Navigation Rules for Exploring Large Multidimensional Data Cubes

Navin Kumar, University of Maryland, Baltimore County, USA

Aryya Gangopadhyay, University of Maryland, Baltimore County, USA

George Karabatis, University of Maryland, Baltimore County, USA

Sanjay Bapna, Morgan State University, USA

Zhiyuan Chen, University of Maryland, Baltimore County, USA

ABSTRACT

Navigating through multidimensional data cubes is a nontrivial task. Although On-Line Analytical Processing (OLAP) provides the capability to view multidimensional data through rollup, drill-down, and slicing-dicing, it offers minimal guidance to end users in the actual knowledge discovery process. In this article, we address this knowledge discovery problem by identifying novel and useful patterns concealed in multidimensional data that are used for effective exploration of data cubes. We present an algorithm for the DIscovery of Sk-NAvigation Rules (DISNAR), which discovers the hidden interesting patterns in the form of Sk-navigation rules using a test of skewness on the pairs of the current and its candidate drill-down lattice nodes. The rules then are used to enhance navigational capabilities, as illustrated by our rule-driven system. Extensive experimental analysis shows that the DISNAR algorithm discovers the interesting patterns with a high recall and precision with small execution time and low space overhead.

Keywords: cube navigation; data cube lattice; OLAP; navigation rules; skewness

INTRODUCTION

With the ever-increasing volume of data collected and archived by organizations, it has become increasingly critical in order to navigate efficiently and effectively through large, multidimensional databases. Dimensional modeling techniques offer modeling paradigms in order to capture measures along multiple dimensions, and On-Line Analytical Processing (OLAP) tools provide various operations such as rollup, drill-down, and slicing-dicing in order to select target datasets and to view them from different angles. However, it is still a daunting task for end users to detect manually the hidden patterns in the voluminous and complex lattice of multidimensional databases. The manual data analysis during cube exploration becomes a bottleneck in the knowledge-discovery process.

In this article, we address this problem by proposing a knowledge-discovery technique in order to identify novel and useful patterns in multidimensional data that are later presented to end users in an apprehensible form. Specifically, we propose DIscovery of Sk-NAvigation Rules (DISNAR), a novel skewness-based algorithm, in order to detect hidden surprises in data cubes, and then we use these surprises to provide a method for cube navigation. In this context, a surprise reveals how anomalous a set of transactions is when compared with another set of closely related transactions in the fact table. The anomalous transactions could be defined either by few outliers in the datasets influencing the aggregated datasets or by a group of transactions showing substantial difference in facts, such as profit or cost, from the remaining transactions. Because the notion of a surprise is an intuitive one, different users may have different impressions on what constitutes a surprise. Our rule-driven system allows users to control the knowledge-discovery process by letting them set the baseline for surprises by simply adjusting the level of significance.

Our work addresses two open research issues. The first one revolves around the inadequate level of decision support provided by most OLAP systems and is limited to aggregated data, which may not be sufficient for all users. Users often need to form concepts related to surprises in the data. The proposed approach using skewness aids in forming these concepts of surprises. The second one deals with the difficulty of navigating through data cube lattices. Currently, a user must have a fairly good understanding of the multidimensional model and a good intuition of what might be discovered in order to navigate through the vast magnitude of combinatorially explosive datasets involving high dimensionality and high variability in a data cube lattice. Without such knowledge, exploring these huge datasets is constrained by minimal system guidance and often misled by aggregated views. For instance, in a hypothetical profit scenario, a user may view that for years 1991 through 1998 the total profit values are very close to each other and may conclude that there is insignificant difference among the years. However, the data may contain some very high as well as very low profit segments when drilled down to lower hierarchical levels defined by quarters, months, and weeks, even though the aggregated values reveal no annual profit differences. This scenario describes a critical issue in cube exploration in which aggregated views with no definite guidance often present incorrect roadmaps to users. We approach this issue by guiding users based on surprises in the dataset. Thus, our methodology provides proper guidance through the discovery of surprises and, at the same time, lets users drive the knowledge exploration process.

This article is fundamentally different from related work in the sense that it identifies surprises by examining the data at the lowest level of granularity at every cuboid in a cube lattice, as opposed to the common practice of using aggregated data. Aggregation often hides the characteristics of the detailed data; an extremely high value and an extremely low value can be aggregated to a moderate value, hiding both extreme values. Using the smallest level of granularity, the problem of hiding a surprise as a side effect of aggregation is avoided. Our key contributions are as follows:

1. We discover hidden surprises with a high recall and precision in multidimensional cubes by using the statistical property of skewness on lattice nodes.
2. We enable users to examine different sets of surprises according to their needs. In our rule-driven system, a user can navigate to the same dataset multiple times by adjusting the critical level of significance of skewness. The practical implication of this adjustment is that some obvious surprises already may be known to the user, and therefore, he or she might search for more fine-grained surprises in different iterations.
3. We detect the surprises for multiple measures simultaneously at lattice nodes. For example, in most real-life settings, multiple measures such as revenue and

cost are viewed simultaneously in order to gain useful insight.

4. We introduce a rule-driven system that presents the discovered surprises in a more comprehensible form, thereby guiding users to proper navigation paths. Our approach contrasts a traditional OLAP system in which presenting information is overwhelming due to the combinatorial explosion in the number of dimensions, levels in the dimensional hierarchies, and transactional cardinality.

The rest of the article is organized as follows. We present the related work next followed by the preliminaries on data cube model. Next, we describe the DISNAR algorithm. We then present a description of a prototype of the rule-driven Sk-Navigation system. Next, we proceed with the detailed experimental results followed by conclusions and future work.

