Model-Based Semantic Compression for Network-Data Tables[*]

Shivnath Babu[†]
Stanford University
shivnath@cs.stanford.edu

Minos Garofalakis
Bell Laboratories
minos@bell-labs.com

Rajeev Rastogi
Bell Laboratories
rastogi@bell-labs.com

Avi Silberschatz
Bell Laboratories
avi@bell-labs.com

ABSTRACT

While a variety of lossy compression schemes have been developed for certain forms of digital data (e.g., images, audio, video), the area of lossy compression techniques for arbitrary data tables has been left relatively unexplored. Nevertheless, such techniques are clearly motivated by the ever-increasing data collection rates of modern enterprises and the need for effective, guaranteed-quality approximate answers to queries over massive relational data sets. In this paper, we propose *Model-Based Semantic Compression (MBSC)*, a novel data-compression framework that takes advantage of attribute semantics and data-mining models to perform lossy compression of massive data tables. We describe the architecture and some of the key algorithms underlying $\mathcal{SPARTAN}$, a model-based semantic compression system that exploits predictive data correlations and prescribed error tolerances for individual attributes to construct concise and accurate *Classification and Regression Tree (CaRT)* models for entire columns of a table. More precisely, $\mathcal{SPARTAN}$ selects a certain subset of attributes for which no values are explicitly stored in the compressed table; instead, concise CaRTs that predict these values (within the prescribed error bounds) are maintained. To restrict the huge search space and construction cost of possible CaRT predictors, $\mathcal{SPARTAN}$ employs sophisticated learning techniques and novel combinatorial optimization algorithms. Our experimentation with several real-life data sets has offered convincing evidence of the effectiveness of $\mathcal{SPARTAN}$'s model-based approach – $\mathcal{SPARTAN}$ is able to consistently yield substantially better compression ratios than existing semantic or syntactic compression tools (e.g., gzip) while utilizing only small data samples for model inference. Several promising directions for future research and possible applications of MBSC in the context of network management are identified and discussed.

1. INTRODUCTION

Many modern enterprises are finding the need to store and analyze massive and fast-growing tables of alphanumeric data. For example, Internet Service Providers (ISPs) continuously collect traffic information to enable key network traffic management applications. The collected data include: network packet and flow traces; active measurements of packet delay, loss, and throughput; router forwarding tables and configuration data; and *Simple Network Management Protocol* (SNMP) data maintained by various network elements (e.g., routers, switches) [6, 10]. Packet traces collected for traffic management in the Sprint IP backbone amount to 600 Gigabytes of data per day [10]. Telecommunication providers typically generate and store records of information, termed "Call-Detail Records" (CDRs), for every phone call carried over their network. A typical CDR is a fixed-length record structure comprising several hundred bytes of data that capture information on various (categorical and numerical) attributes of each call; this includes network-level information (e.g., endpoint exchanges), time-stamp information (e.g., call start and end times), and billing information (e.g., applied tariffs), among others [5]. These CDRs are stored in tables that can grow to truly massive sizes, in the order of several Terabytes per year. These massive tables of network-traffic and CDR data are continuously explored and analyzed to produce the "knowledge" that enables key network-management tasks, including application and user profiling, proactive and reactive resource management, traffic engineering, and capacity planning.

Data compression issues arise naturally in applications dealing with massive data sets, and effective solutions are crucial for optimizing the usage of critical system resources like storage space and I/O bandwidth, as well as network bandwidth (for transferring the data) [5, 10]. Several statistical and dictionary-based compression methods have been proposed for text corpora and multimedia data, some of which (e.g., Lempel-Ziv or Huffman) yield provably optimal asymptotic performance in terms of certain ergodic properties of the data source. These methods, however, fail to provide adequate solutions for compressing a massive data table, as they view the table as a large byte string and do not account for the complex dependency patterns in the table. Compared to conventional compression problems, effectively compressing massive tables presents a host of novel challenges due to several distinct characteristics of table data sets and their analysis.

• **Semantic Compression.** Existing compression techniques are "syntactic" in the sense that they operate at the level of consecutive bytes of data. Such syntactic methods typically fail to provide adequate solutions for table-data compression, since they essentially view the data as a large byte string and do not exploit the complex dependency patterns in the table. Effective table compression mandates techniques that are *semantic* in nature, in the sense that they account for and exploit both (1) existing data dependencies and correlations among attributes in the table; and, (2) the meanings and dynamic ranges of individual attributes (e.g., by taking advantage of the specified error tolerances).

• **Approximate (Lossy) Compression.** Due to the exploratory nature of many data-analysis applications, there are several scenarios in which an exact answer may not be required, and analysts may in fact prefer a fast, approximate answer, as long as the system can guarantee an *upper bound on the error of the approximation*. For example, during a drill-down query sequence in ad-hoc data mining, initial queries in the sequence frequently have the sole purpose of determining the truly interesting queries and regions of the data table. Thus, in contrast to traditional lossless data compression, the compression of massive tables can often afford to be *lossy*, as long as some (user- or application-defined) upper bounds on the compression error are guaranteed by the compression algorithm. This is obviously a crucial differentiation, as even small error tolerances can help us achieve much better compression ratios.

In this paper, we propose *Model-Based Semantic Compression (MBSC)*, a novel data-compression framework that takes advantage of attribute semantics and data-mining models to perform guaranteed-error, lossy compression of massive data tables. Abstractly, MBSC is based on the novel idea of exploiting data correlations and user-specified "loss"/error tolerances for individual attributes to construct concise data mining models and derive the best possible compression scheme for the data based on the constructed models. To make our discussion more concrete, we focus on the architecture and some of the key algorithms underlying $\mathcal{SPARTAN}$[1], a system that takes advantage of attribute correlations and error tolerances to build concise and accurate *Classification and Regression Tree (CaRT)* models [3] for entire columns of a table. More precisely, $\mathcal{SPARTAN}$ selects a certain subset of attributes (referred to as *predicted* attributes) for which no values are

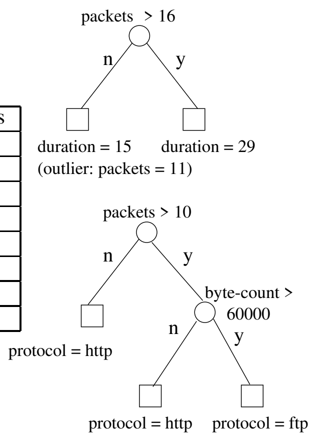
[*] An extended discussion of the technical results in this abstract appears in the *Proceedings of the ACM SIGMOD'2001 International Conference* [2].

