

Using Data Cubes for Metarule-Guided Mining of Multi-Dimensional Association Rules

Micheline Kamber Jiawei Han Jenny Y. Chiang

Database Systems Research Laboratory, School of Computing Science
Simon Fraser University, Burnaby, BC, Canada V5A 1S6
E-mail: {kamber, han, ychiang}@cs.sfu.ca

Abstract

Metarule-guided mining is an interactive approach to data mining, where users probe the data under analysis by specifying hypotheses in the form of *metarules*, or pattern templates. Previous methods for metarule-guided mining of association rules have primarily used a transaction/relation table-based structure. Such approaches require costly, multiple scans of the data in order to find all the large itemsets. In this paper, we employ a novel approach to metarule-guided, multi-dimensional association rule mining which explores a *data cube* structure. We propose algorithms for metarule-guided mining using different cube structures: given a metarule containing p predicates, we compare mining on an n -dimensional cube structure (where $p < n$) with mining on smaller multiple p -dimensional cubes. In addition, we propose an efficient method for precomputing the cube, which takes into account the constraints imposed by the given metarule. Our preliminary performance study shows that a cube-based metarule-guided algorithm for mining multi-dimensional association rules is efficient and can easily be extended to the mining of multi-level, multi-dimensional association rules.

Introduction

Metarule-guided mining is a “human-centered” interactive approach to data mining, whereby the user can probe the data under analysis by specifying hypotheses in the form of *metarules*, or pattern templates. These hypotheses may be based on the user’s experience, expectations, or intuition regarding the data, or automatically generated based on the database schema (Shen et al. 1996). A data mining system attempts to confirm the hypotheses by searching for patterns that match the given metarules. Metarule-guided mining increases the likelihood of finding rules that are of interest to the user and can make the discovery process

more *efficient* by using the metarules to *constrain* the rule search space. For example, metarule (1) can be used to focus the data mining search towards rules disclosing which two factors combined promote the sales of *Pentium* computers.

$$\forall x \in person, P(x, y) \wedge Q(x, w) \Rightarrow buys(x, "pentium"), \quad (1)$$

where P and Q are *predicate variables*, x is a variable representing a person, and y and w are *object variables*. All variables will be instantiated to concrete values in the mining process, e.g., association rule (2) matches or *complies* with metarule (1).

$$\forall x \in person, owns(x, "laptop") \wedge income(x, "high") \Rightarrow buys(x, "pentium") \quad (2)$$

Metarule-guided mining is closely linked with other rule mining methods, especially association rule mining (Agrawal & Srikant 1994; Han & Fu 1995; Park et al. 1995; Srikant & Agrawal 1996; Meo et al. 1996; Toivonen 1996). There are some previous studies on data mining with metarules, such as (Klemettinen et al. 1994) which uses metarules for defining the form of interesting association rules and for filtering the generated rules, (Shen et al. 1996) which uses a metapattern to guide the mining of Bayesian data clustering rules, and (Fu & Han 1995) which uses a relation table-based structure for metarule-guided mining.

With recent progress on data warehousing and OLAP technology (Chaudhuri & Dayal 1997), it is expected that many mining tasks will be performed on data warehouses. Moreover, with efficient techniques developed for computing data cubes (Harinarayan et al. 1996; Agarwal et al. 1996; Zhao et al. 1997), it is important to explore data cube-based rule mining algorithms.

This motivates our study on metarule-guided mining using the data cube structure. Metarule-guided mining makes the search focused on desired rule patterns, whereas data cube structures make good use of structured warehouse and precomputed aggregation information. An integration of both is likely to lead to an efficient mining method. We consider two cases: (1) when a precomputed data cube is available; and (2) dy-

This research is partially supported by NSERC of Canada, by NCE/IRIS, and by research grants from BC Science Council, and MPR Teltech Ltd.