RELATED WORK

A data warehouse is defined as a "subject-oriented, integrated, time-variant, and non-volatile collection of data in support of management's decision making process" (Inmon 1996). There are two essential tasks to be performed in a data warehouse: storage and maintenance of data, and providing end users effective means of navigation and viewing data using multidimensional models. Consequently, work in this field has been performed in the areas of data warehouse development and maintenance (Bouzeghoub, Fabret, & Matulovic, 1999; Chen, Dehne, Eavis, & Rau-Chaplin, 2006; Gupta, Mumick, Rao, & Ross, 2001; Yang & Widom, 2000), view materialization (Gupta & Mumick, 1999; Theodoratos & Sellis, 1999; Theodoratos & Sellis, 1999; Yang, Karlapalem, & Li, 1997), multidimensional modeling (Bauer et al., 2000; Golfarelli, Maio, & Rizzi, 2000; Hurtado & Mendelzon, 2001), query languages and evaluation (Lemire, 2002; Marcel, 1999; Mendelzon & Vaisman, 2000; Park, Kim, & Lee, 2001; Poon, 2003; Vaisman & Mendelzon, 2001), visualization (Choong, Laurent, & Marcel, 2001; Maniatis et al., 2005), indexing (Chan & Ioannidis, 1998; Gupta, Harinarayan, Rajaraman, & Ullman, 1997; Sarawagi, 1997), storage and chunking (Deshpande, Ramasamy, Shukla, & Naughton, 1998; Kaser & Lemire, 2003), and online analytical mining (OLAM) (Chen, 1999; Fu, 2005; Han, 1998; Han, Chee, & Chiang, 1998; Sarawagi, Agrawal, & Megiddo, 1998; Tjioe & Taniar, 2005).

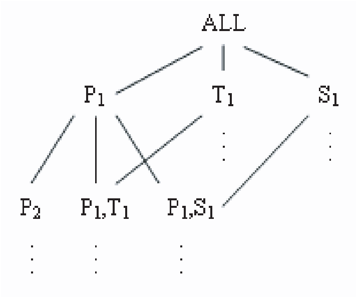
Cube lattices commonly are used in multidimensional data mining in large databases (Casali, Cicchetti, & Lakhal, 2003). Association rule mining, first introduced in Agrawal, Imielinski, and Swami (1993), is one of the key research topics and has been developed in different ways, such as the quantitative rules (Aumann & Lindell, 1999), RLSD (Zhang, Bloedorn, Rosen, & Venese, 2004), strong affinity patterns (Xiong, Tan, & Kumar, 2003), DGX distribution (Bi, Faloutsos, & Korn, 2001), and CCMine (Kim, Lee, & Han, 2004). These methods focus primarily on finding the strong associations among the items sets. An association rule, for example, establishes interesting relationships among items but cannot determine an interesting set of transactions containing anomalies, if its support or confidence is very low. On the other hand, the outlier detection techniques, such as LOADED (Ghoting, Otey, & Parthasarathy, 2004), are useful mainly in identifying the extreme anomalies in the datasets but do not provide adequate solutions to describe the impact of the surprise on the overall patterns in the datasets. The DISNAR algorithm addresses the discovery of hidden patterns in a multidimensional cube and also detects the nodes in the cube lattice that are influenced directly or indirectly by interesting patterns.

Another important issue in multidimensional databases is guiding the users to explore the data cube. While query-driven knowledge discovery (Boulicaut, Marcel, & Rigotti, 1999) and discovery-driven exploration of OLAP data cubes (Sarawagi et al., 1998) have been proposed, they do not address the easy identification of surprises. Sarawagi et al. (1998) define surprises in a rigid manner, implying that users cannot view them differently according to their needs. Furthermore, it is overwhelming for users to study the surprises by looking at the datasets presented in a large number of rows and columns. This work was extended further by Sarawagi (1999, 2000) by explaining differences in aggregate values and by using these to discover surprises in unexplored parts of a data cube using maximum entropy principles. However, the previously mentioned limitations still exist in these studies, and we differ from these by dealing with transactions at the lowest levels of hierarchies. Other related work in the area of subgroup patterns (Klosgen, 2002) addresses the general problem of defining and identifying local subgroups. While important, such methods also presume the way a surprise is defined and do not provide a generalized navigational methodology. Another recent approach in this area is presented by Fabris and Freitas (2006) in which the authors use Simpson's paradox as the basis for defining surprises. In this article, we provide a method for navigation that will discover not only Simpson's paradoxes but also additional surprises, as defined by the user. This article extends the existing work by detecting surprises concealed at the lowest level of granularities in large multidimensional databases and by establishing the concept of cube navigation using the discovered surprises in a flexible manner.

OVERVIEW OF THE DATA CUBE MODEL

Let us assume that the data cube consists of m dimensions, $d_1, d_2, \ldots, d_m$. A dimension d_i is associated with a concept hierarchy containing one or more levels of aggregation. Level l_{ij} represents the j^{th} level of dimension d_i, such that $1 \leq j \leq L_i$ where L_i is the number of levels associated with d_i. A level l_{ij} contains a set of attributes. Let v_{ijk} be the k^{th} attribute at level l_{ij}. The facts are numerical measures, usually the objects of analysis. Assume that there are s facts, $f_1, f_2, \ldots, f_s$ in the fact table and that w_{pq} is the q^{th} value for fact f_p, where $1 \leq p \leq s$, $w_{pq} \in W_p$ where W_p is the domain of possible values of fact f_p. In our example, $W_{[profit]}$ is a set of discrete values associated with *profit*, $W_{[profit]} =$ {low, medium, high}. The fact table contains the complete transaction set $T = [\tau_1, \tau_2, \ldots, \tau_n]$, where n is the total number of transactions. A transaction, τ is represented as $\{(x_1, x_2, \ldots, x_i, \ldots, x_m), (f_1, f_2, \ldots, f_p, \ldots, f_s)\}$, where x_i is an attribute (v_{ijk}) from the lowest level ($=L_i$) of d_i, and f_p is an associated fact.