[†] Work done while visiting Bell Laboratories.

[1] [From Webster] **Spartan:** */'spart-*n/* (1) of or relating to Sparta in ancient Greece, (2) a: marked by strict self-discipline and avoidance of comfort and luxury, b: sparing of words : TERSE : LACONIC.

protocol	duration	byte-count	packets
http	12	2,000	1
http	16	24,000	5
ftp	27	100,000	24
http	15	20,000	8
ftp	32	300,000	35
http	19	40,000	11
http	26	58,000	18
ftp	18	80,000	15

(a) Tuples in Table



(b) CaRT Models

Figure 1: Model-Based Semantic Compression.

explicitly stored in the compressed table; instead, concise CaRTs that predict these values (within the prescribed error bounds) are maintained. Thus, for a predicted attribute X that is strongly correlated with other attributes in the table, $\mathcal{SPARTAN}$ is typically able to obtain a very succinct CaRT predictor for the values of X, which can then be used to completely eliminate the column for X in the compressed table. Clearly, storing a compact CaRT model in lieu of millions or billions of actual attribute values can result in substantial savings in storage.

EXAMPLE 1.1. *Consider the table with 4 attributes and 8 tuples shown in Figure 1(a), where each tuple represents a* data flow *in an IP network. The categorical attribute* protocol *records the application-level protocol generating the flow; the numeric attribute* duration *is the time duration of the flow; and, the numeric attributes* byte-count *and* packets *capture the total number of bytes and packets transferred. Let the acceptable errors due to compression for the numeric attributes* duration*,* byte-count*, and* packets *be 3, 1,000, and 1, respectively. Also, assume that the* protocol *attribute has to be compressed without error (i.e., zero tolerance). Figure 1(b) depicts a regression tree for predicting the* duration *attribute (with* packets *as the predictor attribute) and a classification tree for predicting the* protocol *attribute (with* packets *and* byte-count *as the predictor attributes). Observe that in the regression tree, the predicted value of duration (label value at each leaf) is almost always within 3, the specified error tolerance, of the actual tuple value. For instance, the predicted value of* duration *for the tuple with* packets = *1 is 15 while the original value is 12. The only tuple for which the predicted value violates this error bound is the tuple with* packets = *11, which is an marked as an outlier value in the regression tree. There are no outliers in the classification tree. By explicitly storing, in the compressed version of the table, each outlier value along with the CaRT models and the projection of the table onto only the predictor attributes (*packets *and* byte-count*), we can ensure that the error due to compression does not exceed the user-specified bounds. Further, storing the CaRT models (plus outliers) for* duration *and* protocol *instead of the attribute values themselves results in a reduction from 8 to 4 values for* duration *(2 labels for leaves + 1 split value at internal node + 1 outlier) and a reduction from 8 to 5 values for* protocol *(3 labels for leaves + 2 split values at internal nodes).* ∎

To build an effective CaRT-based compression plan for the input data table, $\mathcal{SPARTAN}$ employs a number of sophisticated techniques from the areas of knowledge discovery and combinatorial optimization. Below, we list some of $\mathcal{SPARTAN}$'s salient features.

- **Use of Bayesian Network to Uncover Data Dependencies.** A Bayesian network is a directed acyclic graph (DAG) whose edges reflect strong predictive correlations between nodes of the graph [16]. $\mathcal{SPARTAN}$ uses a Bayesian network on the table's attributes to dramatically reduce the search space of potential CaRT models since, for any attribute, the most promising CaRT predictors are the ones that involve attributes in its "neighborhood" in the network.
- **Novel CaRT-selection Algorithms that Minimize Storage Cost.** $\mathcal{SPARTAN}$ exploits the inferred Bayesian network structure by using it to intelligently guide the selection of CaRT models that minimize the overall storage requirement, based on the prediction and materialization costs for each attribute. We demonstrate that this model-selection problem is a strict generalization of the *Weighted Maximum Independent Set (WMIS)* problem [11], which is known to be $\mathcal{NP}$-hard. However, by employing a novel algorithm that effectively exploits the discovered Bayesian structure in conjunction with efficient, near-optimal WMIS heuristics, $\mathcal{SPARTAN}$ is able to obtain a good set of CaRT models for compressing the table.
- **Improved CaRT Construction Algorithms that Exploit Error Tolerances.** Since CaRT construction is computationally-intensive, $\mathcal{SPARTAN}$ employs the following three optimizations to reduce CaRT-building times: (1) CaRTs are built using random samples instead of the entire data set; (2) leaves are not expanded if values of tuples in them can be predicted with acceptable accuracy; (3) pruning is integrated into the tree growing phase using novel algorithms that exploit the prescribed error tolerance for the predicted attribute. $\mathcal{SPARTAN}$ then uses the CaRTs built to compress the full data set (within the specified error bounds) *in one pass*.

We have implemented the $\mathcal{SPARTAN}$ system and conducted an extensive experimental study with three real-life data sets to compare the quality of compression due to $\mathcal{SPARTAN}$'s model-based approach with existing syntactic and semantic compression techniques. For all three data sets, and even for small error tolerances (e.g., 1%), we found that $\mathcal{SPARTAN}$ is able to achieve, on an average, 20-30% better compression ratios. Further, our experimental results indicate that $\mathcal{SPARTAN}$ compresses tables better when they contain more numeric attributes and as error thresholds grow bigger. For instance, for a table containing mostly numeric attributes and for higher error tolerances in the 5-10% range, $\mathcal{SPARTAN}$ outperformed existing compression techniques by more than a factor of 3. Finally, we show that our improved CaRT construction algorithms make $\mathcal{SPARTAN}$'s performance competitive, enabling it to compress data sets containing more than half a million tuples in a few minutes.