Technical Report CMPT-TR-97-10, School of Computing Science, Simon Fraser University, May 1997.

dynamic construction of relevant data cubes for metarule-guided mining. Moreover, to mine a metarule with p distinct predicates from n relevant attributes ($p < n$), one may either construct an n -dimensional data cube or construct a number of smaller p -dimensional data cubes. These cases lead to different variations of the algorithms, and their performance is studied and compared in this study.

Preliminaries

A metarule is a rule template of the form

$$P_1 \wedge P_2 \wedge \dots \wedge P_m \Rightarrow Q_1 \wedge Q_2 \wedge \dots \wedge Q_l \quad (3)$$

where P_i ($i = 1, \dots, m$) and Q_j ($j = 1, \dots, l$) are either instantiated predicates or predicate variables, and $p = m + l$ is the number of predicates in the metarule. A rule, R , complies with a metarule, M_R , iff it can be unified with M_R , e.g., (2) complies with metarule (1).

A rule of the form $X \Rightarrow Y$, where *body* X and *head* Y each consists of a set of conjunctive predicates, is a multi-dimensional association rule iff $\{X, Y\}$ contains more than one distinct predicate. For a data set D containing tuples from a relational database, the support of a rule $X \Rightarrow Y$ in D is the probability that the tuples in D contain both X and Y . The confidence of $X \Rightarrow Y$ is the probability that a tuple contains Y under the condition that it contains X (i.e., $Prob\{Y|X\}$). Typically, the rules that do not satisfy user- or expert-provided *minimum support* and *minimum confidence* thresholds are considered uninteresting. A k -predicate set (i.e., a set containing k conjunctive predicates) is large (or frequent) if its support is no less than the minimum support threshold. A rule $X \Rightarrow Y$ is strong if it satisfies both minimum support and minimum confidence thresholds.

A rule in which all the predicates have distinct predicate names is called a non-repetitive predicate multi-dimensional association rule. Our study of metarule-guided mining of strong, multi-dimensional association rules is confined to this case.

Example 1 Suppose a user wishes to mine a set of strong rules in a university database, associating the students' *gpa* with their *major*, *student_id*, *birth_place*, and *residence*. The task can be expressed in a data mining query language, DMQL (Han et al. 1995), as follows.

```
mine multi-dimensional association rules
from student
in relevance to major, student_id, birth_place, residence
set template  $P(s : student, x) \wedge Q(s, y) \Rightarrow gpa(s, z)$ .
with min_support = 0.20, min_confidence = 0.80
```

A data mining system first extracts the task-relevant data, i.e., the set of student records, projected onto $\{gpa, major, student_id, birth_place, residence\}$ using an SQL-query transformed from the provided query. The minimum support threshold is used to perform dimension reduction, i.e., removing attributes with too many distinct values, such as *student_id*, and perform necessary generalization/discretization, e.g., replacing raw data values, such

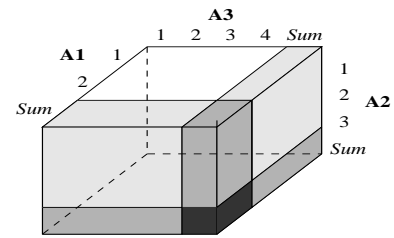


Figure 1: A 3-D data cube with summary layers

as 3.873 for *gpa*, by higher level concepts, such as “*excellent*”. This leaves only three attributes *major*, *birth_place*, *residence* for possible matching of the predicates, P and Q . \square

