Multi-Structural Databases

Ronald Fagin R. Guha Ravi Kumar
Jasmine Novak D. Sivakumar Andrew Tomkins
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
{fagin, guha, ravi, jnovak, siva, tomkins}@almaden.ibm.com

ABSTRACT

We introduce the *Multi-Structural Database*, a new data framework to support efficient analysis of large, complex data sets. An instance of the model consists of a set of data objects, together with a schema that specifies segmentations of the set of data objects according to multiple distinct criteria (e.g., into a taxonomy based on a hierarchical attribute). Within this model, we develop a rich set of analytical operations and design highly efficient algorithms for these operations. Our operations are formulated as optimization problems, and allow the user to analyze the underlying data in terms of the allowed segmentations.

1. INTRODUCTION

Consider a large collection of media articles that could be segmented by a number of "dimensions."

- By time: articles from April 2004, or from 1978;
- By content type: articles from newspapers, or from magazines, and within magazines, from business magazines, or from entertainment magazines;
- By geography: articles from the U.S., or from Europe, and within Europe, from France or from Germany;
- By topic: articles about a war, a hurricane, or an election, and within these topics, by subtopic.

These different dimensions may be correlated; for instance, knowing the content type may give information about the topics. The first dimension in this list (time) can be viewed as numerical and as the example shows, we may be interested in intervals of time at various granularities. The other three dimensions in this list (content type, geography, and topic) can be viewed as hierarchical.

Because of the rich nature of the data (in this case, documents), we would be interested in probing the data with queries that are much richer than those in a typical database system. For example, we might be interested in the following types of queries.

- What are the ten most common topics in the collection?
- From 1990 to 1995, how did articles break down across geography and content type?
- Which subtopics caused the sudden increase in discussion of the war? Are these subtopics different among European newspapers?
- Break the early 1990s into ten subperiods that are most topically cohesive, and explain which topics were hot during each period.
- Break documents that mention the President by topic so that they are most closely aligned with different content types, to capture how different types of media with different audiences focused on particular topics.

One of the first things we notice about these queries is that they are rather fuzzy: there is not necessarily a clear "right answer." Even after we supply certain parameters (as we will do later), there may not be a unique answer. There is another way that our queries are unlike standard database queries. Our queries will be resolved not by, say, searching indices, but instead by an optimization procedure.[1] Most importantly, these queries seek to provide the user with the highlights or salient characteristics of the data set.

Our main contribution is a framework for expressing and computing the highlights of a large and complex data set. This includes a data model that is rich enough to represent the kinds of structures found in real applications. Within this model, we propose Pairwise Disjoint Collections (or PDCs, to be defined shortly) as a basic notion that captures the intuitive idea of a concise set of highlights of a data set. We formulate rich queries as optimization problems that map the underlying data into a succinct PDC within the allowed "schema." Finally, we provide efficient algorithms that give approximately optimal solutions for three important classes of objective functions. We illustrate the use of our framework with experimental results.

The basic framework.

We define a *multi-structural database* (MSDB) to consist of a schema (the set of dimensions, which, in our example, are time, content type, geography, and topic), and a set of

[1]Optimization procedures arise also in relational database systems, such as in trying to find the most efficient way to evaluate an SQL query. However, the answer to the SQL query is independent of the result of the optimization. In contrast, for our queries, the answer to the query is actually the result of the optimization procedure.

PODS June 13–15, 2005, Baltimore, MD.

objects (the data, which, in our example, are media articles).

The intent of the schema is to specify how the set of data objects may be segmented based on each dimension. Notice that each type of dimension admits partitions in a natural way, e.g., a numerical dimension may be partitioned into intervals, and a hierarchical dimension into collections of taxonomy nodes. Our first technical contribution is a unified approach to handling these diverse types. Namely, we formalize each dimension as a *lattice* — e.g., the geography dimension is a lattice whose elements include Europe and France, and the time dimension is a lattice whose elements are intervals at various granularities.

As usual, a lattice is endowed with *meet* and *join* (see Section 3 for definitions). For example, in the geography dimension, the join of the European countries is Europe, and in the time dimension, the meet of two time intervals is their intersection. Each data element (in our example, each document) is associated with various elements in the lattice. When a data item is associated with a lattice element, it is also associated with every element above it in the lattice; for example, a document about France is also a document about Europe. Lattice elements whose meet is bottom, the minimal element of a lattice, are thought of as disjoint concepts. For example, in the topics dimension, Sports and Politics are thought of as disjoint concepts. However, it is important to note that there may be a document associated with both of these concepts (such as a document about a bodybuilder who is also a governor).

We shall also consider new lattices that are the result of taking the product of a subset of our lattices. For example, "French newspaper" is an element of the lattice that is the product of the geography lattice and the content type lattice.

Besides unifying various types of allowed segmentations of data, our lattice-based approach offers the additional benefit of being able to concisely describe subcollections of the data. Elements of a lattice are restrictions like "French newspaper," and hence can be easily understood by a user in the terms used to define the schema. Indeed, the fundamental collection of data objects in our framework is the set of data objects associated with a lattice element.

The next key notion in our framework is that of a *pairwise disjoint collection (PDC)*. This is a set of elements of a lattice whose pairwise meet is bottom. A PDC thus represents a collection of conceptually disjoint concepts, and will be our model of what a data analysis operation returns to the user.

Analytical operations and algorithms.

Analogous to the way we developed an enhanced data model to support data analysis, we also enhance the notion of what constitutes a "query." As mentioned earlier, our analytical operations will be formulated as optimization problems that map the set of data objects into a PDC within the schema. In the present paper, we develop the formalism and algorithms for three main query families, each of which supports a fairly general data analysis operation. We believe that our model is fairly powerful, and anticipate many more rich queries to be developed within our framework.

The DIVIDE operation seeks to determine how a set of objects is distributed across a particular set of dimensions. The goal of the DIVIDE operation is to return a small PDC in which the documents are divided as evenly as possible among the elements of the PDC with respect to the set of dimensions; thus, DIVIDE provides a concise summary of the data. For example, a DIVIDE operation with respect to geography, where the goal is to partition into ten approximately equal-sized pieces, might show that 11% of documents come from North America, 9% come from Western Europe, and so on.

The DIFFERENTIATE operator allows the user to compare two different document sets with respect to certain dimensions. For example, a user could observe that there are many more blog entries in July than the rest of the year, and could employ DIFFERENTIATE to determine how much particular topics or geographies explain the spike. Thus, the DIFFERENTIATE operation helps us to find surprises in the data.

The goal of the DISCOVER operation is to break up a document set so that an internal pattern becomes visible. The operation segments the data using one set of dimensions such that the resulting segments are cohesive and well-separated according to another set of dimensions. For example, a user might ask if there is a way to break up newspaper articles by topic so that a strong geographic bias emerges. Thus, the DISCOVER operation helps us to find interesting "regions" and structure in the data.

While our lattice-based formulation is quite general, for the sake of obtaining efficient algorithms, we focus on two special cases of dimensions: numerical and hierarchical. These two special cases arise frequently in practice and hence it is important to develop efficient algorithms for these. We first show that in general it is NP-hard to approximate any of our analytic operations. On the other hand, we obtain exact algorithms for the single-dimensional case and approximation algorithms for the multi-dimensional case. Most of our algorithms are based on dynamic programming and can be implemented very efficiently.

We have implemented these operations in a prototype system, and created three databases within this system. The first database consists of pages from the World Wide Web; the second contains information about books and their sales ranks at `amazon.com` over time; and the third contains medical articles from medline. In all cases, a set of appropriate hierarchical and numerical dimensions are defined. We give a set of representative results showing that the operations are both natural and useful on real-world data.

2. RELATED WORK

2.1 Online Analytical Processing

Our formulation maps directly to those that have been used in OLAP [4]. A traditional formulation (for example, that of Kimball [19], or see [28] for a survey) distinguishes between *measures*, which are numerical and are the target of aggregations, and *dimensions*, which are often hierarchical. However, this distinction is not required, and there are formulations in which both are treated uniformly, as in our system. See Agrawal et al. [1], and Gyssens and Lakshmanan [14], for such examples.

Further, Harinarayan et al. [16] describe a formulation in which each dimension is a lattice, and a joint lattice is defined over multiple dimensions; this formulation exactly matches our model, except that our goal is to perform optimizations over the resulting multi-dimensional lattice, rather than to characterize the set of possible queries and hence the candidate materializations.