2. OVERVIEW OF APPROACH

2.1 Definitions and Notation

The input to the $\mathcal{SPARTAN}$ system consists of a n-attribute table T, and a (user- or application-specified) n-dimensional vector of *error tolerances* $\bar{e} = [e_1, \ldots, e_n]$ that defines the *per-attribute* acceptable degree of information loss when compressing T. Let $\mathcal{X} = \{X_1, \ldots, X_n\}$ denote the set of n attributes of T and $dom(X_i)$ represent the domain of attribute X_i. Intuitively, e_i, the i^{th} entry of the tolerance vector $\bar{e}$ specifies an upper bound on the error by which any (approximate) value of X_i in the compressed table T_c can differ from its original value in T. For a numeric attribute X_i, the tolerance e_i defines an upper bound on the *absolute difference* between the actual value of X_i in T and the corresponding (approximate) value in T_c. That is, if x, x' denote the accurate and approximate value (respectively) of attribute X_i for *any* tuple of T, then our compressor guarantees that $x \in [x' - e_i, x' + e_i]$. For a categorical attribute X_i, the tolerance e_i defines an upper bound on the *probability* that the (approximate) value of X_i in T_c is different from the actual value in T. More formally, if x, x' denote the accurate and approximate value (respectively) of attribute X_i for *any* tuple of T, then our compressor guarantees that $P[x = x'] \geq 1 - e_i$. (Note that our error-tolerance semantics can also easily capture *lossless* compression as a special case, by setting $e_i = 0$ for all i.)

2.2 Model-Based Semantic Compression

Briefly, our proposed *model-based* methodology for semantic compression of data tables involves two steps: (1) exploiting data correlations and (user- or application-specified) error bounds on individual attributes to construct data mining models; and (2) deriving a good compression scheme using the constructed models. We define the model-based, compressed version of the input table T as a pair $T_c = < T', \{\mathcal{M}_1, \ldots, \mathcal{M}_p\} >$ such that T can be obtained from T_c within the specified error tolerance $\bar{e}$. Here, (1) T' is a small (possibly empty) projection of the data values in T that are retained *accurately* in T_c; and, (2) $\{\mathcal{M}_1, \ldots, \mathcal{M}_p\}$ is a set of data-mining models. A definition of our general model-based semantic compression problem can now be stated as follows.

[Model-Based Semantic Compression (MBSC)]: Given a multi-attribute table T and a vector of (per-attribute) error tolerances $\bar{e}$, find a set of models $\{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$ and a compression scheme for T based on these models $T_c = < T', \{\mathcal{M}_1, \ldots, \mathcal{M}_p\} >$ such that the specified error bounds $\bar{e}$ are not exceeded and the storage requirements $|T_c|$ of the compressed table are minimized. ∎

Given the multitude of possible models that can be extracted from the data, the general MBSC problem definition covers a huge design space of possible alternatives for semantic compression. We now provide a more concrete statement of the problem addressed in our work on the $\mathcal{SPARTAN}$ system.

[$\mathcal{SPARTAN}$ CaRT-Based Semantic Compression]: Given a multi-attribute table T with a set of attributes $\mathcal{X}$, and a vector of (per-attribute) error tolerances $\bar{e}$, find a subset $\{X_1, \ldots, X_p\}$ of $\mathcal{X}$ and a set of corresponding CaRT models $\{\mathcal{M}_1, \ldots, \mathcal{M}_p\}$ such that: (1) model $\mathcal{M}_i$ is a predictor for the values of attribute X_i based solely on attributes in $\mathcal{X} - \{X_1, \ldots, X_p\}$, for each $i = 1, \ldots, p$; (2) the specified error bounds $\bar{e}$ are not exceeded; and, (3) the storage requirements $|T_c|$ of the compressed table $T_c = < T', \{\mathcal{M}_1, \ldots, \mathcal{M}_p\} >$ are minimized. ∎

Abstractly, $\mathcal{SPARTAN}$ seeks to partition the set of input attributes $\mathcal{X}$ into a set of *predicted attributes* $\{X_1, \ldots, X_p\}$ and a set of *predictor attributes* $\mathcal{X} - \{X_1, \ldots, X_p\}$ such that the values of each predicted attribute can be obtained within the specified error bounds based on (a subset of) the predictor attributes through a small classification or regression tree (except perhaps for a small set of outlier values). Note that we do not allow a predicted attribute X_i to also be a predictor for a different attribute. This restriction is important since predicted values of X_i can contain errors, and these errors can cascade further if the erroneous predicted values are used as predictors, ultimately causing error constraints to be violated. The final goal, of course, is to minimize the overall storage cost of the compressed table. This storage cost $|T_c|$ is the sum of two basic components:

1. *Materialization cost*, i.e., the cost of storing the values for all predictor attributes $\mathcal{X} - \{X_1, \ldots, X_p\}$. This cost is represented in the T' component of the compressed table, which is basically the projection of T onto the set of predictor attributes. The storage cost of materializing attribute X_i is denoted by $\texttt{MaterCost}(X_i)$.

2. *Prediction cost*, i.e., the cost of storing the CaRT models used for prediction plus (possibly) a small set of outlier values of the predicted attribute for each model. (We use the notation $\mathcal{X}_i \rightarrow X_i$ to denote a CaRT predictor for attribute X_i using the set of predictor attributes $\mathcal{X}_i \subseteq \mathcal{X} - \{X_1, \ldots, X_p\}$.) The storage cost of predicting attribute X_i using the CaRT predictor $\mathcal{X}_i \rightarrow X_i$ is denoted by $\texttt{PredCost}(\mathcal{X}_i \rightarrow X_i)$; this does *not* include the cost of materializing the predictor attributes in $\mathcal{X}_i$.