Figure 1. A partial lattice of cuboids



Given m dimensions, *latt(m)* is a lattice of cuboids, each being a distinct combination of hierarchical levels of dimensions. Figure 1 illustrates the partial lattice for a grocery database with *product* (P), *time* (T) and *store* (S) dimensions. (P_1) represents a one-dimensional cuboid containing level-1 (*product category*) for *product* dimension. Similarly, (P_1,T_1) represents a two-dimensional cuboid constructed by level-1 of *product* (*category*) and *time* (*year*) dimensions. An edge shows the possible navigation from one cuboid to another; for instance, from (P_1) to (P_1,T_1). A cuboid consists of a set of nodes in which a node contains the attributes v_{ijk}'s and the facts f_p's. For example, the cuboid (P_1) contains three nodes; namely, Food, \$445K, Drinks, \$680K, and Supplies, \$380K, where the profit is the fact. Every node corresponds to a set of transactions, also referred to as a dataset. A navigation path describes a traversal through the nodes in the lattice. For example, the path (P_1) → (P_2) → (P_2,T_1) → (P_2,T_1,S_1) suggests that a user first look at a node at (P_1) *Product category*, drill down to a node at (P_2) *product subcategory*, and subsequently view the nodes at (P_2,T_1) *product subcategory and year* and (P_2,T_1,S_1) *product subcategory, year,* and *region*. We next describe the navigation paths.

A navigation path comprises a set of nodes in a predefined order starting from a root node, which is a node in a one-dimensional cuboid such as (P_1), (T_1), or (S_1). Subsequent nodes are determined either by drilling down one dimension from a preceding node or by including a new dimension that does not exist in the preceding node. This process may continue until there are no more nodes to traverse. As an example of navigation, starting from a current node containing the rule *Year = 1992*, the candidate nodes to examine for traversal include the rules {"*Quarter=Q1-1992*", "*Quarter=Q2-1992*", "*Quarter=Q3-1992*", "*Quarter=Q4-1992*", "*Year=1992 and Product Category = Drinks*", …, "*Year = 1992 and Region = Eastern*", …}.

Suppose nod_{curr} is a lattice node with the fact $f_{p[curr]}$:

$$nod_{curr} = \{l_{1[curr]}, l_{2[curr]}, ...l_{i[curr]}, ...l_{m[curr]}\}.$$

Here $l_{i[curr]}$ is the hierarchical level for dimension d_i at node nod_{curr}. A distance metric, $dist(nod_x, nod_y)$, between two nodes is defined as follows:

$$dist(nod_x, nod_y) = \sum_{i=1}^{m} l_i(nod_x) - l_i(nod_y).$$

A node nod_{cand} is considered a candidate node if its distance from the current node nod_{curr} is 1 under the following conditions:

$$\exists\, l_{i[cand]} : l_{i[cand]} = l_{i[curr]} + 1 \text{ for exactly one } l_{i[cand]};$$

$$\forall\, i\ l_i(nod_{cand}) \geq l_i(nod_{curr}), \text{ where } i = 1, ...m.$$

Assume that nod_{curr} contains m_1 dimensions ($1 \leq m_1 \leq m$) at $l_{ij} \neq 0$. Candidate nodes then are determined by (1) navigating one level down in dimensional hierarchy for d_k, $d_k \in m_1$, and (2) navigating along a new dimension d_l, $d_l \in (m\text{-}m_1)$ dimension set.

THE DISNAR ALGORITHM

We present DISNAR, an algorithm to discover hidden surprises in multidimensional cubes using the property of skewness. DISNAR utilizes normal distribution for detecting skewed nodes existing in cube lattices. To discover the surprises, we follow a four-step recursive process as follows:

1. Given a current node, generate a set of candidate nodes.
2. Measure the skewness of candidate nodes.
3. Apply the test of significance of skewness on candidate nodes.
4. Transform nodes with significant skewness into Sk-navigation rules.

Once a node with a significant skewness is identified, it acts as the current node for generating next-level candidates. The algorithm terminates either when it reaches the lowest level nodes in the lattice or when no more nodes of surprises are discovered in the current iteration. We then establish the concept of cube navigation using Sk-navigation rules. Finally, we present three ways to guide through the navigations in order to enhance the cube exploration capabilities.

Generation of candidate nodes (Step 1) is explained in the previous section. We now explain the next three steps of the algorithm.

Step 2. Measurement of Skewness for Candidate Nodes

A dataset is skewed if it is not symmetrically distributed with reference to its central axis (usually the mean). If μ and σ are the mean and standard deviation of a population, skewness is defined by its third standardized moment as follows:

$$skewness\ \sqrt{\beta_1} = \frac{E(X-\mu)^3}{[E(X-\mu)^2]^{3/2}} = \frac{E(X-\mu)^3}{\sigma^3}.$$

Here, E is the expected value operator. For a symmetric distribution, such as a normal distribution N(μ,σ), the skewness $\sqrt{\beta_1}$ is equal to zero. A nonnormal distribution is skewed either to the left or to the right of its central axis, $\sqrt{\beta_1} \neq 1$. If the distribution is positively (negatively) skewed, $\sqrt{\beta_1} > 0$ ($\sqrt{\beta_1} <$), the portion of the curve on the right (left) of the central axis contains the longer tail.

Once the candidate nod_{cand} is identified, we measure skewness of the dataset for nod_{cand} with respect to its central axis defined by the mean of the fact at nod_{curr}. Evidently, nod_{cand} always will contain a subset of the transactions contained at current node nod_{curr}. For example, a candidate node *product subcategory = frozen food* will contain a subset of the transactions from the current node *product category = food*. Let $T_{[\text{curr}]}$ be the transaction set for nod_{curr}, $T_{[\text{cand}]}$ be the transaction set for nod_{cand}, n_1 be the number of transactions at nod_{curr}, and n_2 be the number of transactions at nod_{cand}. Then:

$T_{[curr]} \subseteq T$, $T_{[cand]} \subseteq T$, and $T_{[cand]} \subseteq T_{[curr]}$, $n_1 \leq n$, $n_2 \leq n$, and $n_2 \leq n_1$.