Gray et al. [10] describe, in the context of the cube operator, aggregation functions that may be distributive, algebraic, or holistic. Our algorithms rely on similar properties of the scoring functions we employ to determine the quality of a particular candidate solution to our optimization problems. We consider both distributive and algebraic scoring functions.

Cody et al [5] consider the "BIKM" problem of unifying business intelligence and knowledge management, by creating data cubes that have been augmented by additional information extracted through text analysis. They use an OLAP model similar to ours in that the granularity of the fact table is a single document.

Much work on OLAP considers materialization of particular views in order to improve query performance. These questions are relevant in our setting, but we do not consider them in this work.

Cabibbo and Torlone [3] propose that OLAP systems should support multiple query frameworks, possibly at different levels of granularity. There have been a number of such approaches suggesting frameworks that move beyond traditional hypothesis-driven query models into discovery-driven models. Han [15] and others have considered data mining to extract association rules on cubes. Sarawagi et al. [24, 25, 26] identify areas of the data space that are likely to be surprising to the user. Their mechanisms allow the user to navigate the product lattice augmented with indicators suggesting which cells at a particular location are surprising, and which paths will lead to surprising cells. Their visualization applies to our techniques, and their notion of "surprisingness" may be viewed as an operator to which our techniques apply. Like us, they produce exceptions at all levels of the cube, rather than simply at the leaves.

Lakshmanan et al. [20] consider partitions of the cube in order to summarize the "semantics" of the cube; all elements of a partition belong to the same element of a particular type of equivalence relation. This partitioning is similar in spirit to our notion of a complete PDC.

A few properties of our model that (are not necessarily distinguishing, but nevertheless) should be kept in mind when comparing to traditional OLAP systems include the following. Typically, an OLAP fact table contains a single entry for each cell of the cube; this is typically not the case in our setting, or in that of [5]. Also, our objects may belong to multiple locations within a dimension, which is typically not the case in OLAP (with certain lattices corresponding to time as notable exceptions). Our dimensions are typically not required to be leveled or fixed-depth. And we often consider single dimensions in which the size of the lattice is quadratic in the number of distinct leaves (all intervals of a line, for example).

Additionally, there is a fundamental distinction between our work and OLAP, which relates to the nature of queries rather than to the data model. OLAP queries typically specify certain nodes of the multi-dimensional lattice for which results should be computed; these nodes might be all the months of 2003 crossed with all the products in a certain category, for example. Our goal instead is to leave the choice of nodes to return as a constrained set of choices available to the algorithm, and to cast the problem of selecting a set of nodes as an optimization.

2.2 Clustering and mining

Some of our operations, especially DISCOVER, can be viewed as solving classification or clustering problems. While there is a vast literature on clustering involving numerical attributes, only recently has the problem of clustering involving a combination of numerical and categorical attributes gained much attention. Systems such as ROCK [13], CACTUS [8] and COOLCAT [2] provide algorithms for clustering data described by a combination of numerical and categorical attributes. In contrast to these, which assume a flat space of categorical attributes, MSDB goes one step further in using a lattice structure of relationships between the different values for each dimension. The addition of this structure allows us to restrict the clusters identified to correspond to nodes that are already in the lattice, which make the clusters easier for the user to understand.

Additionally, our goal of finding useful patterns in a data set is similar to the goals of data mining. While much of the attention in data mining has been on inducing association rules, the work reported in [7, 9, 21] has considered the problem of trend discovery and analysis. We continue towards this goal by enriching the underlying data model and generalizing the notion of a trend.

Finally, the algorithmic questions that arise in certain of our optimization problems are related to various questions that have been studied; we give pointers to this literature in the context of the algorithms themselves.

3. FORMULATIONS

A *lattice* is a representation of a partial ordering on a set of elements. It may be defined in terms of a partial order, or in terms of the operations *meet*, written $\wedge$, and *join*, written $\vee$. We give the latter definition since we will use this formulation more heavily. A lattice then is a set of elements closed under the associative, commutative binary operations meet and join, such that for all elements a and b, we have $a \wedge (a \vee b) = a \vee (a \wedge b) = a$. The lattice induces a natural partial order: $a \leq b$ if and only if $a \wedge b = a$. For a lattice L, we will write $a \in L$ to mean that a is an element of the set.

A lattice is bounded if it contains two elements $\top$ and $\bot$, called *top* and *bottom*, such that $a \wedge \bot = \bot$ and $a \vee \top = \top$ for all elements a. All finite lattices are bounded.

We will use $\#(A)$ to refer to the cardinality of set A.

3.1 MSDB

A *multi-structural database* (or simply *MSDB*) (X, D, R) consists of a universe $X = \{x_1, \ldots, x_n\}$ of objects, a set $D = \{D_1, \ldots, D_m\}$ of dimensions, and a membership relation R specifying the elements of each dimension to which a document belongs. We will treat each x_i as simply an identifier, with the understanding that this identifier may reference arbitrary additional data or metadata. In the following, we will often refer to objects as documents, as this is a key application domain. A dimension D_i is a bounded lattice, and we assume that the lattice nodes used in all lattices are distinct; the vocabulary $V = \cup_i D_i$ consists of all such lattice nodes. The membership relation $R \subseteq X \times V$ indicates that a data object "belongs to" a lattice element. We require that R be *upward closed*, i.e., if $\langle x, \ell \rangle \in R$ and $\ell \leq \ell'$, then $\langle x, \ell' \rangle \in R$. We define $X|_\ell$, read X *restricted to* ℓ, as $X|_\ell = \{x \in X \mid \langle x, \ell \rangle \in R\}$.

Similarly, we can define multiple dimensions to have the same structure as single dimensions. For nonempty $D' \subseteq D$, the *multi-dimension* $MD(D')$ is defined as follows. If D' is a singleton, the multi-dimension is simply the dimension of the single element. Otherwise, if $D' = \{D_1, \ldots, D_d\}$, then $MD(D')$ is again a lattice whose elements are $\{\langle \ell_1, \ldots, \ell_d \rangle \mid \ell_i \in L_i\}$, where $\langle \ell_1^1, \ldots, \ell_d^1 \rangle \vee \langle \ell_1^2, \ldots, \ell_d^2 \rangle = \langle \ell_1^1 \vee \ell_1^2, \ldots, \ell_d^1 \vee \ell_d^2 \rangle$, and likewise for $\wedge$. The membership relation R is then extended to contain $\langle x, \langle \ell_1, \ldots, \ell_d \rangle \rangle$ if and only if it contains $\langle x, \ell_i \rangle$ for all i. This lattice is sometimes called the *direct product* of the dimensional lattices [27].

Interpretation of lattice elements.

We think of each lattice element as a conceptual way of grouping together documents. All elements of the same lattice should represent conceptual groupings within the same logical family, for example, groupings of documents based on their topic. The meet of two conceptual groupings should be seen as the most general concept that is a specialization of both. For example, the meet of documents referring to events of the 1800s and documents referring to events from 1870-1930 should be documents referring to events from 1870-1899. Likewise, the join of two conceptual groupings should be seen as the most specific concept that is a generalization of both. The join of the two concepts described above should be all documents referring to events occurring between 1800 and 1930. Two concepts whose meet is $\perp$ should be seen as conceptually non-overlapping. It is possible that objects in the database could be part of both groups. For example, a dimension capturing a particular way of breaking documents into topics might contain a subtree about Sports, with a subnode about Basketball; and might also contain elsewhere a node about Politics. Just because a document appears that happens to discuss both Sports and Politics (because it is about sports legislation), this does not mean that the two topics must be conflated in the formalism. The membership relation allows a document to belong to two lattice elements whose meet is $\perp$. A strength of our formalism is that dimensions represent abstract conceptual groupings, but individual objects need not adhere exactly to these groupings. Of course, as individual documents belong to more and more conceptually disjoint regions of a dimension, the discriminative power of that dimension will diminish.

3.2 Pairwise disjoint collections

We now introduce our key notion, which will be used in the definition of all three analytical operations. Intuitively, a *pairwise disjoint collection*, abbreviated *PDC*, is a way of breaking a set of documents into conceptually non-overlapping pieces such that each piece can be easily described using the elements of a particular multi-dimension. Formally, for any multi-dimension $MD(D')$ and any set $S = \{\ell_1, \ldots, \ell_d\}$ of elements of the multi-dimension, we say that S is a PDC if $\ell_i \wedge \ell_j = \perp$ for all i, j with $i \neq j$.