Fascicles. Our model-based semantic compression framework generalizes earlier ideas for semantic data compression, such as the very recent proposal of Jagadish, Madar, and Ng on using *fascicles* for the semantic compression of relational tables [12]. To the best of our knowledge, fascicles is the only work on lossy semantic compression of tables with guaranteed upper bounds on the compression error. A fascicle basically represents a collection of tuples (rows) that have *approximately* matching values for some (but not necessarily all) attributes (called *compact attributes*), where the degree of approximation is specified by user-specified error tolerances (similar to $\mathcal{SPARTAN}$). Essentially, fascicles are a specific form of data-mining models: *clusters in subspaces of the full attribute space*, where the notion of a cluster is based on the acceptable degree of loss during data compression.

EXAMPLE 2.1. *Consider the table in Figure 1(a) described in Example 1.1. Error tolerances of 3, 1,000 and 1 for the three numeric attributes* duration*,* byte-count *and* packets*, respectively, result in the following two fascicles (among others):*

F_1

http	**12**	*2,000*	*1*
http	**15**	*20,000*	*8*
http	**16**	*24,000*	*5*

F_2

ftp	**27**	*100,000*	*24*
ftp	**32**	*300,000*	*35*

The tuples in the two fascicles F_1 and F_2 are similar (with respect to the permissible errors) on the protocol *and* duration *attributes. (Two numeric attribute values are considered similar if the difference between them is at most twice the error bound for that attribute.) Substituting for each numeric attribute value, the mean of the maximum and minimum value of the attribute in a fascicle ensures that the introduced error is acceptable. Consequently, in order to compress the table using fascicles, the single (sub)tuple (http, 14) replaces the three corresponding (sub)tuples in the first fascicle and (ftp, 29.5)replaces the two subtuples in the second fascicle. Thus, in the final compressed table, the maximum error for* duration *is not greater than 3, and the number of values stored for the* protocol *and* duration *attributes is reduced from 8 to 5.* ∎

As the above example shows, in many practical cases, fascicles can effectively exploit the specified error tolerances to achieve high compression ratios. However, there exist several scenarios where more a general model-based compression approach is in order. Fascicles only detect "row-wise" patterns, where sets of rows have similar values for several attributes, and ignore "column-wise" patterns/dependencies (e.g., functional dependencies) across attributes in a table. On the other hand, data-mining models like CaRTs capture and model attribute correlations and, thereby, can attain much better semantic compression when such correlations exist. Revisiting Example 1.1, we see that CaRTs result in better compression than fascicles even for our toy example table[2] – the storage for the *duration* attribute reduces from 8 to 4 with CaRTs compared to 5 with fascicles.

We should note here that our proposed CaRT-based compression methodology is essentially *orthogonal* to techniques based on row-wise clustering, like fascicles. It is possible to combine the two techniques for an even more effective model-based semantic compression mechanism and is the subject of ongoing research. As an example, the predictor attribute table T' derived by our "column-wise" techniques can be compressed using a fascicle-based algorithm. (In fact, this is exactly the strategy used in our current $\mathcal{SPARTAN}$ implementation; however, other methods for combining the two are also possible.)

3. $\mathcal{SPARTAN}$ SYSTEM COMPONENTS

As depicted in Figure 2, the architecture of the $\mathcal{SPARTAN}$ system comprises of four major components: the DEPENDENCYFINDER, the CARTSELECTOR, the CARTBUILDER, and the ROWAGGREGATOR. In this section, we provide a detailed description of each $\mathcal{SPARTAN}$ component.

[2]This is not surprising given the strong correlations among the attributes in a table of network-traffic records [1].

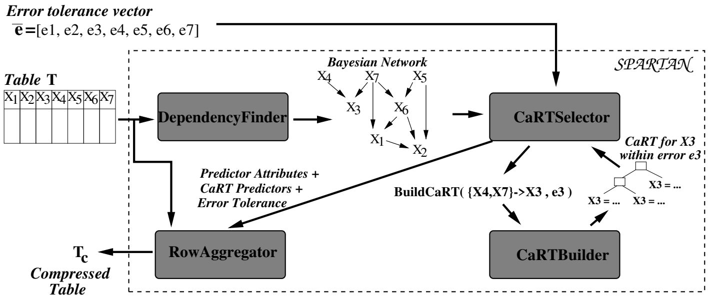


Figure 2: $\mathcal{SPARTAN}$ System Architecure.

3.1 The DEPENDENCYFINDER Component

The search space of $\mathcal{SPARTAN}$'s CaRT-based semantic compression problem is exponential: given any attribute X_i, *any subset* of $\mathcal{X} - \{X_i\}$ could potentially be used to construct a predictor for X_i! The DEPENDENCYFINDER component, therefore, produces an *interaction model* to identify the strongest predictive correlations among the input attributes so that CaRT construction can be limited to truly promising predictors. The current $\mathcal{SPARTAN}$ implementation uses *Bayesian networks*— a specific class of attribute interaction models that capture causal and/or predictive correlations in observational data in the form of a DAG over the attributes. The structure of the Bayesian network is based on the following dependence semantics:

• *Parental Markov Condition [16]:* Given a Bayesian network G over a set of attributes $\mathcal{X}$, any node $X_i \in \mathcal{X}$ is independent of all its non-descendant nodes given its parent nodes in G (denoted by $\pi(X_i)$).

• *Markov Blanket Condition [16]:* Given a Bayesian network G over a set of attributes $\mathcal{X}$, we define the *Markov blanket* of $X_i \in \mathcal{X}$ (denoted by $\beta(X_i)$) as the union of X_i's parents, X_i's children, and the parents of X_i's children in G. Any node $X_i \in \mathcal{X}$ is independent of all other nodes given its Markov blanket in G.

Based on the above conditions, candidates for CaRT-based semantic compression are restricted to predictors of the form $\pi(X_i) \to X_i$ or $\beta(X_i) \to X_i$. $\mathcal{SPARTAN}$'s DEPENDENCYFINDER component implements a *constraint-based* algorithm based on the techniques of Cheng et al. [7] to build the Bayesian network using a random sample of the data table. In the worst case, our algorithm requires $O(n^4)$ passes over the sampled tuples; thus, we ensure that the sample fits into memory so that no I/O is incurred (other than that required to produce the sample).