Let us say that $\overline{f}_{p[curr]}$ is the mean value of fact $f_{p[curr]}$ at nod_{curr}. Let $f_{\text{p(j)[cand]}}$ be f_p's value for the j^{th} transaction $\tau_{j[cand]}$.

The skewness of candidate node nod_{cand} is measured as follows:

$$\sqrt{b_{1[cand]}} = \frac{\sum_{j=1}^{n_2} (f_{p(j)[cand]} - \overline{f}_{p[curr]})^3 / n_2}{\left[\sum_{j=1}^{n_2} (f_{p(j)[cand]} - \overline{f}_{p[curr]})^2 / n_2 \right]^{3/2}}$$

Step 3. Test of Significance of Skewness for Candidate Nodes

In order to determine the pattern (i.e., low-profit region, high-profit region) at a node relative to its parent node, we establish the corresponding side of the central axis (left or right) for the significance of skewness. It is achieved by conducting a one-sided test of skewness as follows:

H_0: normality with $\sqrt{\beta_1} = 0$
H_1: non-normality with $\sqrt{\beta_1} > 0$, or H_1: non-normality with $\sqrt{\beta_1} < 0$

If $\sqrt{\beta_1}$ satisfies the critical skewness at a level of significance α, the null hypothesis H_0 becomes false and is rejected. Based on positive or negative skewness, we infer the element of W_{pq} to which the fact f_p belongs. For example, a positive skewness for the profit means a high-profit region relative to its parent.

We apply the test of significance of skewness (D'Agostino & Stephens, 1986) to discover the candidate nodes with significant skewness. Based on the sample size (n_2) of nod_{cand}, the critical skewness $\sqrt{b_1} \, critical_{[cand]}$ is determined as follows.

Critical Value for Skewness [$5 \leq n_2 \leq 35$]

The skewness $\sqrt{b_1}_{[cand]}$ for a candidate node is compared with the simulation probability points of $\sqrt{b_1}$ called Monte Carlo points at a level of significance α.

Critical Value for Skewness [$n_2 \geq 36$]

We perform a normal approximation of the null distribution of $\sqrt{b_1}$ characterized by a Johnson S_U curve. The obtained Z-value represents approximately a standard normal variable. This Z-value is looked up in the standard normal distribution N(0,1) table at a given α to either reject or accept the null hypothesis H_0. Another way to test the skewness is to compare it against the probability points for $\sqrt{b_1}$ computed from Johnson S_U approximation. These probability points are available for sample sizes up to 10,000. We applied the least square fitting technique on available probability points to extend them for sample sizes greater than 10,000.

Test of Significance of Skewness

If $\sqrt{b_1}_{[cand]} > \sqrt{b_1} \, critical_{[cand]}$, we reject the null hypothesis; therefore, the skewness $\sqrt{b_1}_{[cand]}$ of candidate node nod_{cand} is found significant at α, suggesting a significantly skewed pattern at nod_{cand} on the navigation path from nod_{curr} to nod_{cand}.

If $\sqrt{b_1}_{[cand]} \leq \sqrt{b_1} \, critical_{[cand]}$, the null hypothesis cannot be rejected, implying that the candidate dataset normally is distributed with no significant skewness in $\overline{f}_{p[curr]}$. Figure 2 presents Step 1 to Step 3 of the DISNAR algorithm. The

Figure 2. The DISNAR algorithm

```
discover_sk_navigation_rules ()
Input: dimensional hierarchy, fact table, level of significance α
Output: complete set of sk-navigation rules (rs_skewbar)

let $L_i$ = number of levels for dimension $d_i$;
set curr_navig_level ← 0;
set ns_current ← empty; // current nodeset
set max_navig_level ← $\Sigma L_i - 1$;
while curr_ navig_level < max_navig_level++
  generate ns_current ← nodes_of_skewness(curr_navig_level);
  for every node $nod_{cand}$ in ns_current
    apply test of significance for $\sqrt{b_1}_{[cand]}$ at (n, α);
    if $nod_{cand}$ is significant then
      add $nod_{cand}$ to rs_skewbar;
      rule_navig_level ($nod_{cand}$) ← $\Sigma l_{ij}$;
  end for;
end while;
end;

nodes_of_skewness (curr_navig_level)
Input: curr_navig_level  // current level of navigation
Output: set of candidate nodes $nod_{cand}$ and their skewness $\sqrt{b_1}_{[cand]}$
set rule_navig_level ← null; // navigation level for a rule
if curr_navig_level = 0, then
  $nod_{curr} : \{l_{10}, l_{20}, ..., l_{i0}, ..., l_{m0}\}$;
else  set rs_current ← rs_skewbar
        where rule_navig_level ← (curr_navig_level-1) and current node is not null;
let t ← number of rules in rs_current;
for each of t rules
  $nod_{curr} : \{l_{1t}, l_{2t}, ..., l_{it}, ..., l_{mt}\}$;
  generate candidate nodes $nod_{cand}$ such that $dist(nod_{curr}, nod_{cand}) = 1$;
  measure $\sqrt{b_1}_{[cand]}$ for every $nod_{cand}$ and $nod_{curr}$;
end for;
end;
```

function *nodes_of_skewness()* describes the first two steps, generation of candidate nodes and measurement of skewness. The main function *discover_sk_navigation_rules()* describes Step 3, the test of significance of skewness.