So a PDC, in contrast to a general set of clusters, can be communicated easily and concisely to the user in terms that already exist in the schema; namely, the particular elements of $MD(D')$ that define each PDC member. Each of our analytical operations takes a multi-dimension, and other information, and returns a PDC over the given multi-dimension, using the other information to determine which PDC should be returned.

In general, we think of a PDC as a segmentation of the conceptual space of the database. For example, a PDC might contain both the Sports and Politics nodes of a dimension capturing topic, as these nodes meet at $\perp$ — they do not share any subconcepts. Of course, as discussed above, there may be a document that exists in both nodes at once because it discusses Sports Legislation.

As another example, consider two numerical dimensions, and the associated multi-dimension. A PDC in this multi-dimension corresponds to a tiling of the plane using axis-parallel rectangles.

We say a PDC is *complete* for a subset $X' \subseteq X$ of the objects if for every $x \in X'$, there is some ℓ in the PDC such that $\langle x, \ell \rangle \in R$; that is, every document belongs to some element of the PDC. We say that a PDC is complete to mean that it is complete for X.

The reader might ask why the dimensions of an MSDB are not simply formulated as a set system of documents with intersection and union, rather than a lattice; in this case, a PDC would be a collection of mutually non-overlapping sets of documents. This notion of PDC is problematic because as we discussed, documents in practice may belong to multiple conceptually distinct categories. We circumvent this problem by defining PDCs at the schema level, rather than the instance level.

Restricted classes of PDCs.

In many situations, conveying to the user an arbitrary PDC over a complex multi-dimension may be quite difficult, and we may choose to restrict the allowable PDCs. Likewise, in some situations, optimal instances of a restricted family of PDCs may be easier to compute than for general PDCs. We define three types of increasingly restrictive PDCs. First, we require a definition. Consider a PDC H over $MD(D')$, and assume dimension $D_i \in D'$. Let $H|_{D_i} = \{\ell_i \mid \langle \ell_1, \ldots, \ell_d \rangle \in H\}$ and for any element $\ell \in H|_{D_i}$, let $H|_{\overline{D_i}}(\ell) = \{\langle \ell_1, \ldots, \ell_{i-1}, \ell_{i+1}, \ldots, \ell_d \rangle \mid \langle \ell_1, \ldots, \ell_d \rangle \in H, \ell_i = \ell\}$.

General: A general PDC has no restrictions.

Sequential: Intuitively, a sequential PDC first breaks the data via a PDC in some dimension, then recursively subdivides each resulting data set using the next dimension, and so on. In our earlier example of two numerical dimensions, a PDC that is sequential will contain either horizontal or vertical strips, each of which may be broken respectively by arbitrary vertical or horizontal lines. Formally, a sequential PDC is defined recursively as follows.

A PDC over a single dimension is sequential.

A PDC H over $D' \subseteq D$ is sequential for an ordering $(D'_1, \ldots, D'_d)$ of the dimensions D' if $H|_{D'_d}$ is a PDC and $H|_{\overline{D'_d}}(\ell)$ is sequential for $(D'_1, \ldots, D'_{d-1})$ for all $\ell \in H|_{D'_d}$.

A PDC is sequential for D' if there exists an ordering of D' for which it is sequential.

Factored: Intuitively, a PDC is factored if it represents the cross-product of PDCs in each dimension. In our example of two numerical dimensions, a factored PDC is defined by a set of vertical and horizontal lines. Formally, a PDC H over $D' \subseteq D$ is factored if $H = H|_{D_1} \times \cdots \times H|_{D_{\#(D')}}$. Every factored PDC is a sequential PDC for every ordering of the dimensions.

A graphical representation of examples of the three types of PDCs for two numerical dimensions is shown in Figure 1.

3.3 Two important special cases

We now present two important special cases of dimensions that arise in practice. We will later show that our analytical operations can be performed efficiently for these special cases.

Numerical dimensions.

An important special type of dimension is a numerical dimension, corresponding to points on the real line. Consider a setting in which each element of X has a real-valued timestamp associated with it. The lattice elements of this dimension are the intervals of the real line, with meet and join of two elements defined as the largest interval belonging to both elements, and the smallest interval containing both elements, respectively.

If required, the metric corresponding to a line is simply the line distance $|x - y|$.

Hierarchical dimensions.

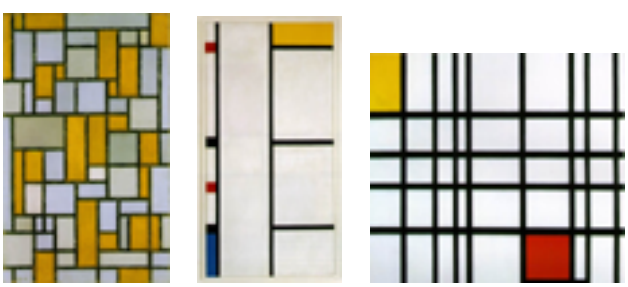
Another important special case is the hierarchical dimension, corresponding to a tree. As an example, consider the taxonomy of topics described above, in which top-level nodes correspond to high-level topics such as "Politics" or "Sports", while lower-level nodes correspond to more specific topics such as "The Republican Convention" (under Politics), or "Curling" (under Sports).

The corresponding lattice has an element for each node of the tree, plus a new node $\perp$ (as the root fulfills the requirement for $\top$). The meet of a set of of nodes is simply the largest node contained in all nodes of the set, and the join is the smallest node that contains all nodes in the set.

A PDC of a hierarchical dimension then corresponds to an antichain of the corresponding tree.

If a metric is required for this dimension, the metric is simply distance in the tree. This can easily be extended to include weighted distance, in which each edge of the tree is given a length.

For convenience, we will adopt specialized notation for hierarchical dimensions, as follows. Let T be a tree corresponding to some hierarchical dimension. Observe that there is a one-to-one correspondence between the nodes of T and the elements of the lattice, so we will speak of tree nodes for convenience. Let root(T) be the root of T, and $\#(T)$ be the number of nodes in T. If a is an internal node with degree Δ in T, we use $a_1, \ldots, a_\Delta$ to denote the Δ children of a. Let depth(T) be the depth of T.



General Sequential Factored

Figure 1: Types of PDCs, as envisioned by Piet Mondrian.

4. ANALYTICAL OPERATIONS

In this section, we describe a set of analytical operations over an MSDB. The operations we describe have specific definitions with concrete measures. However, MSDBs are not formulated solely to support these three operations. We anticipate both extensions and modifications of our analytical operations, and new operations entirely, over the same framework.

The three analytical operations we describe are intended to capture three common tasks in data analysis. All the operations begin with a set $X' \subseteq X$ of objects in an MSDB (X, D, R). For the purposes of illustration, we will adopt our running example of the MSDB of documents. Here $X =$ `Docs`, a large collection of documents, and the schema $D =$ `Dims` consists of four dimensions — `topic`, `time`, `geo`, and `source` (media type). We think of X' as being either the entire set, or a subset generated by one of the following mechanisms:

(1) X' consists of all objects that belong to some element of the multi-dimension; for example, all the documents from 1995 (interval of `time`), or all the European newspaper articles from June of 2004 (interval of `time` combined with European node of `geo` and the newspaper node of `source`);

(2) X' is the result of an MSDB analytical operation;

(3) X' is the result of an operation outside the MSDB, such as a keyword search or a SQL query, possibly even including other data.

In addition to requiring a subset $X' \subseteq X$, each operation also requires a subset $D' \subseteq D$ of dimensions, and a positive integer k. In all cases, we will use D' to break X' into k conceptually disjoint collections by finding a PDC of $MD(D')$ that optimizes some measure. The differences between the operations lies in the measure used to evaluate the PDC. One of the consequences of formulating the operations as optimization problems is that, unlike normal database operations, the result of our operations are not unique. Of course, this accords with our intuition that results of data analysis are almost never unique, and often depend on the algorithm that performs the analysis. Our formulation offers a principled approach in which even if the results are not unique, they are guaranteed to be optimal (or approximately optimal, in case we employ an approximation algorithm) with respect to well-defined objective functions.

The following list gives a high-level intuition for the operations:

DIVIDE: (*Summarize the data*) Provide a high-level understanding of the entire content of X', according to the partitions allowed by D'. The goal is primarily to get a gist of the data (X') from a particular perspective (that of D'). Specifically, we seek to partition the collection X' of objects into a small number of "pieces," where each piece has a succinct description as an element of $MD(D')$, and the pieces have more or less equal "volume."

