of the hierarchy. This is largely because of the fact that the partition size variance *p-var* is not affected by the changes in the hierarchy height (Figure 11(c)). The diversity of the summaries, however, increases with the height of the hierarchy (Figure 11(b)): since diversity is measured by the distances of the nodes in the hierarchy, when the height increases, the diversity also increases. The height of the hierarchy does not affect the time cost of the summarization process for both *tRedux* and original alternatives (Figure 11(d)).

Effect of the Number of Attributes in the Summary Figure 12 shows the effect of the number of attributes. Each of the plots includes results from many experiments (varying, as described in Section 6, two data sets, number of tuples in the table and *t-var*, number, sizes, and heights of hierarchies) in a single chart; these experiments are clustered in terms of the number of attributes in the data being summarized and trend lines are drawn to help observe the general trends. Figure 12(a) shows that, for both Adult (red line) and synthetic (dashed blue line) data sets, the amount of dilution increases with the number of attributes, but the



(a) (b) (c) (d)

Figure 8: (a) Dilution, (b) diversity, (c) partition size variance, and (d) time as a function of tuple value count variance



(a) (b) (c) (d)

Figure 9: (a) Dilution, (b) diversity, (c) partition size variance, and (d) time as a function of the number of tuples

loss due to *tRedux* stays more or less constant, $\leq 20\%$. On the synthetic data, diversity shows a $\sim 10\%$ increase in loss when the number of attributes increase from 1 to 3; on the Adult data set, however, the impact of the number of attributes is rather negligible. It is also interesting to note that, on the Adult data set, the loss in diversity due to the use of *tRedux* is very close to 0. As Figure 12(c) shows, the execution time gain due to the use of *tRedux* increases with the number of attributes; for the Adult set the gain increases from around 30% (i.e., $\sim 1.5\times$ as fast as the original scheme) for the case with a single attribute to more than 40% (i.e., almost $2\times$ as fast as the original scheme) for the case with three attributes.

7. CONCLUSIONS

In this paper we have shown that, by pre-processing input value hierarchies before they are used in metadata supported table summarization process, we can significantly reduce the summarization cost, without causing significant quality degradations in the resulting table summaries. We have also introduced a novel hierarchy summarization approach, *tRedux*, tailored towards this task and have shown that this approach provides the best time gain vs. quality trade-off against alternative schemes.

8. REFERENCES

[1] K. Selçuk Candan, H. Cao, Y. Qi, and M. L. Sapino, “Alphasum: size-constrained table summarization using value lattices,” in *Proc. EDBT*, 2009, pp. 96–107.

[2] S. Chaudhuri and U. Dayal, “An overview of data warehousing and olap technology.” *SIGMOD Record*, vol. 26, no. 1, pp. 65–74, 1997.

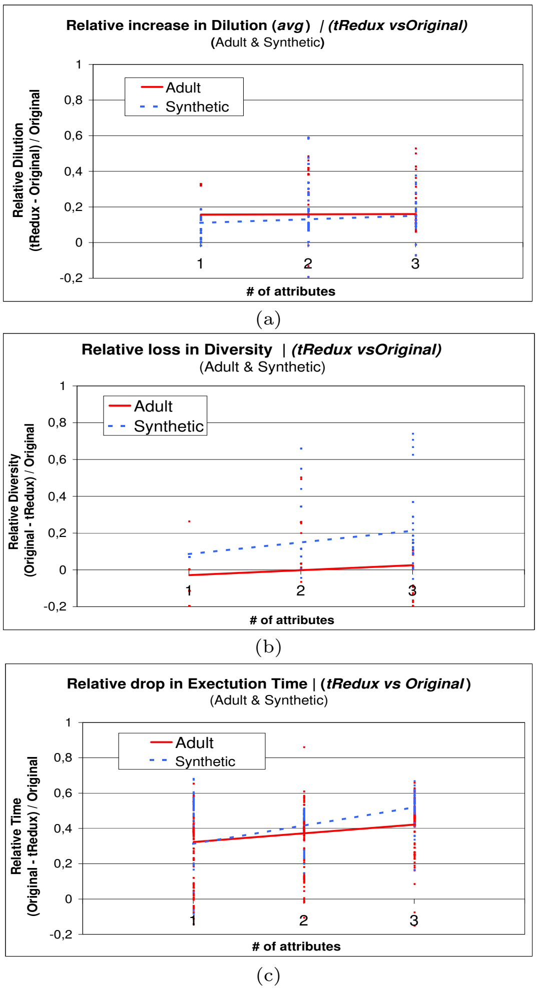
[3] P. Samarati, “Protecting respondents’ identities in microdata release,” *IEEE Trans. Knowl. Data Eng.*, vol. 13, no. 6, pp. 1010–1027, 2001.