3.2 The CARTSELECTOR Component

The CARTSELECTOR implements the core algorithmic strategies for solving $\mathcal{SPARTAN}$'s CaRT-based semantic compression problem. Given the input data table and error tolerances, as well as the Bayesian network capturing the attribute interactions, the goal of the CARTSELECTOR is to select (1) a subset of attributes to be predicted and (2) the corresponding CaRT-based predictors, such that the overall storage cost is minimized within the specified error bounds. Deciding on a storage-optimal set of predicted attributes and corresponding predictors poses a hard combinatorial optimization problem: even in the simple case where the set of predictor attributes to be used for each attribute is fixed, the problem reduces to the *Weighted Maximum Independent Set (WMIS)* problem [2], which is known to be $\mathcal{NP}$-hard and notoriously difficult to approximate for general graphs [11], Given the inherent difficulty of the CaRT-based semantic compression problem, $\mathcal{SPARTAN}$'s CARTSELECTOR implements two algorithms based on heuristic search using the Bayesian network model of T built by the DEPENDENCYFINDER component.

The Greedy CaRT Selector. $\mathcal{SPARTAN}$'s **Greedy** CaRT-selection algorithm visits each attribute X_i in the topological-sort order imposed by the constructed Bayesian network G and tries to build a CaRT predictor for each attribute based on its predecessors. If X_i has no parent nodes in G, then **Greedy** directly places X_i in the subset of materialized attributes $\mathcal{X}_{mat}$. Otherwise (i.e., $\pi(X_i)$ is not empty in G), $\mathcal{SPARTAN}$'s CARTBUILDER component is invoked to construct the predictor $X_{mat} \to X_i$, where X_{mat} is the current set of attributes chosen for materialization. X_i is chosen to be predicted if $\frac{\texttt{MaterCost}(X_i)}{\texttt{PredCost}(X_{mat} \to X_i)} > \theta$, for an input parameter θ.

Although, **Greedy** constructs *at most* $(n-1)$ CaRT predictors, it suffers from two major shortcomings: (1) if it decides to predict attribute X_i, it ignores the potential benefits of using X_i itself as a (materialized) predictor attribute for its descendants in G; and, (2) selecting a reasonable value for θ is non-trivial.

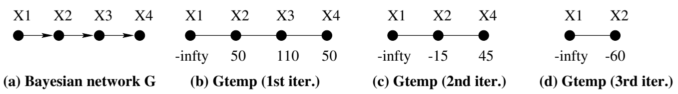
procedure MaxIndependentSet($T(\mathcal{X})$, $\bar{e}$, G , $\texttt{neighborhood}()$)
Input: n-attribute table T; n-vector of error tolerances $\bar{e}$; Bayesian network G on the set of attributes $\mathcal{X}$; function $\texttt{neighborhood}()$ defining the "predictive neighborhood" of a node X_i in G (e.g., $\pi(X_i)$ or $\beta(X_i)$).
Output: Set of materialized (predicted) attributes $\mathcal{X}_{mat}$ ($\mathcal{X}_{pred} = \mathcal{X} - \mathcal{X}_{mat}$) and a CaRT predictor $\texttt{PRED}(X_i) \to X_i$ for each $X_i \in \mathcal{X}_{pred}$.
begin
1. $\mathcal{X}_{mat} := \mathcal{X}$, $\mathcal{X}_{pred} := \phi$
2. $\texttt{PRED}(X_i) := \phi$ for all $X_i \in \mathcal{X}$, improve := **true**
3. **while** (improve $\neq$ **false**) **do**
4. **for each** $X_i \in \mathcal{X}_{mat}$
5. $\texttt{mater\_neighbors}(X_i) := (\mathcal{X}_{mat} \cap \texttt{neighborhood}(X_i)) \cup \{\texttt{PRED}(X) : X \in \texttt{neighborhood}(X_i), X \in \mathcal{X}_{pred}\} - \{X_i\}$
6. $\mathcal{M} := \texttt{BuildCaRT}(\texttt{mater\_neighbors}(X_i) \to X_i, e_i)$
7. let $\texttt{PRED}(X_i) \subseteq \texttt{mater\_neighbors}(X_i)$ be the set of predictor attributes used in $\mathcal{M}$
8. $\texttt{cost\_change}_i := 0$
9. **for each** $X_j \in \mathcal{X}_{pred}$ such that $X_i \in \texttt{PRED}(X_j)$
10. $\texttt{NEW\_PRED}_i(X_j) := \texttt{PRED}(X_j) - \{X_i\} \cup \texttt{PRED}(X_i)$
11. $\mathcal{M} := \texttt{BuildCaRT}(\texttt{NEW\_PRED}_i(X_j) \to X_j, e_j)$
12. set $\texttt{NEW\_PRED}_i(X_j)$ to the (sub)set of predictor attributes used in $\mathcal{M}$
13. $\texttt{cost\_change}_i := \texttt{cost\_change}_i + (\texttt{PredCost}(\texttt{PRED}(X_j) \to X_j) - \texttt{PredCost}(\texttt{NEW\_PRED}_i(X_j) \to X_j))$
14. **end**
15. **end**
16. build an undirected, node-weighted graph $G_{temp} = (\mathcal{X}_{mat}, E_{temp})$ on the current set of materialized attributes $\mathcal{X}_{mat}$, where:
17. (a) $E_{temp} := \{(X,Y) : \forall$ pair $(X,Y) \in \texttt{PRED}(X_j)$ for some X_j in $\mathcal{X}_{pred}\} \bigcup \{(X_i, Y) : \forall Y \in \texttt{PRED}(X_i)\ , X_i \in \mathcal{X}_{mat}\}$
18. (b) $\texttt{weight}(X_i) := \texttt{MaterCost}(X_i) - \texttt{PredCost}(\texttt{PRED}(X_i) \to X_i) + \texttt{cost\_change}_i$ for each $X_i \in \mathcal{X}_{mat}$
19. $S := \textbf{FindWMIS}(G_{temp})$
20. /* select (approximate) maximum $\texttt{weight}$ independent set in */
21. /* G_{temp} as "maximum-benefit" subset of predicted attributes */
22. **if** $(\sum_{X \in S} \texttt{weight}(X) \leq 0)$ **then** improve := **false**
23. **else** /* update $\mathcal{X}_{mat}$, $\mathcal{X}_{pred}$, and the chosen CaRT predictors */
24. **for each** $X_j \in \mathcal{X}_{pred}$
25. **if** ($\texttt{PRED}(X_j) \cap S = \{X_i\}$) **then**
26. $\texttt{PRED}(X_j) := \texttt{NEW\_PRED}_i(X_j)$
27. **end**
28. $\mathcal{X}_{mat} := \mathcal{X}_{mat} - S$, $\mathcal{X}_{pred} := \mathcal{X}_{pred} \cup S$
29. **end**
30. **end** /* while */
end

Figure 3: The MaxIndependentSet CaRT-Selection Algorithm.