DIFFERENTIATE: (*Find surprises in the data*) Find particular "regions," according to the segmentations allowed by D', where objects in X' occur significantly more (or significantly less) frequently than we would expect based on a "background set" $B \subseteq X$ of objects. The goal is primarily to find regions that are "surprising" in the sense that they differ from the user-specified benchmark B that represents the "expectation" of the user.

DISCOVER: (*Find structure in the data*) Partition X' into

pieces according to the segmentations allowed by D' such that the pieces represent cohesive, well-separated clusters when evaluated according to a metric on the lattice of another set $M \subseteq D$ of dimensions. The goal is to take the undifferentiated set X' of objects and find some way to break it up according to D' so that structure emerges, in the sense that each of the identified regions looks different through the lens of the measurement dimensions M.

We now formally define these operations.

4.1 DIVIDE

The DIVIDE operation allows the user to see at a high level how a set of objects is distributed across a particular set of dimensions. In our example MSDB, a DIVIDE operation with respect to `geo`, where the goal is to partition into ten approximately equal-sized pieces, might show that 11% of media documents come from North America, 9% come from Western Europe, etc. The goal of the DIVIDE operation is to identify a PDC from a specified set D' of dimensions that is complete for X'. On the one hand, we would like the PDC to be small, that is, to consist of a small number of elements of $MD(D')$, so that it may be conveyed to the user succinctly (e.g., on a screen); on the other hand, a PDC with ten elements, one of which contains 99% of the documents in X', is a poor choice since it does not give the user a good view of how the objects are spread out along D'. Ideally, we would prefer a PDC with ten sets, each of which contains 10% of the objects in X'. The DIVIDE operation allows the user to specify the size of the PDC, and computes the best PDC of that size.

Formalism

DIVIDE$(X, D; X', D', k)$
Input: $X' \subseteq X$; $D' \subseteq D$; $k \in \mathbb{Z}^+$
Output: A PDC H of $MD(D')$ of size k such that H is complete for X', and $\max_{h \in H} \#(X'|_h)$ is minimal over all such H.

Examples

The DIVIDE operation may be applied in our running example of the `Docs` MSDB in the following ways:

(1) To partition all documents from 2004 into ten roughly equal-sized collections, where each collection can be labeled by a specific topic, one may perform
DIVIDE(`Docs`, `Dims` $; X', \{$`topic`$\}, 10)$,
where $X' = $ `Docs`$|_{\texttt{time}=2004}$.

(2) To partition all documents into ten roughly equal-sized collections, where each collection can be labeled by a specific pair of time and geographic region, one may perform
DIVIDE(`Docs`, `Dims` $;$ `Docs`$, D', 10)$,
where $D' = \{$`time`, `geo`$\}$.

4.2 DIFFERENTIATE

The DIFFERENTIATE operation allows the user to compare two different sets of objects with respect to certain dimensions. This operation is useful when a user has two collections X' and B of objects — intuitively, the *foreground* and *background* collections — that the user knows (or believes) to be different in their distribution across some set D' of dimensions, and wishes to find out which regions of $MD(D')$ best explain the difference between the two collections.

In our (`Docs`, `Dims`) example, suppose a user observes that there are significantly more blogs in July than in the rest of the year; she might seek an explanation in terms of the other dimensions, namely `topic` and `geo`. Similarly, she may seek to explain the difference in the distribution of documents about information technology across various `source` media types from that of documents about management styles.

The DIFFERENTIATE operation assumes some measure μ that quantifies, for every element $h \in MD(D')$, the difference between $X'|_h$ and $B|_h$; intuitively, μ measures how unlike B the set X' is, from the viewpoint of h. For example, if 2% of all blogs are about art and there are 200K blog entries in July, one would expect about 4K of these blog entries to be about art; if, on the other hand, 13K blogs in July are about art, then the excess of 9K is a good indicator of how unlike the rest of the year July was for blog entries about art. With a measure μ in hand, DIFFERENTIATE seeks to find a small PDC — again, motivated by conciseness of the output returned to the user — such that the sum of the μ difference between X' and B over all the elements of the PDC is maximized. Thus, in our example, the user might learn that art, travel, and baseball form the best set of three topics that distinguish the July blogs from the rest.

Formalism

DIFFERENTIATE$(X, D; X', B, D', \mu, k)$
Input: $X' \subseteq X$; $B \subseteq X$; $D' \subseteq D$; $k \in \mathbb{Z}^+$;
$\mu : MD(D') \times 2^X \times 2^X \to \mathbb{R}$
Output: A PDC H of $MD(D')$ of size k such that $\sum_{h \in H} \mu(h; X', B)$ is maximal over all such H.

The Measure μ

We briefly discuss a suitable choice of the measure function μ that satisfies certain desirable conditions. Recall that the goal of defining μ is that, for a given $h \in MD(D')$ and foreground and background sets X' and B, the measure μ should quantify how unlike $B|_h$ the set $X'|_h$ is. Specifically, if we treat B as a set of objects that define a "baseline," for every set Y of objects, we expect roughly a fraction $\frac{\#(B|_h)}{\#(B)}$ of the objects in Y to be in $Y|_h$; thus, we expect roughly the same fraction of the objects in X' to be in $X'|_h$. When we are interested in explaining upward surges, a natural candidate for μ is the excess in $\#(X'|_h)/\#(X')$ over this quantity, namely we define the *peak measure* μ_p by

$\mu_p(h; X', B) = \frac{\#(X'|_h)}{\#(X')} - \frac{\#(B|_h)}{\#(B)}$.

If we are interested in explaining downward trends, we may use the analogous "valley measure," defined by $\mu_v(h; X', B) = -\mu_p(h; X', B)$. If we are interested in large upward surges and downward trends primarily for their magnitude, we may use the "absolute measure" $\mu_a(h; X', B) = |\mu_p(h; X', B)|$.

Note that the definition of μ_a simultaneously addresses two important considerations — how *surprising* $X'|_h$ is relative to $B|_h$, as well as how *impactful* $X'|_h$ is. It is obvious that to achieve a high value of $\left|\frac{\#(X'|_h)}{\#(X')} - \frac{\#(B|_h)}{\#(B)}\right|$, the fractions should differ substantially, that is, the collection X' should look surprisingly different from B with respect to h; furthermore, since (the absolute value of) this difference is upper-bounded by $\max\{\frac{\#(X'|_h)}{\#(X')}, \frac{\#(B|_h)}{\#(B)}\}$, it is easy to see that μ_a would only pick elements h for which either $\#(X'|_h)$ represents a large fraction of $\#(X')$ or $\#(B|_h)$ is a large fraction of $\#(B)$. Similar comments apply to μ_p and μ_v as well.

Examples

The DIFFERENTIATE operation may be applied in our running example of the `Docs` MSDB in the following ways:

(1) To find the top ten geographic regions in which the sport of rugby is covered more heavily than sports in general, one may perform

DIFFERENTIATE($\texttt{Docs}, \texttt{Dims}$; $X_r, X_s, D', \mu_p, 10$),

where $X_r = X|_{\texttt{topic=Rugby}}$, $X_s = X|_{\texttt{topic=Sports}}$, and $D' = \{\texttt{geo}\}$.

(2) To find the top three topics whose discussion decreased significantly, going from 2003 to 2004, one may perform

DIFFERENTIATE($\texttt{Docs}, \texttt{Dims}$; $X_4, X_3, D', \mu_v, 3$),

where $X_4 = X|_{\texttt{time=2004}}$, $X_3 = X|_{\texttt{time=2003}}$, and $D' = \{\texttt{topic}\}$.

(3) To find the five best combinations of topics and media types that occur very differently between Europe and Asia, one may perform

DIFFERENTIATE($\texttt{Docs}, \texttt{Dims}$; $X_E, X_A, D', \mu_a, 5$),

where $X_E = X|_{\texttt{geo=Europe}}$, $X_A = X|_{\texttt{geo=Asia}}$, and $D' = \{\texttt{topic}, \texttt{source}\}$.

(4) Suppose a user notices a spike in the discussion of war from July to November of 2002. To find the top ten combinations of media types and geographic regions that account for this spike, she may perform

DIFFERENTIATE($\texttt{Docs}, \texttt{Dims}$; $X_w, X_o, D', \mu_p, 10$),

where $X_w = X|_{\texttt{topic=War,time=[7/2002,11/2002]}}$, $X_o = X \setminus X_w$, and $D' = \{\texttt{geo}, \texttt{source}\}$.