Step 4. Transforming Nodes of Significant Skewness Into Sk-Navigation Rules

A candidate node nod_{cand}, identified for a significant skewed pattern, is transformed into an Sk-navigation rule. The individual dimensions, along with their hierarchical levels, and attributes from nod_{cand} are represented in the antecedent. If a dimension does not exist in nod_{cand}, it is assumed to be aggregated to *ALL*. The consequent consists of a fact and its skewness. For example, a positive skewness for profit suggests a high profit region at nod_{cand} relative to its parent node. Similarly, a negative skewness suggests a low profit region at nod_{cand} relative to its parent node. An Sk-navigation rule *skr* then is represented as follows:

$$(d_1\text{: } l_{1j}=v_{1jk},\ d_2\text{: } l_{2j}=v_{2jk},\ \ldots,\ d_i\text{: } l_{ij}=v_{ijk}, \ldots,\ d_m\text{: } l_{mj}=v_{mjk}) \rightarrow f_p = w_{pq}, [\alpha, \sqrt{b_1}].$$

For instance, if a node "Drinks, profit: $680K" is positively skewed at α = 0.05 and

$\sqrt{b_1}$ = 2.63, the corresponding Sk-navigation rule will be *product category = Drinks → profit = high [0.05, 2.63]* relative to its parent node *product category = All.*

The support (υ) of an Sk-navigation rule is defined by the number of transactions to establish the rule and is a useful parameter to control the number of rules generated by DISNAR. A very small sample size may produce a large number of rules, in which case υ can be used to restrict the rule generation by qualifying only those candidate nodes that satisfy the condition $n_2 \geq \upsilon$. However, DISNAR allows us to identify skewed datasets for a sample size (n_2) as small as 5. As we drill down to the near-lowest levels in dimensional hierarchies, the sample size is greatly reduced. To our advantage, test of skewness still qualifies for these small sample sizes.

It is important to note that Sk-navigation rules are neither association rules found in data mining nor production rules found in expert systems. Association rules cannot discover surprises, as they detect the most commonly found patterns as opposed to uncommon patterns. On the other hand, the primary utility of Sk-navigation rules is in aiding an OLAP user traverse the data cube efficiently and effectively.

Cube Traversal Using Sk-Navigation Rules

In this section, we formally define cube traversals using Sk-navigation rules. A rule also is called a *node of surprise,* because essentially it represents a lattice node containing a significant skewed pattern relative to its parent. A user begins the navigation with a root node (level-1 rules) and continues drilling down to children nodes until no further rules are detected. The rules guide the users to reach surprises of interest. The rules also provide a better choice of cube exploration for the following reasons:

1. They are simple to comprehend. It is easier to interpret an Sk-navigation rule than to manually scan a large dataset and try to find surprises, if any.
2. They are convenient to use. It is quite easy to navigate through Sk-navigation rules. Cube navigation is straightforward, because the rules can be understood and recollected better than the real datasets.
3. Cube exploration using Sk-navigation rules provides guidance for cube navigation, which does not exist in current OLAP tools.

The following properties of Sk-navigation rules and navigation paths are used in cube navigation:

1. A *navigation path* (np) is defined by the complete traversal from a root node to the leaf node. If a navigation path np_x contains a total of t Sk-navigation rules, then:

   $np_x = \{skr_i: 1 \leq i \leq t\}$.

   Here, np_x represents an ordered set of rules arranged from 1 to t in their order of navigation.
2. The level of navigation for a rule is determined by summing up the dimensional levels in its antecedent. For a rule skr_p that contains q dimensions in its antecedent ($q \leq n$), the level of navigation, l_{navig}, is given by:

   $$l_{navig}(skr_p) = \sum_{j=1}^{q} l_j.$$

   For example, $l_{navig}(skr_2)$ is 3 for a rule skr_2 *year = 1993, product subcategory = frozen food → profit=high [0.05, 1.53].*
3. A rule is the root node in a navigation path, if and only if it does not connect to a parent. It is defined as follows:

   $np_x = \{skr_i: 1 \leq i \leq t\}$
   $\exists\ sk_{j[parent]}: l_{navig}(skr_{j[parent]}) < l_{navig}(skr_i)$ for all $i > j$

   For example, *year = 1993 → profit = high [0.05, 1.79]* is a root node, because it does not have any parents.
4. A rule is the leaf node (end point) in a navigation path, if and only if it does not

extend to any children rules. The level of navigation for a leaf node ranges from 1 to ($\Sigma L_i - 1$). That is:

$np_x = \{skr_i: 1 \le i \le t\}$
$\exists\ sk_{j[leaf]}: l_{navig}(skr_{j[leaf]}) < l_{navig}(skr_i)$ for all $i \neq j$

5. Two rules, skr_1 and skr_2, exist in the same navigation path, if and only if one rule is an ancestor rule of another:

   $(skr_j, skr_k) \in np_x$ iff $skr_j = ancestor(skr_k)$ or $skr_k = ancestor(skr_j)$.

6. The length of a navigation path, len_{np}, represents the total number of rules in the complete traversal. A path is called a *shallow-navigation path* (np_{sh}) if len_{np} is equal to 1. The longest path, recognized by $len_{np} = (\Sigma L_i - 1)$, is called a *dense-navigation path* (np_{den}). We use dense-navigation paths to describe the most detailed paths to reach the lowest possible nodes of surprises.

An important property that arises from the Sk-navigation rules is that a rule may belong to multiple navigation paths, since the cuboids in the lattice are interlinked. A node of surprise such as in cuboid (T_1, P_1) can be reached either from cuboid (T_1) or from cuboid (P_1). For example, a user can navigate to *year = 1993, product category = food → profit = high [0.05, 1.53]* either from *year = 1993 → profit = high [0.05, 1.79]* or from *product category='food' → profit = high [0.05, 1.38]*.

Proposed Ways to Guide Through Navigation Paths

The properties of Sk-navigation rules explained in the previous section are used in our rule-driven system to navigate through data cubes. To steer the user to one of several possible navigation paths, three complementary yet independent controls available to the user are proposed.