Our study explores rule mining using a data cube structure. An n -dimensional data cube, $C[A_1, \dots, A_n]$, is an n -D database where A_1, \dots, A_n are n dimensions. Each dimension of the cube, A_i , represents an attribute and contains $|A_i| + 1$ rows where $|A_i|$ is the number of distinct values in the dimension A_i . The first $|A_i|$ rows are *data rows*. Each distinct value of A_i takes one data row. The last row, the *sum* row, is used to store the summation of the counts of the corresponding columns of the above rows. A data cell in the cube, $C[a_{1,i_1}, \dots, a_{n,i_n}]$, stores the *count* of the corresponding (generalized) tuple of the initial relation, $r(A_1 = a_{1,i_1}, \dots, A_n = a_{n,i_n}, count)$. A sum cell in the cube, such as $C[sum, a_{2,i_2}, \dots, a_{n,i_n}]$, where *sum* is often represented by * or a keyword *all*, stores sum_1 , the *sum of the counts* of the generalized tuples which share the same values for all the second to the n -th columns, i.e., $r(*, A_2 = a_{2,i_2}, \dots, A_n = a_{n,i_n}, sum_1)$. Conceptually, a data cube can be viewed as a lattice of *cuboids*. The n -D space (i.e., *base cuboid*) consists of all data cells (i.e., no *’s in any dimension). The $(n-1)$ -D space consists of all the cells with a single * in any dimension, such as $r(*, a_{2,i_2}, \dots, a_{n,i_n}, sum_1)$, and so on. Finally, the 0 -D space consists of one cell with n *’d dimensions, i.e., $r(*, *, \dots, *, sum_n)$. A 3-D data cube is shown in Fig. 1. Notice that to handle sparse cubes efficiently, sparse matrix technology (Kimball 1996; Zhao et al. 1997) should be applied in data cube construction.

Example 2 Fig. 2 shows a 4-D data cube with the dimensions *gpa*, *major*, *birth_place* (*b_p*), and *residence* (*res*). The respective domains of each dimension are $\{\text{“excellent” (E), “fair” (F)}\}$, $\{\text{“Math” (Ma), “CS”, “Eng.”}\}$, $\{\text{“Canada” (Ca), “US”, “Europe” (Eu), “Asia” (As)}\}$, and $\{\text{“Vancouver” (Van), “suburbs” (sub)}\}$. \square

Methods for Metarule-Guided Mining Using Data Cubes

Our discussion of cube-based algorithms for metarule-guided mining of multi-dimensional association rules is divided into two cases: 1) mining on a precomputed data cube, and 2) mining integrated with the dynamic construction of data cube(s).

<i>gpa</i>	<i>maj</i>	<i>b_p</i>	<i>res</i>	<i>count</i>
E	Ma	As	Van	120
E	CS	Ca	sub	50
E	CS	US	Van	200
E	CS	US	sub	210
E	CS	Eu	sub	160
E	Eng	As	Van	300
F	Eng	As	Van	150
..

<i>gpa</i>	<i>maj</i>	<i>b_p</i>	<i>res</i>	<i>count</i>
E	Ma	*	*	290
E	CS	*	*	670
E	Eng	*	*	400
F	Ma	*	*	440
F	CS	*	*	50
F	Eng	*	*	150
E	*	Ca	*	70
E	*	US	*	490
E	*	Eu	*	210
E	*	As	*	590
..

<i>gpa</i>	<i>maj</i>	<i>b_p</i>	<i>res</i>	<i>count</i>
E	*	*	*	1360
F	*	*	*	640
*	Ma	*	*	730
*	CS	*	*	720
*	Eng	*	*	550
*	*	Ca	*	70
..

<i>gpa</i>	<i>maj</i>	<i>b_p</i>	<i>res</i>	<i>count</i>
*	*	*	*	2000

Figure 2: A 4-D data cube with the precomputed aggregation layers, mapped into relations.

The following notations are used in our discussion: M_R is a metarule, in conjunctive normal form as in Eq. (3); p is the number of instantiated predicates or predicate variables occurring in M_R ($p \leq n$, where n is the dimension of the cube); L_k is the set of large k -predicate sets, and each member of L_k has two fields, *predicate set* and *support count*; and R is the set of all strong association rules that comply with M_R .

Mining on an existing data cube

Owing to the wide availability of data warehouses and rapid progress of data cube technology, it is expected that a precomputed n -dimensional data cube (including the summary layers) is sometimes available for metarule-guided mining.