4.3 DISCOVER

The goal of the DISCOVER operation is to unearth collections of objects according to some set D' of dimensions such that, viewed with respect to a second set M of "measurement" dimensions, the collections emerge as cohesive and well-separated "clusters". In our MSDB of documents, it is reasonable to expect that the collections of documents that correspond, respectively, to the topics "Sinn Fein," "Three Gorges Dam," "Yukos," and "Atlanta Falcons" exhibit a strong geographic bias. Similarly, it is reasonable to expect documents on topics "Ronald Reagan," "George Bush," and "Bill Clinton" to reveal a strong correlation to the `time` dimension. One may ask: given a collection $X' \subseteq X$ of objects and two sets D', M of dimensions, is it possible to *algorithmically discover* portions of X' with respect to D' that exhibit such strong localizations with respect to M? This is precisely what the DISCOVER operation accomplishes. Here we think of M as the set of "measurement dimensions."

As suggested by the above examples, some natural applications of this operation, in our MSDB of documents, might ask the following: What are the topics among European documents that emerge naturally as cohesive, well-separated clusters in the `time` dimension? What are the most significant pairs in the (`geo`, `time`) dimensions, such that documents about these geographic regions that appear in the corresponding time intervals happen to focus on largely disjoint topics? Note that the latter type of question offers an exciting interface into a document collection, namely to find news highlights in the space–time plane.

Formalism

DISCOVER$(X, D; X', D', M, \eta, k)$

Input: $X' \subseteq X$; $D' \subseteq D$; $M \subseteq D$; $k \in \mathbb{Z}^+$;
$\eta : 2^D \times 2^X \times MD(D') \to \mathbb{R}$

Output: A PDC H of $MD(D')$ of size k such that $\sum_{h \in H} \eta(M, X'; h)$ is maximal over all such H.

The Measure η

The quality measure η is defined as follows. A PDC H of $MD(D')$ is considered to be of high quality if it manifests two properties:

Cohesion: Objects that belong to the same h in H tend to be "nearby" in M;

Separation: Objects that belong to different h's in H tend to be "distant" in M.

To implement the notions of "nearby" and "distant," we also assume that for every subset $M \subseteq D$ of dimensions, we have a metric d_M on the set X of objects. To define the metric d_M on the set of objects, we may extend the natural metric on the lattice corresponding to M. For example, for a numeric dimension, we may define $d_M(x, y)$ as the width of the smallest interval that contains both x and y; for a hierarchical dimension, we may define the metric as the tree distance between two nodes, one of which contains x and the other one contains y, minimized over all such pairs of nodes. For multiple dimensions, we may take d_M to be a (possibly weighted) sum of the distance in each individual dimension of M.

Given such a metric, we will define, for $h \in MD(D')$, the *cohesion of* h by

$C(M, X'; h) = \frac{\sum_{x,y \in X'|_h} d_M(x,y)}{\#(X'|_h)^2}$.

Similarly, the *separation of* h is defined by

$S(M, X'; h) = \frac{\sum_{x \in X'|_h,\ y \in X' \setminus (X'|_h)} d_M(x,y)}{\#(X'|_h)\ \#(X' \setminus (X'|_h))}$.

Finally, we define

$$\eta(M, X'; h) = S(M, X'; h) - \gamma C(M, X'; h),$$

where $\gamma > 1$ sets the relative importance of these two desired properties; in our experiments, we take $\gamma = 2$. Generally, since $\gamma > 1$, it follows that h will have a positive value of η only if the average separation between an object in the cluster corresponding to h and an object not in that cluster is strictly more than the average separation between two objects within the cluster corresponding to h.

Examples

The DISCOVER operation may be applied in our running example of the `Docs` MSDB to accomplish the following trend discovery tasks.

(1) To find out if particular media types tend to cover particular sports more often than others, we may employ

DISCOVER($\texttt{Docs}, \texttt{Dims}; X_s, \{\texttt{source}\}, \{\texttt{topic}\}, \eta, 5$),

where $X_s = X|_{\texttt{topic=Sports}}$. This will compute the top five media types, each of which covers some sports more than others.

(2) To identify the the ten most significant pairs in the (`geo`, `time`) dimensions, where the collections of documents in each of these space–time regions focus on different topics, we may apply

DISCOVER($\texttt{Docs}, \texttt{Dims}; \texttt{Docs}, D', \{\texttt{topic}\}, \eta, 10$).

where $D' = \{\texttt{geo}, \texttt{time}\}$. This may reveal, for example, the collections of documents corresponding to the region `geo` = United States, `time` = [9/2001, 3/2002], with strong correlation the topic "terrorism," and the region `geo` = Europe, `time` = [5/2004, 6/2004], with correlation to the topic "soccer," etc.

(3) To find out if documents about President George W. Bush can be broken into pieces by topic so that each topic is correlated with some geographic region, we may perform

DISCOVER($\texttt{Docs}, \texttt{Dims}; X_b, \{\texttt{topic}\}, \{\texttt{geo}\}, \eta, 10$),

where $X_b = X|_{\texttt{topic=George W. Bush}}$.

5. ALGORITHMS AND COMPLEXITY

We now discuss algorithmic and hardness results for the three analytical operations. Some of the proofs for this section are deferred to the full version of this paper.

5.1 Hardness results

Using the hardness of approximating set cover [22, 6], we show:

THEOREM 1. *There is a constant $c > 0$ such that* DIVIDE *is NP-hard to approximate to within a factor of $c \log n$, where n is the number of objects in the database.*

This result can be strengthened for *factored* PDCs by using a result of Grigni and Manne [11] to show that DIVIDE is NP-hard to approximate to within a factor of 2. Next, using the hardness of approximating maximum clique [17], we show:

THEOREM 2. *For every constant $\epsilon > 0$,* DIFFERENTIATE *and* DISCOVER *are NP-hard to approximate to within a factor of $n^{1-\epsilon}$, where n is the number of lattice elements.*

5.2 Algorithms for a single dimension

We give dynamic programming algorithms to compute PDCs over a single numerical or hierarchical dimension. We observe that for all three operations, a weight can be defined for each element of the multi-dimension $MD(D')$ such that the quality of a resulting PDC can be written as the max (for DIVIDE) or sum (for DIFFERENTIATE and DISCOVER) over all elements of the weight of the element. We explicitly compute these weights in our algorithms.

THEOREM 3. *Given a single numerical dimension with n points,* DIVIDE, DIFFERENTIATE, *and* DISCOVER *can be solved exactly in polynomial time.*

PROOF. Let $v_1, \ldots, v_n$ be the distinct values of the numerical dimension, in ascending order. We will define the dynamic program over the indices $1, \ldots, n$. For each positive integer $k' \leq k$ and each index i, we consider all solutions that place k' non-overlapping intervals in the range from $1 \ldots i$, with the last interval ending exactly at i. We compute the optimal value $C(k', i)$ by considering all possible such last intervals. The dynamic program for each of the operations is described below.

For DIVIDE, we set the weight $w(a, b)$ of an interval $[a, b]$ to be $w(a, b) = \#(X'|_{[a,b]})$. Then the dynamic program is given by

$$C(k', i) = \min_{j=1}^{i-1} \left\{ \max \left\{ C(k' - 1, j), w(j + 1, i) \right\} \right\}.$$

For DISCOVER, we set the weight to be $w(a, b) = \eta(M, X'; [a, b])$, and the corresponding dynamic program is given by

$$C(k', i) = \max_{j=1}^{i-1} \{ C(k' - 1, j) + w(j + 1, i) \}.$$

For DIFFERENTIATE, we similarly set $w(a, b) = \mu(X'|_{[a,b]}; F, B)$ and use the same dynamic program as DISCOVER. For DIVIDE and DIFFERENTIATE, the corresponding weights can be computed in $O(\max\{\#(X'), n^2\})$ time and for DISCOVER, the weights can be computed in $\tilde{O}(\max\{\#(X'), n^2\} \cdot \#(M))$ time; in this version we omit the details of how to achieve these running times. Once the weights are computed, the best PDC using budget exactly k is $C(k, n)$; the running time of the dynamic program is therefore $O(kn^2)$. □

We note that a recent algorithm of Khanna et al. [18] can be used to to obtain an $O(n \log n)$ algorithm for DIVIDE for a single numerical dimension; we omit the details in this version.