1. **Level of Significance (α).** The level of significance is a key parameter in our rule-driven system. The system allows users to dynamically adjust α to view different sets of rules at the same lattice node. The value of α is user-defined and is initially set to 0.05 by default. A user, looking for only extreme surprises, just needs to decrease α to 0.01 or 0.005. Similarly, α can be relaxed to 0.1 for less-extreme surprises. Next, we explain the effect of change of α on discovery of Sk-navigation rules.
   If α_1 and α_2 are two levels of significance ($\alpha_1<\alpha_2$), a node nod_{curr} contains a Sk-navigation rule under the following circumstances:

   (a) $|\sqrt{b_1}_{[curr]}| > \sqrt{b_1}\,critical_{[curr]}$ at (n, α_1)
   In this case, skewness $\sqrt{b_1}_{[curr]}$ always will be greater than the critical skewness at (n, α_2) because $\alpha_1<\alpha_2$. Therefore, nod_{curr} contains a rule at both α_1 and α_2.
   (b) $|\sqrt{b_1}_{[curr]}| < \sqrt{b_1}\,critical_{[curr]}$ at (n, α_1), and $|\sqrt{b_1}_{[curr]}| > \sqrt{b_1}\,critical_{[curr]}$ at (n, α_2)
   Here, nod_{curr} contains a rule at α_2 but is rejected as a non-significant skewed node at α_1. This results in different sets of Sk-navigation rules at different αs.
   (c) $|\sqrt{b_1}_{[curr]}| < \sqrt{b_1}\,critical_{[curr]}$ at (n, α_1), and $|\sqrt{b_1}_{[curr]}| < \sqrt{b_1}\,critical_{[curr]}$ at (n, α_2)
   Because $\sqrt{b_1}_{[curr]}$ does not satisfy the critical skewness at either α_1 or α_2, nod_{curr} does not contain a rule at either of the levels of significance.

2. **Interestingness of Sk-Navigation Rules.** An important metric of discovered knowledge in association rules mining is the interestingness of rules, and it is considered an important post-data mining issue. Using this metric, users can prune the large number of rules generated to only the ones that are interesting. While it is difficult to formalize interestingness, it can take the form of unexpected rules or in terms of objective measures such

as support and confidence. Since our approach is based on detecting surprises, we adopt the interestingness of rules in terms of unexpectedness.

While navigating through the data cube, the user expects the children rules to provide additional details beyond what is provided by their parent in terms of surprises. If a parent rule contains more than one child rule, only one of the children is certain to conform to the parent's surprise. The remaining children rules may or may not encode the similar surprise as that of parent. For instance, while the parent rule shows a positive skewness, a subset of the same transaction set (a child rule) may behave differently and may show negative skewness. To examine this unexpectedness, we introduce a neighborhood-based categorization of Sk-navigation rules and then examine the rules for their unexpectedness based on their categories. The rules are grouped into three categories: expected, unexpected, and NA (not applicable), based on the skewness pattern for the pairs of parent and child. A rule is called *expected* if it matches the prior knowledge of its parent rule's consequent, meaning that the user is consistently navigating through the rules having the same nature of skewness. An *unexpected* rule indicates a perceptible change on the skewness while navigating from parent to child. The consequent of an unexpected rule differs from that of its parent. A root node is indicated by *NA* if it is the first node of navigation and does not have a parent rule with which to compare.

Let us assume that $\text{skr}_{\text{parent}}$ and $\text{skr}_{\text{child}}$ are the parent and child Sk-navigation rules. Then:

(a) $\text{skr}_{\text{child}}$ is *expected* if
$\text{ante}(\text{skr}_{\text{child}}) \supset \text{ante}(\text{skr}_{\text{parent}})$, and
$\text{cons}(\text{skr}_{\text{child}}) = \text{cons}(\text{skr}_{\text{parent}})$

(b) $\text{skr}_{\text{child}}$ is *unexpected* if
$\text{ante}(\text{skr}_{\text{child}}) \supset \text{ante}(\text{skr}_{\text{parent}})$, and
$\text{cons}(\text{skr}_{\text{child}}) \neq \text{cons}(\text{skr}_{\text{parent}})$

(c) $\text{skr}_{\text{parent}}$ is *NA* if
$l_{navig}(skr_{parent}) = 1$, or $\text{skr}_{\text{parent}}$ is a root node in its navigation path.

For instance, the algorithm discovered an unexpected surprise, *quarter = 1993-Q2 → profit = low* , when navigating from *year = 1993 → profit = high*. This categorization adds purposeful information to navigation paths. We can rank the navigation paths by their number of unexpected rules. Users then simply can drill down on the paths that contain a large number of unexpected Sk-navigation rules and examine the corresponding datasets that contain many highs and lows in the transactions.

3. **Dealing With Multiple Facts.** Our rule-driven system allows users to work simultaneously with many rules that contain multiple facts in their consequents. Given a navigation path, a user can study the behavior of different facts at the same time by looking at different consequents while maintaining the same antecedent. If skr_j and skr_k are two rules representing two different facts, then:

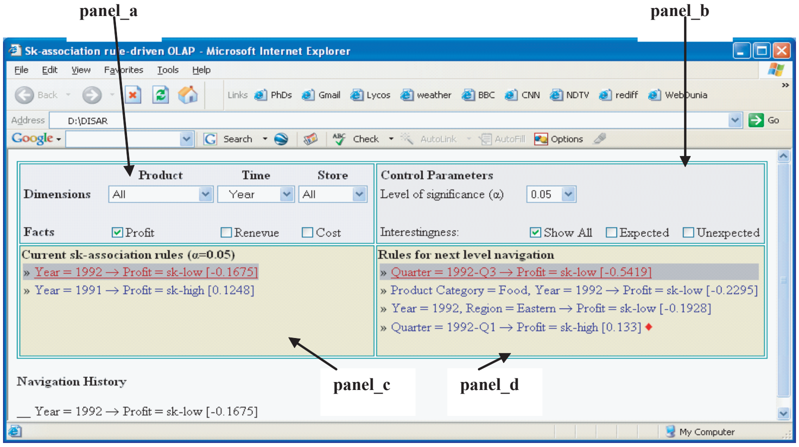
$l_{navig}(skr_j) = l_{navig}(skr_k)$,
$ante(skr_j) = ante(skr_k)$, and
$cons(skr_j) \neq cons(skr_k)$.

For example, two Sk-navigation rules, skr_1 *year = 1993 → profit = high [0.05, 1.79]* and skr_2 *year = 1993→ cost = low [0.05, -2.06]*, represent the same lattice node but refer to two different facts: *profit* and *cost*.