We examine two cube-based algorithms for this case: (1) **multi-D-slicing**, which finds large 1-predicate sets in p (where p is the number of predicates in a given metarule) and uses them to perform multi-dimensional slicing on the data cube. This algorithm adopts the spirit of table-based approaches to association rule mining (such as Apriori of Agrawal & Srikant 1994) as well as our earlier table-based study on metarule-guided mining (Fu & Han 1995), although here a data cube structure is used; and (2) **n-D cube search**, which directly examines the p -D cells of the precomputed n -D cube in order to find large p -predicate sets, L_p .

Algorithm 1 (multi-D-slicing) A multi-dimensional slicing technique for metarule-guided mining of multi-dimensional association rules.

Input: (1) Metarule, M_R , containing p distinct predicates; (2) An n -D data cube, $C[A_1, \dots, A_n]$, with all of its dimension space precomputed ($p \leq n$), where the cube contains the dimensions relevant to answering the specified metarule-guided query, and (3) min_sup and min_conf thresholds.

Output: R , the set of strong association rules that comply with M_R .

Method: First find the large 1-predicate sets in each dimension and then use the large predicate sets in dimension $(k - 1)$ to grow large k -predicate sets by multi-dimensional slicing on the data cube until large p -predicate sets are found.

- 1) **for each** dimension A_i ,
 $L_{1,(A_i)} = find_large_1_predicate_sets(C, A_i, min_sup)$;
- 2) $L_1 = \cup_i L_{1,(A_i)}$;
- 3) **for** ($k = 2$; ($k \leq p$) & $L_{k-1} \neq \emptyset$; $k++$) {
- 4) // intersect the $(k-1)$ dimensions to find L_k , i.e.,
 $L_k = find_large_predicate_sets(C, L_{k-1}, min_sup)$;
- 5) $R = rule_gen(L_p, M_R, C, min_conf)$;
- 6) **return** $\{R\}$;

Notice that at the k -th iteration, only the relevant summary layer needs to be loaded into main memory. If the layer itself is larger than the available main memory, the layer can be partitioned into *chunks* (Zhao et al. 1997) and only the corresponding chunks needs to be loaded for efficient computation.

Rationale. Step 1 finds the large 1-predicate sets for each dimension A_i by searching the 1-D space of the cube, which stores the occurrence frequency of each dimension value. A *candidate* predicate set is a potentially large predicate set. If L_{k-1} is not empty (step 3), step 4 uses L_{k-1} to derive the candidate k -predicate sets by intersecting the dimension vectors in L_{k-1} , based on the principle of Apriori i.e., *every subset of a large itemset must be large* (Agrawal & Srikant 1994). The counts of the candidates in k -space are examined in order to find L_k . Search continues until L_p is found (since L_p predicate sets are needed to generate rules complying with a metarule having p predicates). In step 5, a typical rule generation method is used to generate rules from L_p that comply with M_R . Only the rules that pass the min_conf threshold are returned. \square

Example 3 Suppose the mining task is the one specified in Ex. 1, and the data cube is given as in Fig. 2. Let $min_sup = 0.2$, and $min_conf = 0.8$. Following Algorithm 1, the mining is performed as follows.

1. Search 1-D cube space to find large 1-predicate sets, i.e.,
 $L_{1,(gpa)} = \{(E, *, *, *, 1360), (F, *, *, *, 640)\}$, $L_{1,(maj)} = \{(*, Ma, *, *, 730), (*, CS, *, *, 720), (*, Eng, *, *, 550)\}$, and so on.
2. Find large 2-predicate sets by searching 2-D space based on the candidate sets obtained by intersecting the large 1-D planes, e.g., intersecting $L_{1,(gpa)}$ and $L_{1,(maj)}$ leads to six candidate pairs: $C_{2,(gpa,maj)} = \{(E, Ma, *, *, *), \dots, (F, Eng, *, *, *)\}$. Checking 2-D space of the cube, $L_{2,(gpa,maj)} = \{(E, CS, *, *, 670), (E, Eng, *, *, 400), (F, Ma, *, *, 440)\}$. Similarly, $L_{2,(gpa,b_p)} = \{(E, *, US, *, 490), (E, *, As, *, 590)\}$, and so on.