The MaxIndependentSet CaRT Selector. $\mathcal{SPARTAN}$'s **MaxIndependentSet** CaRT-selection algorithm, depicted in Figure 3, alleviates the drawbacks of **Greedy** mentioned above. Intuitively, the **MaxIndependentSet** algorithm starts out by assuming all attributes to be materialized, i.e., $\mathcal{X}_{mat} = \mathcal{X}$ (Step 1), and then works by iteratively solving WMIS instances that try to improve the overall storage cost by moving the nodes in the (approximate) WMIS solution to the subset of predicted attributes $\mathcal{X}_{pred}$. Consider the *first iteration* of the main while-loop (Steps 3–30). Algorithm **MaxIndependentSet** starts out by building CaRT-based predictors for each attribute X_i in $\mathcal{X}$ based on X_i's "predictive $\texttt{neighborhood}$" in the constructed Bayesian network G (Steps 5–7); this `neighborhood` function is an input parameter to the algorithm and can be set to either X_i's parents or its Markov blanket in G. Then, based on the "predicted-by" relations observed in the constructed CaRTs, **MaxIndependentSet** builds a node-weighted, undirected graph G_{temp} on $\mathcal{X}$ with (a) all edges (X_i, Y), where Y is used in the CaRT predictor for X_i, and (b) weights for each node X_i set equal to the storage-cost benefit of predicting X_i (Steps 16–18). Finally, **MaxIndependentSet** finds a (near-optimal) WMIS of G_{temp} and the corresponding nodes/attributes are moved to the predicted set $\mathcal{X}_{pred}$ with the appropriate CaRT predictors (assuming the total benefit of the WMIS is positive) (Steps 19–29).

Later iterations of **MaxIndependentSet**'s main while-loop try to further optimize the initial WMIS solution by exploiting natural "transitivity" properties of predictive correlations: if X is a good predictor of Y, and Y is a good predictor of Z in the directed chain $X \rightarrow Y \rightarrow Z$, then X could be a good predictor of Z (i.e., only X needs to be materialized). For each currently materialized attribute X_i, **MaxIndependentSet** finds its "materialized neighborhood" in the Bayesian model G, that comprises for each node X in the `neighborhood` of X_i: (1) X itself, if $X \in \mathcal{X}_{mat}$ and (2) the (materialized) attributes currently used to predict X, if $X \in \mathcal{X}_{pred}$ (Step 5). A CaRT predictor for X_i based on its materialized neighbors is then constructed (Step 6). Now, since X_i may already be used in a number of predictors for attributes in $\mathcal{X}_{pred}$, we also need to account for the change in storage cost for these predictors when X_i is replaced by its materialized neighbors used to predict it; this change (denoted by $\texttt{cost\_change}_i$) is estimated in Steps 8–14. The node-weighted, undirected graph G_{temp} is then built on $\mathcal{X}_{mat}$ with the weight for each node X_i set equal to the overall storage benefit of predicting X_i, including $\texttt{cost\_change}_i$ (Steps 16-18). (Note that this benefit may very well be negative.) Finally, a (near-optimal) WMIS of G_{temp} is chosen and added to the set of predicted attributes $\mathcal{X}_{pred}$ with the appropriate updates to the set of CaRT predictors. Note that, since our algorithm considers the "transitive effects" of predicting each materialized node X_i in isolation, some additional care has to be taken to ensure that *at most one* predictor attribute from each already selected CaRT in $\mathcal{X}_{pred}$ is chosen at each iteration. This is accomplished by ensuring that all attributes belonging to a predictor set $\texttt{PRED}(X_j)$ for some $X_j \in \mathcal{X}_{pred}$ form a *clique* in the construction of G_{temp} (Step 17). Then, by its definition, the WMIS solution can contain at most one node from each such set $\texttt{PRED}(X_j)$. **MaxIndependentSet**'s while-loop continues until no further improvements on the overall storage cost are possible (Step 22).

MaxIndependentSet builds at most $O(\rho\frac{n^2}{2})$ CaRT predictors, where n is the number of attributes in $\mathcal{X}$ and ρ is an upper bound on the number of predictor attributes used for any attribute in $\mathcal{X}_{pred}$. Under assumptions slightly less pessimistic than the worst case, it can further be shown that **MaxIndependentSet** builds at most $O(\rho n \log n)$ CaRT predictors [2].



(a) Bayesian network G (b) Gtemp (1st iter.) (c) Gtemp (2nd iter.) (d) Gtemp (3rd iter.)

Figure 4: Example Instance for CARTSELECTOR Algorithms.