PROTOTYPE OF A RULE-DRIVEN Sk-NAVIGATION SYSTEM

In this section, we describe our rule-driven prototype and illustrate how it assists end users to explore data cubes and to reach low-level surprises. In order to discover surprises, our system supports drill-down and rollup opera-

Figure 3. Selecting a rule at year level from panel_c


panel_a
panel_b
Sk-association rule-driven OLAP - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Links PhDs Gmail Lycos weather BBC CNN NDTV rediff WebDunia
Address
Go
Google
Search
Check
AutoLink
AutoFill
Options
panel_c
panel_d
My Computer

Address D:\DISAR

	Product	Time	Store
Dimensions	All	Year	All
Facts	☑ Profit	☐ Renevue	☐ Cost

Control Parameters

Level of significance (α) 0.05

Interestingness: ☑ Show All ☐ Expected ☐ Unexpected

Current sk-association rules (α=0.05)

» Year = 1992 → Profit = sk-low [-0.1675]

» Year = 1991 → Profit = sk-high [0.1248]

Rules for next level navigation

» Quarter = 1992-Q3 → Profit = sk-low [-0.5419]

» Product Category = Food, Year = 1992 → Profit = sk-low [-0.2295]

» Year = 1992, Region = Eastern → Profit = sk-low [-0.1928]

» Quarter = 1992-Q1 → Profit = sk-high [0.133] ♦

Navigation History

__ Year = 1992 → Profit = sk-low [-0.1675]

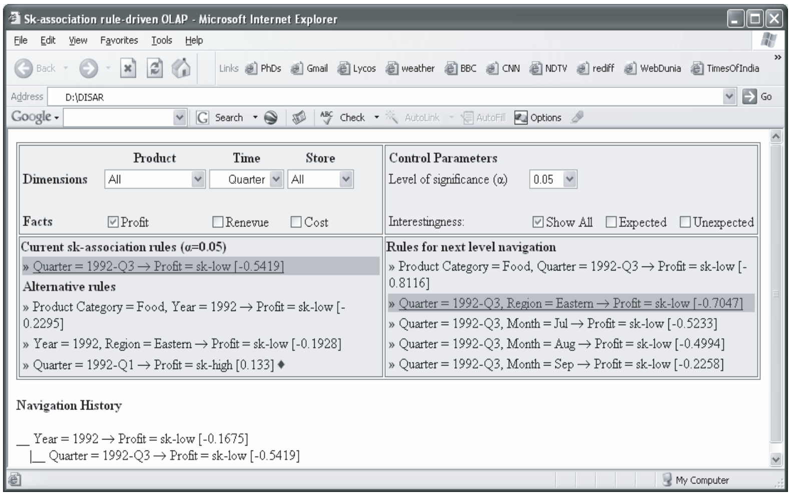
tions on Sk-navigation rules in the same manner as OLAP. In addition, the system provides novel methods of cube navigation and a navigation history to the end users.

We illustrate the rule-driven system with few sample screens for exploring a grocery data cube consisting of three dimensions, *Product*, *Time*, and *Store*, with the lowest dimensional levels as *Product Brand, Month,* and *City*, respectively. In Figure 3, the top left panel (panel_a) displays the three dimensions and their respective top-level hierarchies in drop-down lists. A user simply can select a hierarchical level available for navigating from the drop-down lists, such as *Category* from *Product, Year* from *Time,* or *Region* from *Store* dimension, and view the corresponding Sk-navigation rules. The user also can select one or more facts to view simultaneously (Profits, Revenue, Costs), which are displayed on the next row. Panel_c displays the current rules in *descending order of skewness* with the highest skewed rule at the top. If the user selects *Year* from *Time* dimension, the system displays all the rules containing only *Year* in panel_c. After selecting a rule in panel_c, the rule is highlighted with a different color, and its children rules (next navigation level) are presented in panel_d.

Let us assume that a user first selects a rule skr$_1$ *year = 1992* → *profit = Sk-low* in panel_c, as shown in Figure 3. The corresponding children rules are displayed in panel_d. For the next navigation level, the user selects a child rule from panel_d. For instance, when a rule skr$_2$ *quarter = 1992-Q3* → *profit = Sk-low* is selected from panel_d, it automatically is moved to panel_c, implying that the user has navigated to skr$_2$, which then becomes the current rule (Figure 4). As a result, the dropdown list for *Time* dimension changes from *Year* to *Quarter*. Unselected rules are moved from panel_d to panel_c and become available as the alternate paths. Panel_d is refreshed to display children rules for skr$_2$. By continuing the navigation in a similar fashion, a user finally reaches a low-level surprise, *product subcategory = 'Frozen Food', quarter = '1992-Q3', month = 'Jul', state = 'PA'* → *profit = Sk-low*.

Our system also provides the control parameters in the top right panel (panel_b). Users can dynamically adjust the level of significance (α) to study different sets of rules at the same cuboid. The possible values of α are 0.005, 0.01, 0.025, 0.05, and 0.1. Panel_c and panel_d are refreshed based on selection of α. For instance, a decrease in the value of α results in displaying

Figure 4. View after selecting a child rule at quarter level


Sk-association rule-driven OLAP - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Links PhDs Gmail Lycos weather BBC CNN NDTV rediff WebDunia TimesOfIndia
Address D:\DISAR
Go
Google
Search
Check
AutoLink
AutoFill
Options
My Computer

	Product	Time	Store
Dimensions	All	Quarter	All
Facts	☑ Profit	□ Renevue	□ Cost

Control Parameters

Level of significance (α) 0.05

Interestingness: ☑ Show All □ Expected □ Unexpected

Current sk-association rules (α=0.05)

» Quarter = 1992-Q3 → Profit = sk-low [-0.5419]

Alternative rules

» Product Category = Food, Year = 1992 → Profit = sk-low [-0.2295]

» Year = 1992, Region = Eastern → Profit = sk-low [-0.1928]

» Quarter = 1992-Q1 → Profit = sk-high [0.133] ♦

Rules for next level navigation

» Product Category = Food, Quarter = 1992-Q3 → Profit = sk-low [-0.8116]

» Quarter = 1992-Q3, Region = Eastern → Profit = sk-low [-0.7047]

» Quarter = 1992-Q3, Month = Jul → Profit = sk-low [-0.5233]

» Quarter = 1992-Q3, Month = Aug → Profit = sk-low [-0.4994]

» Quarter = 1992-Q3, Month = Sep → Profit = sk-low [-0.2258]

Navigation History

```
__ Year = 1992 → Profit = sk-low [-0.1675]
   |__ Quarter = 1992-Q3 → Profit = sk-low [-0.5419]
```

only the very highly skewed patterns, such as very high profit or very low profit.