3. Similarly, large 3-predicate sets are derived, resulting in $L_{3,(gpa,maj,b-p)} = \{(E, CS, US, *, 410)\}$, $L_{3,(gpa,maj,res)} = \{(E, CS, *, sub, 420)\}$, $L_{3,(gpa,b-p,res)} = \{(E, *, As, Van, 420)\}$, $L_{3,(maj,b-p,res)} = \{(*, Eng, As, Van, 450)\}$. These form L_3 . Since the number of predicates in the metarule is 3, the search for large predicate sets terminates.
4. L_3 is used to generate the strong rules that comply with the metarule, where $[s, c]$ represents the rule support (s) and confidence (c). These rules are: $major="CS" \wedge birth_place="US" \Rightarrow gpa="Excellent"$ [0.20,0.93], and $major="CS" \wedge residence="suburbs" \Rightarrow gpa="Excellent"$ [0.21, 0.95]. \square

Instead of searching for L_p from L_1, L_2 , etc., the count at each p -D cell can be examined directly to find the large p -predicate sets, from which rules matching the given metarule can be generated. This leads to,

Algorithm 2 (n-D cube search) A direct metarule-guided mining of multi-dimensional association rules from an n -D cube.

Input/Output: The same as Algorithm 1.

Method: 1. Examine the cell count of each p -D cell. If a cell count satisfies min_sup , then add the corresponding p -predicate set to L_p .

2. Call the same *rule_gen* procedure as in Algorithm 1, returning strong rules that comply with M_R from L_p .

Rationale. In step 1, we can find L_p directly by examining the p -D cells since the summary layers of the cube are computed. Since the entire data cube may be larger than the available memory, only the p -D summary layer need be loaded in. In step 2, given the large p -predicate sets, a typical rule generation method can be used to generate rules that comply with the metarule. \square

Example 4 Using the same metarule-guided query and cube as is Ex. 3, Algorithm 2 examines each 3-D cell, since p is 3. All 3-D cells that satisfy min_sup are added to L_3 , resulting in the same L_3 and rules as in Ex. 3. \square

Integration of cube construction with metarule-guided mining

If there is no precomputed data cube available for a given metarule-guided mining task, the mining process must be integrated with the precomputation of the relevant cube. Recall that a metarule, M_R , contains p predicates, of which m are in the body. In order to mine strong rules complying with M_R , only the m - and p -D layers of the cube need be computed. The p -D layer can be scanned to find L_p . The m -D layer is required in order to compute the confidence of rules satisfying M_R . Often, only a portion of an n -D cube needs to be constructed (where n is the number of relevant dimensions, and $n \geq p$); this is referred to as *abridged construction*.

In this study, we examine two approaches to metarule-guided mining integrated with cube construction: 1) *abridged n-D cube construction*, which computes the p - and m -D layers of an n -D cube, while searching p -D for L_p ; and 2) *abridged multiple p-D cube*

construction, which constructs a set of small multiple p -D cubes instead of one big n -D cube for mining. As in 1), only a subset of each cube is computed.

When a data cube cannot fit in main memory, the proposed algorithms employ a *cube chunking* strategy, where the cube is broken up into smaller, memory-fit chunks. This method is similar to a recent study (Zhao et al. 1997) where a single- or multi-pass multiway aggregation is explored for efficient cube construction, depending on the available memory. Our proposed hierarchy-based chunking technique scans through the data cube in a single pass since it requires smaller memory space. Moreover, the technique facilitates multi-level rule mining as well.

We describe our algorithms in two steps. First, the motivating ideas are presented behind the two abridged cube construction algorithms, by assuming that the relevant cubes fit into main memory. When the memory is insufficient, a chunking algorithm is applied for cube computation. For lack of space, only the memory-based algorithms are outlined, and the chunking technique is briefly discussed afterwards.

Algorithm 3 (abridged n-D cube construction)

Metarule-guided mining using metarule information to construct and search only a subset of an n -D cube.

Input: The same as Algorithm 1 except that the cube contains no computed summary layers.