We close our discussion of numerical dimensions with the following theorem regarding a simple approximation algorithm for DIVIDE. This theorem is based on a greedy algorithm that requires only a single scan of the data.

THEOREM 4. *Given a single numerical dimension with n points,* DIVIDE, *can be efficiently approximated to within a factor of 2 in time $O(n)$.*

Now we turn to algorithms for a single hierarchical dimension, and show the following theorem:

THEOREM 5. *Given a single hierarchical dimension implied by a tree T,* DIVIDE, DIFFERENTIATE, *and* DISCOVER *can be solved exactly in polynomial time.*

PROOF. Let T be the tree implied by the hierarchical dimension. The dynamic programming algorithm is driven by the following rule. Let $a = \text{root}(T)$ and let $a_1, \ldots, a_\Delta$ be the children of a. The best PDC of size at most k in T is either a itself, in which case none of the descendants of a can be included in the PDC, or is the union of the best PDCs $C_1, \ldots, C_\Delta$ of the subtrees rooted at $a_1, \ldots, a_\Delta$ respectively with the constraint that $\sum_{i=1}^{\Delta} |C_i| \leq k$. A naive implementation of this rule would involve partitioning k into Δ pieces in all possible ways and solving the dynamic program for each of the subtrees. This is expensive as a function of the degree Δ.

We address this problem by creating a binary tree T' from T with the property that the best PDC in T' corresponds to the best PDC in T. Construct T' top-down as follows. Each node a of T with more than two children $a_1, \ldots, a_\Delta$ is replaced by a binary tree of depth at most $\log \Delta$ with leaves $a_1, \ldots, a_\Delta$. The weights of $a, a_1, \ldots, a_\Delta$ are copied over from T and the weights of the internal nodes created during this process are set to ∞ for DIVIDE, and $-\infty$ for DIFFERENTIATE and DISCOVER. The construction is now repeated on $a_1, \ldots, a_\Delta$ to yield a weighted tree T'. It is easy to verify that the best PDC in T' is the same as the best PDC in T. Also, the tree size at most doubles, i.e., $\#(T') \leq 2 \cdot \#(T)$.

Since T' is binary, the dynamic programming algorithm to compute the best PDC in T' is more efficient. For each operation, let w be a function assigning a weight to each node of T', as shown in the following table:

DIVIDE	$w(a) = \#(X'|_a)$
DIFFERENTIATE	$w(a) = \mu(a; F, B)$
DISCOVER	$w(a) = \eta(M, X'; a)$

We compute the optimal solution using dynamic programming, as follows. Let $C(k', a)$ be the score of the best choice of k' incomparable nodes in the subtree rooted at node a of T'. We can fill in the entries of C using the following update rule:

$$C(k', a) = \begin{cases} w(a) & k' = 1 \\ \text{worstval} & k' > 1 \text{ and } a \text{ a leaf} \\ \text{bestsplit}(k', a) & \text{otherwise} \end{cases}$$

where worstval and bestsplit are operation-dependent.

In DIVIDE, we require a complete PDC, and the weight of the maximum node must be minimized; thus, we set

worstval $= \infty$ and define bestsplit as follows:

$$\text{bestsplit}(k', a) = \min_{k''=1}^{k'-1} \left\{ \max\{C(k'', a_1), C(k' - k'', a_2)\} \right\}.$$

For DIFFERENTIATE and DISCOVER, a complete PDC is not required, and the sum of the weights of all nodes in the PDC must be maximized. Thus, we instead set worstval $= -\infty$ and define bestsplit as follows:

$$\text{bestsplit}(k', a) = \max_{k''=0}^{k'} \left\{ C(k'', a_1) + C(k' - k'', a_2) \right\}.$$

Observe that the bounds for the min or max operator in the two variants of bestsplit range from 1 to $k' - 1$ in the case of DIVIDE, and from 0 to k' for the other operators. This implements the requirement that DIVIDE return a complete PDC by requiring that at least one unit of budget is sent to each child; the general PDC required for DIFFERENTIATE and DISCOVER allows zero units of budget to be sent to either child.

For DIVIDE and DIFFERENTIATE, it can be shown that the weights can be computed in $O(\max\{\#(X') \cdot \text{depth}(T), \#(T)\}$ time and for DISCOVER, the weights can be computed in $\max\{\#(X') \cdot \text{depth}(T), \#(M) \cdot \#(T) \cdot \max\{n, \#(T)\}\}$ time, where n is maximum number of distinct points in any dimension of M and T is the largest tree in M; we omit the details of these steps in this version. Once the weights are calculated, the dynamic program is computed for each $k' = 1, \ldots, k$ and for each $a \in T'$ and the final value is $C(k, \text{root}(T'))$. For a given k', a, computing $C(k', a)$ takes time k', which is at most k. Since the dynamic programming table is of size $k \cdot \#(T')$, the total running time is $k^2 \#(T') \leq 2k^2 \#(T)$. □

5.2.1 *Augmented* DIVIDE

Recall that a key characteristic of a multi-dimension is that each element may be easily and efficiently named in terms familiar to the user; for example, "European magazines" refers to an element of the multi-dimension over geography and media type. Consider a high-degree node such as "People" in a hierarchical dimension. If the budget k is smaller than the number of children of People then no complete PDC will ever contain a descendant of People. However, it is straightforward to convey to the user a PDC of three elements: People/Politicians, People/Sports Figures, and People/Other; and such a PDC maintains the desirable property that all nodes can be efficiently named—even though the meaning of People/Other is defined only in the context of the remaining nodes of the PDC. Thus, for operations like DIVIDE that require a complete PDC, it is desirable to allow "Other" nodes as part of the solution. We therefore introduce the *augmented* DIVIDE operation, an extension of DIVIDE on a single hierarchical dimension, to allow this type of solution.

First, we must formalize the intuition behind "Other" nodes. Consider a hierarchical dimension rooted at People, with children People/Politicians, People/Movie Stars, and People/Sports Figures. Assume that Politicians includes William Clinton and George Bush, while Sports Figures and Movie Stars each contain numerous children. Consider the following candidate PDC: {People/Sports Figures, People/Other}. Intuitively, this PDC is complete since all the politicians and movie stars are captured by the People/Other node. Now consider instead the following PDC: {People/Sports Figures, People/Politicians/William Clinton, People/Other}. We will consider this PDC to be incomplete, since the inclusion of People/Politicians/William Clinton means that the People/Other node no longer covers Politicians, and hence People/Politicians/George Bush is not covered. People/Other refers only to subtrees of the People node that are not mentioned at all in the PDC. Thus, an "Other" node of a given parent will cover the entire subtree of a child of that parent if and only if no other elements of the same subtree are present in the PDC.

Formally, for any hierarchical dimension T, we define a new tree aug(T) by adding `other` nodes to T as follows:

$$\text{aug}(T) = T \cup \{\texttt{t.other} \mid t \text{ is an internal node of } T\}.$$

Each node t.`other` has parent t and no children. Thus, every internal node of T now contains an `other` child. In the following, we will say that a is an *ancestor* of t if a can be obtained by applying the parent function to t zero or more times; thus, t is considered to be its own ancestor. We now consider an extended notion of complete PDCs over aug(T). As we observed above, the elements of T that are covered by a particular `other` node depend on the remainder of the PDC, so the behavior of an `other` node is defined only in a particular context. Fix $H \subseteq \text{aug}(T)$. We will first describe the behavior of the `other` nodes of H, and will then give the conditions under which H is a complete PDC for aug(T).

For each $h \in H$, define $C_H(h)$ to be the nodes of T covered by h, with respect to H, as follows. If h is a node of T; that is, if h is not an `other` node, then $C_H(h) = \{t \in T \mid h \text{ is an ancestor of } t\}$. This is the traditional definition of the subtree rooted at h, and requires no changes due to the presence of `other` nodes. Now consider nodes $h \in H$ such that $h = p$.`other`, so h is the `other` node corresponding to some node p. Then $C_H(h) = \{t \in T \mid \text{some child } c \text{ of } p \text{ is an ancestor of } t, \text{ and } c \text{ is not an ancestor of any element of } H\}$. This definition captures the idea that a subtree rooted at some child c of p is covered by p.`other` if and only if the subtree does not intersect H.