[4] G. R. Régis Saint-Paul and N. Mouaddib, “General purpose database summarization,” in *Proc. VLDB*, 2005, pp. 733–744.

[5] ——, “Database summarization: The saintetiq system.” in *Proc. of ICDE*, 2007, pp. 1475–1476.

[6] R. Alfred and D. Kazakov, “Data summarization approach to relational domain learning based on frequent pattern to support the development of decision making,” in *ADMA*, 2006, pp. 889–898.

[7] M.-L. Lo, K.-L. Wu, and P. S. Yu, “Tabsum: A flexible and dynamic table summarization approach,” *ICDCS*, 2000.



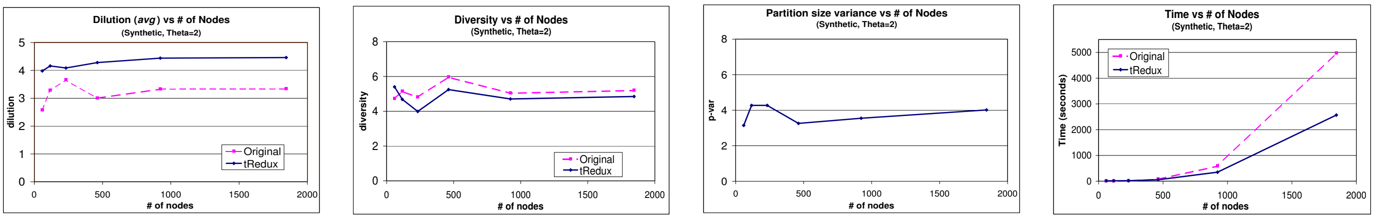
(a)

(b)

(c)

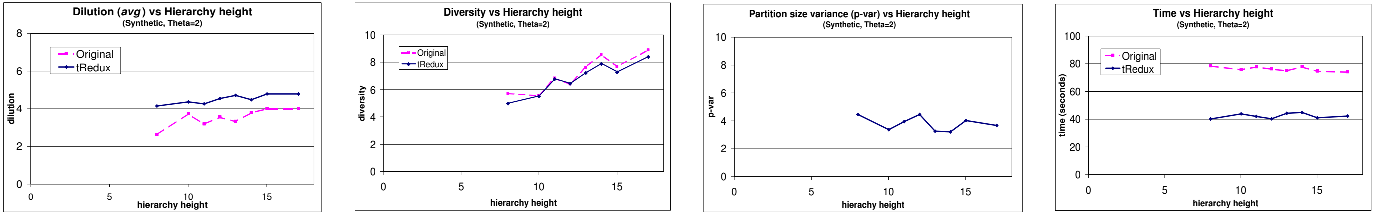
Figure 12: Dilution, diversity and relative execution time with varying number of attributes

[8] W. K. Ng and C. V. Ravishankar, “Relational database compression using augmented vector quantization,” in *ICDE*, 1995, pp. 540–549.



(a) (b) (c) (d)

Figure 10: (a) Dilution, (b) diversity, (c) partition size variance, and (d) time as a function of hierarchy size



(a) (b) (c) (d)

Figure 11: (a) Dilution, (b) diversity, (c) partition size variance, and (d) time as a function of hierarchy height

[9] A. L. Buchsbaum, G. S. Fowler, and R. Giancarlo, “Improving table compression with combinatorial optimization,” *J. ACM*, vol. 50, no. 6, pp. 825–851, 2003.

[10] F. Buccafurri, F. Furfaro, D. Sacca, and C. Sirangelo, “A quad-tree based multiresolution approach for two-dimensional summary data,” in *SSDBM’2003*, USA, 2003, pp. 127–140.

[11] A. Cuzzocrea, D. Saccà, and P. Serafino, “A hierarchy-driven compression technique for advanced olap visualization of multidimensional data cubes,” in *DaWaK*, 2006, pp. 106–119.

[12] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson, “Supporting imprecision in multidimensional databases using granularities,” in *SSDBM*, 1999, pp. 90–101.

[13] Y. Qi, K. S. Candan, J. Tatemura, S. Chen, and F. Liao, “Supporting olap operations over imperfectly integrated taxonomies,” in *Proc. of ACM SIGMOD*, 2008.

[14] G. Aggarwal, K. K. Tomas Feder, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu, “Approximation algorithms for k-anonymity,” *Journal of Privacy Technology*, 2005.