EXAMPLE 3.1. *Consider the Bayesian network graph defined over attributes $X_1, \ldots, X_4$ shown in Figure 4(a). Let the materialization cost of each attribute be 125. Further, let the prediction costs of CaRT predictors be as follows:*

$$\begin{array}{ll} \texttt{PredCost}(\{X_1\} \rightarrow X_2) = 75 & \texttt{PredCost}(\{X_2\} \rightarrow X_3) = 15 \\ \texttt{PredCost}(\{X_1\} \rightarrow X_3) = 80 & \texttt{PredCost}(\{X_2\} \rightarrow X_4) = 80 \\ \texttt{PredCost}(\{X_1\} \rightarrow X_4) = 125 & \texttt{PredCost}(\{X_3\} \rightarrow X_4) = 75 \end{array}$$

For $\theta = 1.5$, **Greedy** *gives $\mathcal{X}_{mat} = \{X_1, X_4\}$, and and $\mathcal{X}_{pred} = \{X_2, X_3\}$; the chosen predictors being $X_1 \rightarrow X_2$ and $X_1 \rightarrow X_3$. The storage cost of this solution is $125 + 75 + 80 + 125 = 405$.*

Further, suppose that the `neighborhood` *function for a node X_i is its parents. Figures 4 (b), (c), and (d), represent the graph G_{temp} during the first, second, and third iterations (respectively) of* **MaxIndependentSet**. *The algorithm stops after the third iteration with $\mathcal{X}_{mat} = \{X_1, X_2\}$ and $\mathcal{X}_{pred} = \{X_3, X_4\}$; the chosen predictors being $X_2 \rightarrow X_4$ and $X_2 \rightarrow X_3$. The total storage cost of this solution is $125 + 125 + 15 + 80 = 345$, which is, in fact, the optimal solution for this instance.* ∎

3.3 The CARTBUILDER Component

$\mathcal{SPARTAN}$'s CARTBUILDER component constructs a CaRT predictor $\mathcal{X}_i \rightarrow X_i$ for the attribute X_i with $\mathcal{X}_i$ as the predictor attributes. The CARTBUILDER's objective is to construct the minimum cost (storage space for the tree and outliers) CaRT model such that each predicted value (of a tuple's value for attribute X_i) deviates from the actual value by at most e_i, the error tolerance for attribute X_i. If attribute X_i is *categorical*, then CARTBUILDER builds a compact decision tree with values of X_i serving as class labels using decision-tree construction algorithms from [17]. If attribute X_i is *numeric*, then CARTBUILDER builds a regression tree using integrated building and pruning enabled through a novel technique for computing a lower bound on the cost of a yet-to-be-expanded subtree [2].

3.4 The ROWAGGREGATOR Component

$\mathcal{SPARTAN}$'s ROWAGGREGATOR component uses a fascicle-based algorithm [12] to further compress the table T' of predictor attributes. Since fascicle-based compression is lossy, ROWAGGREGATOR has to ensure that errors in predictor attributes do not ripple to predicted attributes. For a numeric predictor attribute X_i, define value v to be a split value for X_i if $X_i > v$ is a split condition in some CaRT $\mathcal{M}_i$ in T_c. In a fascicle, we say that an attribute X_i is compact if the range $[x', x'']$ of X_i-values in the fascicle, in addition to having width at most $2e_i$, also satisfies the property that for every split value v, either $x' > v$ or $x'' \leq v$. This stricter definition of compact attributes prevents errors in predictor attributes from rippling to the predicted attributes. The fascicle computation algorithms developed in [12] can be extended in a straightforward manner to accommodate our new definition [2].

4. OVERVIEW OF EXPERIMENTAL RESULTS

We have conducted an extensive empirical study whose objective was to compare the quality of compression due to $\mathcal{SPARTAN}$'s model-based approach with existing syntactic (gzip) and semantic (fascicles) compression techniques. We summarize the major findings of our study here and refer the reader to [2] for a detailed presentation.

• **Better Compression Ratios.** On all data sets, $\mathcal{SPARTAN}$ produces smaller compressed tables compared to gzip and fascicles. The compression due to $\mathcal{SPARTAN}$ is more effective for tables containing mostly numeric attributes, at times outperforming gzip and fascicles by a factor of 3 (for error tolerances of 5-10%). Even for error tolerances as low as 1%, the compression due to $\mathcal{SPARTAN}$, on an average, is 20-30% better than existing schemes.

• **Small Sample Sizes are Effective.** For the data sets, even with samples as small as 50KB (0.06% of one data set), $\mathcal{SPARTAN}$ is able to compute a good set of CaRT models that result in excellent compression ratios. Thus, small samples can be used to build the Bayesian network and CaRT models and speed up $\mathcal{SPARTAN}$ significantly.

• **Best Algorithms for $\mathcal{SPARTAN}$ Components.** The **MaxIndependentSet** CaRT-selection algorithm compresses the data more effectively that the **Greedy** algorithm. **MaxIndependentSet** with parents performs quite well across all data sets with respect to compression ratio as well as running time.

5. FUTURE RESEARCH DIRECTIONS

We strongly believe that our research on the $\mathcal{SPARTAN}$ system has only scratched the surface of model-based semantic compression and its potential applications in managing and exploring network management data. We are currently extending our work on MBSC in a number of different directions.

• **MBSC for Continuous Data Streams: The Online-$\mathcal{SPARTAN}$ System.** Network-management systems collect and store *continuous streams of data*. For example, ISPs continuously collect packet- and flow-level traces of traffic in their network to enable sophisticated network traffic management applications such as traffic demand computation, capacity planning and provisioning, and determining pricing plans [6]. This data arrives at the network-management station as a continuous stream and archived in massive and fast-growing data tables. The goal of the Online-$\mathcal{SPARTAN}$ system is to build on the MBSC ideas of $\mathcal{SPARTAN}$ to effectively compress such continuous data streams. Applying MBSC over streaming data introduces a number of novel technical challenges.

More specifically, the data characteristics (e.g., attribute correlations) of streams can vary over time. $\mathcal{SPARTAN}$'s methodology of capturing the characteristics of the input table using data mining models and deriving a good compression plan from the constructed models would have to become *online* and *adaptive* for the data-stream scenario. This means that the models have to be maintained online as new tuples arrive over the data stream [8] and the compression plan should adapt appropriately to changes in data characteristics. The goal, of course, is to maintain the best possible compressed representation of the data seen thus far.

• **Distributed MBSC Plans.** Many traffic management applications (e.g., traffic demand computation, capacity planning [6]) for ISP networks need a global view of the network traffic, requiring data to be collected from multiple monitoring points distributed over the network [4]. Given the massive volume of collected data, it is highly desirable to compress that data before it is transferred to data warehouses and repositories for processing and archival [10]. Suppose that MBSC is the compression method of choice for the data collected at each monitoring point. One approach would be to build data mining models based solely on the locally-collected data. However, these models are not guaranteed to capture the characteristics of the overall network traffic and might, therefore, result in compression schemes that are suboptimal for the overall traffic data – more sophisticated approaches are needed. Note that this problem relates to problems studied in the context of *distributed data mining*, especially to the problem of combining/consolidating mining models built using portions of a distributed data collection [13].