A subset $H \subseteq \text{aug}(T)$ is said to be a *complete augmented PDC* of T if and only if every leaf of T belongs to $C_H(h)$ for exactly one $h \in H$.

Finally, for every node $h \in H$, we define

$$\#_H(X', h) = \# \left(\bigcup_{t \in C_H(h)} X'|_t \right).$$

Formalism

Augmented DIVIDE($X, D; X', T, k$)

Input: $X' \subseteq X$; hierarchical dimension $T \in D$; $k \in \mathbb{Z}^+$

Output: A complete augmented PDC H of T of size k such that $\max_{h \in H} \#_H(X', h)$ is minimal over all such H.

This problem admits a highly efficient algorithm, as shown by the following theorem:

THEOREM 6. *Given a single hierarchical dimension T, and access to an oracle that can provide $\#(X'|_t)$ in constant time for every $t \in T$, the augmented* DIVIDE *problem with budget k can be solved optimally in time $O(k)$.*

5.3 Algorithms for multiple dimensions

In this section we discuss algorithms for multiple hierarchical and numerical dimensions.

Let d be the number of dimensions. Our first result states that for a given ordering on the dimensions, optimal sequential PDCs can be found for all three operations in polynomial time. The main idea is an iterative application of the optimal algorithm for the one-dimensional case.

THEOREM 7. *Given an ordering on the dimensions,* DIVIDE, DIFFERENTIATE, *and* DISCOVER *can be solved to find the optimal sequential PDC (under that ordering) in polynomial time.*

PROOF. (Sketch) For numerical dimensions, let $C(X', i, x, k)$ be the score of the best PDC on documents X' over the last i dimensions of the sequence, over the interval $(-\infty, x)$, with budget k. The dynamic program is $C(X', i, x, k) = \max_{\substack{y<x \\ 1\le j<k}} C(X', i, y, k-j) \circ C(X'|_{[y+1,x]}, i-1, \infty, j)$. Here $\circ$ stands for max in the case of DIVIDE, and for $+$ in the case of DIFFERENTIATE and DISCOVER. For hierarchical dimensions, let $C(X', i, a, k)$ be the score of the best PDC on documents X' over the last i dimensions of the sequence, over the subtree rooted at a, with budget k. We assume the tree has been made binary as in the proof of Theorem 5, and that a_1 and a_2 are the children of node a. Let r_j be the root of the j-th dimension from the last, according to the ordering. Then, $C(X', i, a, k) = \min\{C(X'|a, i-1, r_{i-1}, k), \min_{1\le j<k} C(X', i, a_1, j) \circ C(X', i, a_2, k-j)\}$. □

Finally, we note that for some special cases of general and factored PDCs, one can use approximation algorithms that have been developed in other contexts. For instance, Paluch [23] recently obtained a 17/8-approximation algorithm to cover a given array of numbers by rectangular tiles to minimize the maximum sum of each tile; this is equivalent to a general PDC for DIVIDE with two numerical dimensions. Another example is the algorithm of Khanna et al. [18], which can be used to obtain an $O(\log n)$ approximation to the factored PDC for DIVIDE.

6. EXPERIMENTS

We implemented a prototype multi-structural database system, and created a set of databases based on real-world data. In this section, we describe some uses of this system with the aim of providing a flavor for the kind of insight that can be gained by using the data model and operators described in this paper. Our examples are drawn from a range of domains in order to demonstrate the breadth of the formulation. The dimensions we employ are all either hierarchical or numerical.

6.1 Data

Web pages. We retrieved a set of approximately 5,000 web pages from IBM's WebFountain system [12], along with various metadata for each document. This metadata includes a list of entities, such as people and corporations, that occur on each page. Each entity occupies one or more nodes in a taxonomy. For example, the entity "George W. Bush" is included in the node "/People/Politics/US/Exec/President." Each page is also tagged as to which node in a source taxonomy it belongs. For example, a page from Time Magazine is included in the node "/Media/Magazines/General News." We retrieved a set of 5000 documents, each containing one or more entities, belonging to one or more source collections, and tagged with a date. There are two hierarchical dimensions, namely Entities (e.g., Politicians or Companies), and Source (e.g., Sports Magazines or Newspapers), and a numerical dimension, namely Date of Publication.

Medline articles. This dataset includes over 10,000 articles from Medline, most published during the year 2001. Each article is associated with a set of nodes in the MeSH (Medical Subject Headings) taxonomy created and maintained by the National Library of Medicine. There are two dimensions, namely a hierarchical dimension Topic (e.g., diseases or chemicals), and a numerical dimension Publication Date.

Amazon book sales. From Amazon's web service, we gathered information on several thousand books available on Amazon.com. We collected sales ranks for each book that entered the top 300 from July to October of 2004. We also gathered the date of publication, and the subject category of each book. The subject categories are organized into a taxonomy. For example, the book "A Christmas Carol" by Charles Dickens belongs to the category "Literature & Fiction/World Literature/British/Classics." Many books fall under multiple categories in the taxonomy. There is one hierarchical dimension, namely Subjects, and two numerical dimensions, namely Date of Publication and Sales Rank.

Subject ↓ Sales Rank →	1-521	522-1369	1370-2473
Subject/Business&Investing	155	137	117
Subject/Children's Books	149	118	79
Subject/Literature&Fiction	138	96	68
Subject/NonFiction	123	92	76
Subject/Other	882	1004	1107

Table 1: An example two-dimensional DIVIDE.

6.2 Operator examples

DIVIDE. We extracted a factored PDC using DIVIDE on the Amazon books database, where $X' = X$, the set of all objects, and D' contains the numerical Sales Rank dimension and the hierarchical Subject dimension. As Subject is hierarchical, we perform an augmented DIVIDE operation allowing `other` nodes. The factored PDC is computed by finding the optimal PDC for the Sales Rank dimension and the optimal augmented PDC for the Subject dimension, and then taking the cross-product of the resulting PDCs. Table 1 shows the results.

Observe that the categories chosen by the algorithm are all one level deep in the Subject tree. This reflects a good design on the part of Amazon: books are spread fairly evenly across the top-level categories of the tree. However, once the size of the Category PDC expands sufficiently (in this case, to 29 nodes), the system selects a non-uniform frontier through the tree, with some nodes deeper than others.

Also observe that the boxes in the table are not entirely uniform. There are two reasons for this. First, notice that the Subjects/Other category contains more content than the earlier rows. Although the one-dimensional PDC for the Subjects dimension alone is optimal for that dimension, the optimal solution need not contain a balanced split. In some cases, like this one, the tree will force especially uneven splits. Second, the factored PDC need not be optimal, and may even contain table cells with no documents whatsoever. However, the PDC does suffice to give a general idea of how the data is distributed within those dimensions.

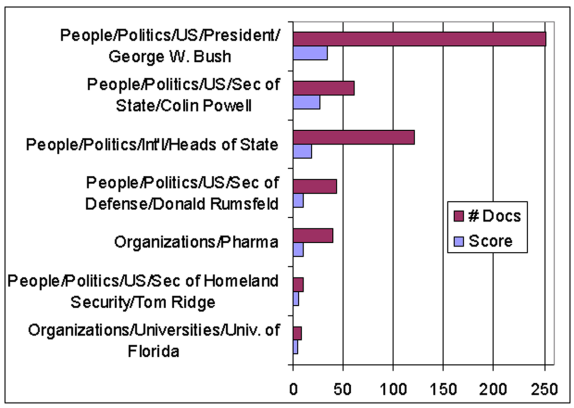


Figure 2: Example results from DIFFERENTIATE.

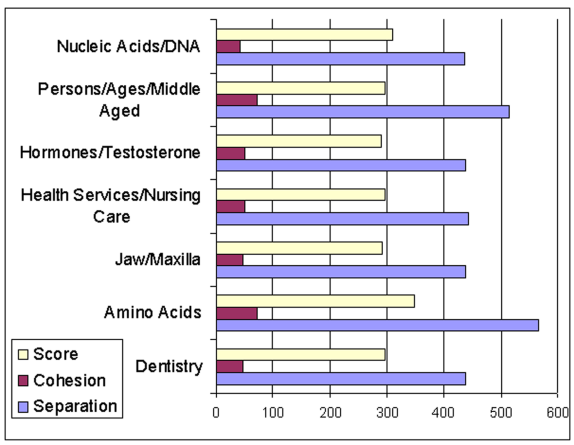


Figure 3: Example results from DISCOVER.