[15] A. Meyerson and R. Williams, “On the complexity of optimal k-anonymity,” in *Proc. PODS*, 2004, pp. 223–228.

[16] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, “Incognito: Efficient full-domain k-anonymity,” in *Proc. of ACM SIGMOD*, 2005, pp. 49–60.

[17] A. Fokoue, A. Kershenbaum, L. Ma, E. Schonberg, and K. Srinivas, “K.: The summary abox: Cutting ontologies down to size,” in *ISWC*, 2006, pp. 343–356.

[18] X. Zhang, G. Cheng, and Y. Qu, “Ontology summarization based on rdf sentence graph,” in *WWW*, USA, pp. 707–716.

[19] R. T. Ng and J. Han, “Clarans: A method for clustering objects for spatial data mining,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 5, pp. 1003–1016, 2002.

[20] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proc. of ACM SIG-KDD*, 1996, pp. 226–231.

[21] G. Karypis, E.-H. S. Han, and V. Kumar, “Chameleon: A hierarchical clustering algorithm using dynamic modeling,” *IEEE Computer*, 1999.

[22] S. Guha, R. Rastogi, and K. Shim, “Rock: A robust clustering algorithm for categorical attributes,” *Proc. of ICDE*, vol. 0, p. 512, 1999.

[23] N. Tishby and N. Slonim, “Data clustering by markovian relaxation and the information bottleneck method,” in *NIPS*, 2000, pp. 640–646.

[24] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis, “Graph clustering and minimum cut trees,” *Internet Mathematics*, vol. 1, pp. 385–408, 2004.

[25] R. Goldman and J. Widom, “Dataguides: Enabling query formulation and optimization in semistructured databases,” in *VLDB’97*, 1997, pp. 436–445.

[26] D. Rafiei, D. L. Moise, and D. Sun, “Finding syntactic similarities between xml documents,” *Database and Expert Systems Applications*, vol. 0, pp. 512–516, 2006.

[27] A. Nierman and H. V. Jagadish, “Evaluating structural similarity in xml documents,” in *WebDB*, 2002, pp. 61–66.

[28] S. Helmer, “Measuring the structural similarity of semistructured documents using entropy,” in *VLDB ’07*, 2007, pp. 1022–1032.

[29] R. J. Bayardo and R. Agrawal, “Data privacy through optimal k-anonymization,” in *Proc. of ICDE*, 2005, pp. 217–228.

[30] V. S. Iyengar, “Transforming data to satisfy privacy constraints,” in *Proc. of ACM SIG-KDD*, 2002, pp. 279–288.

[31] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, “l-diversity: Privacy beyond k-anonymity,” in *Proc. of ICDE*, 2006.

[32] G. Ghinita, P. Karras, P. Kalnis, and N. Mamoulis, “Fast data anonymization with low information loss,” in *Proc. VLDB*, 2007, pp. 758–769.

[33] J. Byun, A. Kamra, E. Bertino, and N. Li., “Efficient k-anonymization using clustering techniques,” in *DASFAA*, 2007.

[34] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. Fu, “Utility-based anonymization using local recoding,” in *Proc. of ACM SIG-KDD*, 2006, pp. 785–790.

[35] R. Rada, H. Mili, E. Bicknell, and M. Blettner, “Development and application of a metric on semantic nets,” vol. 19, pp. 17–30, 1989.

[36] R. Richardson, A. F. Smeaton, A. F. Smeaton, J. Murphy, and J. Murphy, “Using wordnet as a knowledge base for measuring semantic similarity between words,” AICS, Tech. Rep., 1994.

[37] J. W. Kim and K. S. Candan, “Cp/cv: concept similarity mining without frequency information from domain describing taxonomies,” in *CIKM ’06*, 2006, pp. 483–492.

[38] W. C. G.A. Miller, “Contextual correlates of semantic similarity,” *Language and Cognitive Processes*, vol. 6, no. 1, 1991.

[39] P. Resnik, “Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language,” *Journal of Artificial Intelligence Research*, vol. 11, pp. 95–130, 1999.

[40] S. Guha, R. Rastogi, and K. Shim, “CURE: an efficient clustering algorithm for large databases,” 1998, pp. 73–84. [Online]. Available: citeseer.ist.psu.edu/guha98cure.html

[41] G. Karypis and V. Kumar, “A fast and high quality multilevel scheme for partitioning irregular graphs,” *SIAM Journal on Scientific Computing*, vol. 20, pp. 359–392, 1998.

[42] D. Harel and Y. Koren, “On clustering using random walks,” in *FSTTCS*, 2001, pp. 18–41.

[43] A. Asuncion and D. Newman, “UCI machine learning repository,” 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html