Output: The same as Algorithm 1.

Method: Construct the p -D layer from the n -D data cells. Construct the m -D layer using only the p -D layer cells satisfying L_p . If the cube cannot fit into memory, use the chunking procedure described below. Call a rule generation algorithm to return strong rules.

Note. Each non-empty n -D data cell is visited only once, with its count accumulated in each of the corresponding p -D planes. The m -D layer is computed from a scan of the p -D cells having support greater than the minimum support threshold. A flag can be set to indicate that an m -D plane has been constructed, so that none of the counts on a p -D plane is inappropriately added more than once to the same m -D plane. \square

Recall that n is the number of dimensions relevant to the mining task. Instead of constructing an n -D cube for mining a p -predicate metarule, the mining can be performed on smaller p -D cubes ($p < n$). Since there are $\binom{n}{p}$ ways to choose p dimensions, mining can be performed by constructing $\binom{n}{p}$ p -D cubes and their corresponding m -D summary layers.

Algorithm 4 (abridged multiple p-D cube construction)

Metarule-guided mining of a p -predicate metarule with n related dimensions by construction of $\binom{n}{p}$ p -D cubes.

Input: The same as Algorithm 3.

Output: The same as Algorithm 1.

Method: Compute $\binom{n}{p}$ p -D cubes, then compute the m -D summary layer only for those "large" p -D data cells. A chunking algorithm is used, as described below, to compute the relevant layers if there is insufficient memory.

Since the (p -D) data cells already have their counts associated with them, only the m -D layer of each cube is computed. \square

Computing summary layers by chunking

When there is no sufficient main memory to hold a data cube for computation, a *chunking* technique which partitions the data cube into small subcubes (“chunks”) has been popularly used for cube construction (Sarawagi & Stonebraker 1994; Agarwal et al. 1996; Zhao et al. 1997).

An efficient, array-based algorithm for simultaneous multidimensional aggregation has been studied recently (Zhao et al. 1997). At any given time, at least one chunk is loaded into memory. The data cells of the chunk are scanned, and the corresponding summary layers of the data cube are updated. Hence, memory is required not only for the chunk being processed, but also for the portions of the summary layers that are being updated. If the memory is sufficient to hold the chunk and the summary layer portions, the summary layers can be computed in a single sweep through each cube chunk. Otherwise, some chunks have to be scanned more than once.

A similar chunking-based method for cube computation has been proposed in our study. The method is based on partitioning a cube (“chunking”) according to dimension hierarchies, computing each chunk as an independent subcube, and merging summary layers of subcubes into a cube summary layer. Since each chunk is partitioned according to the corresponding dimension hierarchies, a chunk forms a semantically meaningful entity and its computation is equivalent to computing an interesting meaningful subcube which may facilitate mining of multiple-level rules (Han & Fu 1995). Also, each chunk and its immediate summary layers are small enough to fit in main memory. Moreover, one scan of a chunk derives all of its summary layers, and these layers are sufficient for computing the summary layers of the entire cube. Therefore, a single pass through each chunk is sufficient for computing the entire cube.

In comparison with the algorithm proposed by (Zhao et al. 1997), although each chunk will be accessed only once, the subsequent computation of cube summary layers may need to fetch the summary layers of multiple chunks, which may cost I/O operations. However, the ordering in which summary layers are computed can be explored, e.g., by computing some summary layers simultaneously if there is sufficient memory. Therefore, this hierarchy-based chunking and first computation of chunk summary layers is an interesting alternative to (Zhao et al. 1997) for cube computation. This is especially beneficial for meta-rule guided mining since only a few cube layers, such as p and m , need to be computed in rule mining.

Performance Study

Four proposed algorithms (multi-D-slicing, n-D cube search, abridged n-D cube construction (preliminary) and abridged multiple p-D cube construction (preliminary)) were implemented and tested on a SUN/SPARC-Station SS-20 with 32MB of main memory running Solaris 2.5. The testbed consists of a set of synthetic data cubes with either the summary layers precomputed or computed on-the-fly based on the nature of the algorithm.