DIFFERENTIATE. Figure 2 shows the results of an application of DIFFERENTIATE to the web pages database. We noticed that the database showed a significant number of documents in the period from January 20, 2003 through March 22, 2003, so we set X' to contain these documents, and used the remaining documents as the background set. We then apply DIFFERENTIATE with D' as the entity hierarchy. Figure 2 figure shows the best PDC of size 7. Each line of the figure represents a node of the entity tree. Notice that, as required by a PDC, these nodes are all disjoint. The histogram contains two bars for each tree node. The top bar is the total number of documents in the foreground set which are members of that node; the bottom bar is the number of *additional* documents at this node in the foreground set, compared to the expectations raised by the background set alone. We observe, for example, that there are many references to George W. Bush in both data sets, but the "surplus" in the time range we consider is about 15% of the references. On the other hand, almost half the references to Colin Powell are surplus, suggesting that we could explore further to determine why this entity occurs so much more frequently in the selected time range.

DISCOVER. Figure 3 shows the results of a DISCOVER operation in the Medline database. The partition dimension is the MeSH category, and the measurement dimension is the publication date. The system returns seven topics that are temporally well-clustered. Amino Acids and Middle Aged Persons show maximal cohesion, while Amino Acids show maximal separation. While the MeSH taxonomy contains over 22K nodes, the DISCOVER operation has cast light on a small number of areas that met the requirement of containing a large number of documents within a particular interval of time. Such an operation would be useful, for example, to track trends in medical research.

6.3 Putting it together: An example workflow

We now give a real example from our prototype system showing how the operations can be combined into a workflow to allow the user to dynamically explore data, generate summaries, and find and explain anomalies. We use the Web pages data base for these experiments.

We begin by asking whether any particular entities have caught the public eye during focused periods of time. We apply the DISCOVER operator, partitioning by entity, and measuring by time. The results include a strong reference to Mel Gibson. We graph references to Mel Gibson over time, and determine that a strong spike appears around February, 2004. The movie "The Passion of the Christ" was released in mid-February, so this interval corresponds strongly to its release.

We decide to explore more broadly how movie stars show up over time. We restrict to a subset X' consisting of documents that reference movie stars, and then perform a two-dimensional DIVIDE operation using entities and time. This operation results in a two-dimensional table with entities on one dimension and time ranges on the other. We observe that the system has expanded movie stars to a node consisting of all actresses; a few particular actors, including Mel Gibson (as expected), Jim Caviezel (the lead in "The Passion"), and Arnold Schwarzenegger; and then an `other` node capturing remaining actors. But we would like to understand why Arnold Schwarzenegger appears. We observe from the results that there is relatively consistent coverage of Arnold across all the date ranges returned by the algorithm. We restrict to the set X' consisting of mentions of the Arnold entity, and perform a DIFFERENTIATE operation over the time dimension, using the entire document set as the background, to determine whether there are any particular time ranges during which Arnold occurs significantly more frequently than other subjects. The results show that Arnold appeared with surprising frequency in documents dated from February 6 to March 4 of 2004. Upon exploring the set of documents about Arnold during this time period, we see that the press attention was due to the buzz leading up to and including the California primary elections on March 2, 2004.

7. CONCLUSIONS

Our main contribution is MSDB, a framework for expressing and computing the highlights of a dynamic data set. This includes a general data model that is rich enough to represent the kinds of structures found in real-world examples. We propose PDCs as the basic notion for capturing the highlights of any data set. We introduce three basic analytic operations, namely, DIVIDE, DIFFERENTIATE, and DISCOVER to compute PDCs with various properties. We develop very efficient exact or approximation algorithms for these basic operations when the underlying dimensions are

numerical and hierarchical, the most commonly encountered types in practice. We believe that our general framework is applicable to many different data analytic techniques. There remain several important algorithmic issues.

Acknowledgments

We gratefully acknowledge many insightful discussions with Phokion Kolaitis.

8. REFERENCES

[1] R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. In *Proc. 13th Intl. Conference on Data Engineering*, pages 232–243, 1997.

[2] D. Barbará, Y. Li, and J. Couto. COOLCAT: An entropy-based algorithm for categorical clustering. In *Proc. 11th Intl. Conference on Information and Knowledge Management*, pages 582–589, 2002.

[3] L. Cabibbo and R. Torlone. A logical framework for querying multidimensional data. In *Intl. Seminar on New Techniques and Technologies for Statistics*, pages 155–162, 1998.

[4] E. F. Codd, S. B. Codd, and C. T. Salley. *Providing OLAP (on-line analytical processing) to user analysts: An IT mandate*, 1993. Arbor Software, now Hyperion Solutions Corp., White Paper.

[5] W. F. Cody, J. T. Kreulen, V. Krishna, and W. S. Spangler. The integration of business intelligence and knowledge management. *IBM Systems Journal*, 41(4):697–713, 2002.

[6] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[7] R. Feldman and I. Dagan. Knowledge discovery in textual databases (KDT). In *Knowledge Discovery and Data Mining*, pages 112–117, 1995.

[8] V. Ganti, J. Gehrke, and R. Ramakrishnan. Cactus: clustering categorical data using summaries. In *Proc. 5th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, pages 73–83, 1999.

[9] S. Gollapudi and D. Sivakumar. Framework and algorithms for trend analysis in massive temporal data sets. In *Proc. 13th Intl. Conference on Information and Knowledge Management*, pages 168–177, 2004.

[10] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total. In *Proc. 12th Intl. Conference on Data Engineering*, pages 152–159, 1996.

[11] M. Grigni and F. Manne. On the complexity of the generalized block distribution. In *Proc. 3rd Intl. Workshop on Parallel Algorithms for Irregularly Structured Problems*, pages 319–326, 1996.

[12] D. Gruhl, L. Chavet, D. Gibson, J. Meyer, P. Pattanayak, A. Tomkins, and J. Zien. How to build a WebFountain: An architecture for very large-scale text analytics. *IBM Systems Journal*, 43(1):64–77, 2004.

[13] S. Guha, R. Rastogi, and K. Shim. Rock: A robust clustering algorithm for categorical attributes. In *Proc. 15th Intl. Conference on Data Engineering*, page 512, 1999.

[14] M. Gyssens and L. Lakshmanan. A foundation for multi-dimensional databases. In *Proc. 23rd Intl. Conference on Very Large Data Bases*, pages 106–115, 1997.

[15] J. Han. Towards on-line analytical mining in large databases. *SIGMOD Record*, 27(1):97–107, 1998.

[16] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *Proc. ACM SIGMOD Intl. Conference on Management of Data*, pages 205–216, 1996.

[17] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, pages 105–142, 1999.

[18] S. Khanna, S. Muthukrishnan, and S. Skiena. Efficient array partitioning. In *Proc. 24th Intl. Colloquium on Automata, Languages and Programming*, pages 616–626, 1997.

[19] R. Kimball. *The Data Warehouse Toolkit*. J. Wiley and Sons, Inc, 1996.

[20] L. Lakshmanan, J. Pei, and J. Han. Quotient cube: How to summarize the semantics of a data cube. In *Proc. 28th Intl. Conference on Very Large Data Bases*, pages 778–789, 2002.

[21] B. Lent, R. Agrawal, and R. Srikant. Discovering trends in text databases. In *Proc. 3rd Intl. Conference on Knowledge Discovery in Databases and Data Mining*, August 1997.

[22] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.

[23] K. E. Paluch. A 2(1/8)-approximation algorithm for rectangle tiling. In *Proc. 31st Intl. Colloquium on Automata, Languages and Programming*, pages 1054–1065, 2004.

[24] S. Sarawagi. User-adaptive exploration of multidimensional data. In *Proc. 26th Intl. Conference on Very Large Data Bases*, pages 307–316, 2000.

[25] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. In *Proc. 6th Intl. Conference on Extending Database Technology*, pages 168–182, 1998.

[26] S. Sarawagi and G. Sathe. i^3: Intelligent, interactive investigation of OLAP data cubes. In *Proc. ACM SIGMOD Intl. Conference on Management of Data*, page 589, 2000.

[27] J. Tremblay and R. Manohar. *Discrete Mathematical Structures with Applications to Computer Science*. McGraw Hill Book Company, 1975.

[28] P. Vassiliadis and T. Sellis. A survey of logical models for OLAP databases. *SIGMOD Record*, 28(4):64–69, 1999.