• **Other Network-Management Applications.** The general MBSC methodology of constructing data mining models to capture the characteristics of network data has many potential applications in network monitoring and traffic management apart from compression. For example, $\mathcal{SPARTAN}$'s DEPENDENCYFINDER component produces an *interaction model* (e.g., a Bayesian network) that identifies the strong predictive correlations in an input data collection. Such predictive correlations lie at the core of the *event-correlation/filtering* and *root-cause analysis* engines of modern network-management systems, where the goal is to filter out uninteresting, "secondary" events and identify the root cause of multiple fault indications (alarms) arriving from different points inside a large ISP network [18, 14]. Thus, $\mathcal{SPARTAN}$-derived models are a natural fit for this very important application. Furthermore, in an *online scenario*, data mining models that capture the current characteristics of input data streams can be very useful for network-monitoring applications such as *anomaly detection* (e.g, detecting intrusions and (distributed) denial-of-service attacks [9]).

• **New Techniques for MBSC.** The basic thesis of MBSC is that the model, or combination of models, which compresses a data table optimally, is dependent on the data table. The current implementation of $\mathcal{SPARTAN}$, which uses CaRT-based column-wise compression, followed by fascicles-based row-wise compression, is a specific instantiation of the general MBSC philosophy. More sophisticated combinations of row-wise and column-wise models are possible. As an example, $\mathcal{SPARTAN}$'s fascicles-based ROWAGGREGATOR component can be replaced by a column-wise differential-coding-based component [15, 5]. In the long run, we envision $\mathcal{SPARTAN}$ as a *self-tuning* semantic compression system that derives the best compression scheme from the input data table, without being restricted to a specific family of models.

Currently $\mathcal{SPARTAN}$ exploits *data semantics* in two ways: (1) by capturing data correlations; and, (2) by using user- or application-specified per-attribute error tolerances which specify the acceptable degree of information loss. Developing other interfaces to specify data semantics in a general fashion, and also MBSC techniques for exploiting these interfaces is an interesting open problem. It is desirable to keep the interface general and independent of specific nature of the data so that the underlying MBSC tools and techniques are widely applicable.

6. REFERENCES

[1] "NetFlow Services and Applications". Cisco Systems White Paper, 1999.

[2] S. Babu, M. Garofalakis, and R. Rastogi. "SPARTAN: A Model-Based Semantic Compression System for Massive Data Tables". In *Proc. of the 2001 ACM SIGMOD Intl. Conference on Management of Data*, Santa Barbara, California, May 2001.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *"Classification and Regression Trees"*. Chapman & Hall, 1984.

[4] Y. Breitbart, C.-Y. Chan, M. Garofalakis, R. Rastogi, and A. Silberschatz. "Efficiently Monitoring Bandwidth and Latency in IP Networks". In *Proc. of IEEE INFOCOM'2001*, Anchorage, Alaska, April 2001.

[5] A. L. Buchsbaum, D. F. Caldwell, K. Church, G. S. Fowler, and S. Muthukrishnan. "Engineering the Compression of Massive Tables: An Experimental Approach". In *Proc. of the 11th Annual ACM-SIAM Symp. on Discrete Algorithms*, San Francisco, California, January 2000.

[6] R. Caceres, N. Duffield, A. Feldmann, J. Friedmann, A. Greenberg, R. Greer, T. Johnson, C. Kalmanek, B. Krishnamurthy, D. Lavelle, P. Mishra, K.K. Ramakrishnan, J. Rexford, F. True, and J. van der Merwe. "Measurement and analysis of IP network usage and behavior". *IEEE Communications Magazine*, May 2000.

[7] J. Cheng, D. A. Bell, and W. Liu. "Learning Belief Networks from Data: An Information Theory Based Approach". In *Proc. of the 6th Intl. Conference on Information and Knowledge Management*, Las Vegas, Nevada, November 1997.

[8] P. Domingos and G. Hulten. "Mining high-speed data streams". In *Proc. of the 6th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, Boston, Massachusetts, August 2000.

[9] N.G. Duffield and M. Grossglauser. "Trajectory Sampling for Direct Traffic Observation". In *Proc. of ACM SIGCOMM*, September 2000.

[10] C. Fraleigh, S. Moon, C. Diot, B. Lyles, and F. Tobagi. "Architecture of a Passive Monitoring System for Backbone IP Networks". Technical Report TR00-ATL-101-801, Sprint Advanced Technology Laboratories, October 2000.

[11] M.R. Garey and D.S. Johnson. *"Computers and Intractability: A Guide to the Theory of NP-Completeness"*. W.H. Freeman, 1979.

[12] H.V. Jagadish, J. Madar, and R. Ng. "Semantic Compression and Pattern Extraction with Fascicles". In *Proc. of the 25th Intl. Conference on Very Large Data Bases*, Edinburgh, Scotland, September 1999.

[13] H. Kargupta and P. Chan. *"Advances in Distributed and Parallel Knowledge Discovery"*. MIT Press, Cambridge, Massachusetts, 2000.

[14] I. Katzela and M. Schwarz. "Schemes for Fault Identification in Communication Networks". *IEEE/ACM Trans. on Networking*, 3(6):753–764, December 1995.

[15] W. K. Ng and C. V. Ravishankar. "Block-Oriented Compression Techniques for Large Statistical Databases". *IEEE Trans. on Knowledge and Data Engineering*, 9(2):314–328, March 1997.

[16] J. Pearl. *"Causality – Models, Reasoning, and Inference"*. Cambridge University Press, 2000.

[17] R. Rastogi and K. Shim. "PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning". In *Proc. of the 24th Intl. Conference on Very Large Data Bases*, New York, USA, August 1998.

[18] S. Yemini, S. Kliger, E. Mozes, Y. Yemini, and D. Ohsie. "High Speed & Robust Event Correlation". *IEEE Communications Magazine*, May 1996.