To reduce the implementation effort, all the algorithms are running in main memory with disk-based accesses simulated and computed. The estimates of the CPU computation and I/O accessing costs in the simulation experiments are: each cube cell takes 10 bytes in size (including the index fields and count); each data page holds 100 cube cells; a maximum of 100 pages in main memory is allocated to hold the cube data; search and comparison of each data cube cell in main memory takes 50 units CPU time; search and fetch of a page from disk to main memory takes 5000 units I/O and CPU time. We assume that all the data are stored in the $B+$ -tree structures, and the root and level-1 non-leaf node pages of the $B+$ -trees are always allocated in the main memory.

Each simulation was tested using a metarule containing $p = 3$ predicates, of which $m = 2$ are in the metarule body. The minimum support and minimum confidence thresholds were 2% and 80%, respectively. Fig. 3a) shows the relative performance of the multi-D-slicing and n-D cube search algorithms, tested on pre-computed data cubes representing 100 to 10,000 generalized tuples, or cells. Each data cube contains five dimensions, with ten distinct values per dimension. For the data tested, n-D cube search was up to four times more efficient than multi-D-slicing. Regardless of the number of generalized tuples, n-D cube search must examine the same number of cells (i.e., the p -D layer) in order to find L_p . Hence, it exhibits better scalability than multi-D-slicing, whose execution time is dependent on the number of candidates found, which typically increases with the data set size.

The abridged n-D cube construction and abridged multiple p-D cube construction algorithms, where the input data cubes do not have precomputed summary layers, were compared. Fig 3b) illustrates the effect of increasing the number of values (v) per dimension, given 10,000 generalized tuples, where n , the number of relevant dimensions is 6, $p = 3$, and $m = 2$. Fig 3c) illustrates the effect of increasing n while p , m , and v are constant. Essentially, these figures compare the performance of metarule-guided mining (with dynamic cube construction) on one n -D cube versus on multiple smaller p -D cubes. When the size of the cube is relatively small, mining on one n -D cube is more efficient than mining on several smaller cubes. However, as the size of the n -D cube grows (due to increasing n or v), then it becomes more efficient to mine on smaller

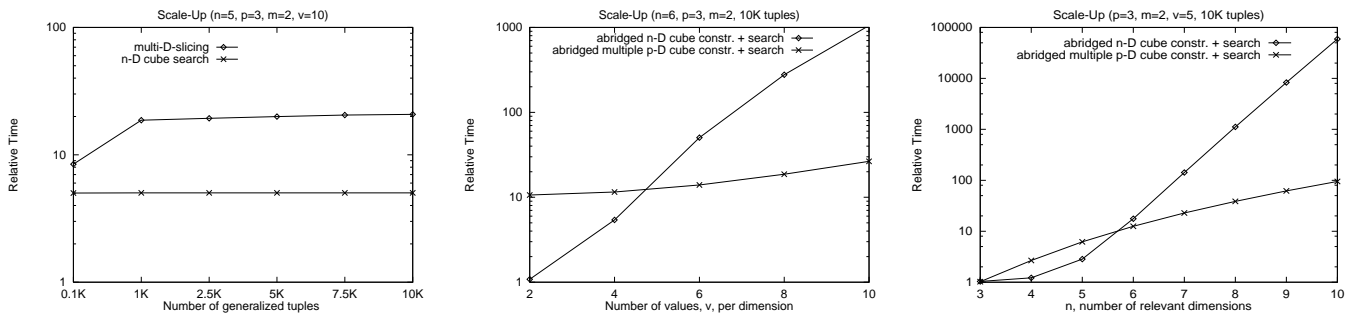


Figure 3: Performance with (a) the growth of generalized cells (tuples) in the cube, (b) the growth of v , the number of distinct values per dimension, and (c) the growth of n , the number of relevant dimensions.

multiple p -D cubes. Hence, with respect to the number of relevant dimensions and the number of values per dimension, abridged multiple p -D cube construction exhibits greater scalability than abridged n -D cube construction.

Based on the performance study analysis, we recommend the n -D cube search algorithm when the data cube is available. Otherwise (when the data cube must be computed on-the-fly), we recommend abridged multiple p -D cube construction.

Discussion and Conclusions

Previous methods for metarule-guided mining of association rules have primarily used a table-based structure. Such approaches require costly, multiple scans of the data. We have proposed to explore a data cube structure for metarule-guided mining of multidimensional association rules. Four algorithms were proposed based on the availability of data cubes. When an appropriate n -D cube is available, an n -D cube search of the p -layer summary cells can be applied (where p is the number of metarule predicates, $p \leq n$). If there is no suitable data cube available, the information from the metarule, such as the number of predicates, can be used for efficient construction of a relevant cube and mining of the required rules. Preliminary results show that when the number of relevant dimensions (or values per dimension) is large, it is more efficient to construct and mine from multiple, smaller p -D cubes rather than from one large n -D cube. For efficient cube construction, a hierarchy-based chunking algorithm is proposed which requires the scan of each chunk at most once. Moreover, the storage of summary layers of chunks on disk may facilitate mining of multiple-level rules.

This study is confined to mining rules with no repetitive predicates. Efficient methods for metarule-guided mining of more general forms of rules using data cubes is an interesting topic for future research.

References

Agarwal, S.; Agrawal, R.; Deshpande, P. M.; Gupta, A.; Naughton, J. F.; Ramakrishnan, R.; and Sarawagi, S.

1996. On the computation of multidimensional aggregates. In *VLDB'96*, 506–521.

Agrawal, R., and Srikant, R. 1994. Fast algorithms for mining association rules. In *VLDB'94*, 487–499.

Chaudhuri, S., and Dayal, U. 1997. An overview of data warehousing and OLAP technology. *SIGMOD Record* 26:65–74.

Fu, Y., and Han, J. 1995. Meta-rule-guided mining of association rules in relational databases. In *Proc. Int'l Workshop on Integration of Knowledge Discovery with Deductive and Object-Oriented Databases*, 39–46.

Han, J., and Fu, Y. 1995. Discovery of multiple-level association rules from large databases. In *VLDB'95*, 420–431.

Harinarayan, V.; Rajaraman, A.; and Ullman, J. D. 1996. Implementing data cubes efficiently. In *Proc. ACM-SIGMOD Int. Conf. Management of Data*, 205–216.

Klemettinen, M.; Mannila, H.; Ronkainen, P.; Toivonen, H.; and Verkamo, A. 1994. Finding interesting rules from large sets of discovered association rules. In *Proc. Int. Conf. Information & Knowledge Management*, 401–408.

Meo, R.; Psaila, G.; and Ceri, S. 1996. A new SQL-like operator for mining association rules. In *Proc. Int. Conf. Very Large Data Bases*, 122–133.

Park, J.; Chen, M.; and Yu, P. 1995. An effective hash-based algorithm for mining association rules. In *Proc. ACM-SIGMOD Int. Conf. Management of Data*, 175–186.

Shen, W.; Ong, K.; Mitbander, B.; and Zaniolo, C. 1996. Metaqueries for data mining. In Fayyad, U.; et al. (eds), *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press. 375–398.

Sarawagi, S.; and Stonebraker, M. 1994. Efficient organization of large multidimensional arrays. *Proc. 1994 Int. Conf. Data Engineering*. 328–336.

Srikant, R.; and Agrawal, R. 1996. Mining quantitative association rules in large relational tables. In *Proc. ACM-SIGMOD Int. Conf. Management of Data*, 1–12.

Toivonen, H. 1996. Sampling large databases for association rules. In *Proc. 1996 Int. Conf. Very Large Data Bases*, 134–145.

Zhao, Y.; Deshpande, P. M.; and Naughton, J. F. 1997. An array-based algorithm for simultaneous multidimensional aggregates. In *Proc. ACM-SIGMOD Int. Conf. Management of Data*, 159–170.