We also utilize the interestingness of Sk-navigation rules to allow users to instantly view the expected, unexpected, or both types of patterns. Unexpected rules are differentiated visually from expected rules through a red diamond symbol following the rule presentation. For instance, one simply can navigate through unexpected rules by checking the unexpected checkbox, thereby filtering panel_c and panel_d to display only the unexpected rules.

Additionally, we provide a navigation history (shown at the bottom left) in order to trace the cube exploration by individual users. The history facilitates navigation by allowing the users to return to a previsited rule without the need to backtrack or to recollect the previous navigation steps. The navigation paths for individual users also are recorded in the database so that the history can be promptly brought back to the users the next time they log in.

EXPERIMENTAL RESULTS

This section presents results from a set of experiments to evaluate the performance of DISNAR algorithm by studying the quality and scalability of Sk-navigation rules. Specifically, we evaluate (a) the recall and precision of Sk-navigation rules, (b) the execution time, and (c) the space overhead. We further present the experimental results using a real-life dataset. All experiments were performed on a 1.7GHz Pentium IV machine with 512MB RAM running Windows XP. The algorithms were implemented in PL/SQL on an Oracle 10g database.

Experimental Setup

We adapted the grocery database (Kimball, 2002) in order to generate five test datasets, as shown in Table 1. The number of navigation nodes is obtained by summing up the total lattice nodes in all possible cuboids in the grocery data cube. The primary reason for using simulated data in experiments is to identify precisely the hidden surprises and to compare them to discovered Sk-navigation rules.

Each of the datasets contains three dimensions, *Product*, *Time*, and *Store*, and two facts, *Profit* and *Quantity*. The dimensional hierarchies are *Product* {*Category*, *Subcategory*, *Brand*}, *Time* {*Year*, *Quarter*, *Month*}, and *Store* {*Region*, *State*, *City*}. The number of attributes at the lowest level varied from 16 to 64 for *Product*, 20 to 192 for *Time*, and 14 to 31 for *Store* dimension. We generated surprises

Table 1. Summary of the five experimental datasets

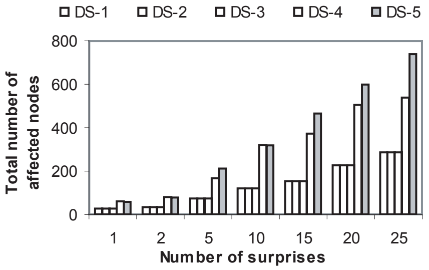
Dataset	# Records	# Nodes of Navigation
DS-1	4,480	5,893
DS-2	32,240	37,119
DS-3	97,384	107,095
DS-4	20,736	131,759
DS-5	190,464	1,035,893

that represent transactions containing 15-20% high or low profit values compared to the rest of the transaction set. The number of transactions with surprises varied from 1 to 25 for each of the datasets. The surprises also were generated in a manner such as to conceal their presence at the higher levels of aggregations (e.g., the aggregate *Product category=Drinks* would not show the surprises in transactions *Product Brand = Pepsi* and *Product Brand = Ahold 2% Milk*).

Due to the nature of navigating the lattice nodes, a surprise at the lowest level affects its higher-level lattice nodes, such as a surprise at a node *city = Fairfax* will influence two additional nodes, *state = VA* and *region = Eastern*. We measure this phenomenon by defining an influence rate, which is calculated as the number of nodes affected divided by the total number of nodes. Figure 5 shows the number of nodes in the grocery cube lattice that were affected by the lowest-level surprises. For instance, in dataset DS-5, the existence of only one lowest-level surprise (affecting a single transaction), *Product subcategory = Orange Juice, month = 1991-Q1_Jan, city = Baltimore*, affected 62 nodes from a total of 1,035,893 nodes in the lattice, giving an influence rate of 0.005%. As expected, increasing the number of surprises to five in the same dataset resulted in a higher influence rate of 0.020% by affecting a total of 215 nodes. At the maximum, when 25 lowest-level surprises existed for DS-5, the influence rate was 0.071%, affecting 738 nodes.

Apart from varying the number of surprises, we also tested the DISNAR algorithm by varying the level of significance (α) from 0.005 to 0.1. The support of Sk-navigation rules (υ) was set to 5 throughout the experiments.

Figure 5. Surprise vs. total affected nodes



Quality of Sk-Navigation Rules

The purpose of this set of experiments was to examine how many true surprises were discovered (= recall), and then to analyze the percentage of Sk-navigation rules that actually were the true surprises (= precision). Figure 6 shows how a change in α affects the recall and precision of discovered rules for DS-5 for different levels of surprises.

We observed that DISNAR detected the surprises with high recall and precision values at $\alpha = 0.05$. A high recall was achieved for every set of surprises ranging from 1 to 25. For instance, DISNAR discovered 519 Sk-navigation rules when the number of surprises was 20 and α was 0.05, thereby resulting in a recall of 75.21%. These 519 rules (i.e., nodes) were discovered from a total of 1,035,893 lattice nodes. The recall increased to 88.11% when α was increased to 0.1 as a result of detecting more surprises. When

Figure 6. Effect of α on (a) recall (on left) and (b) precision (on right) for DS-5

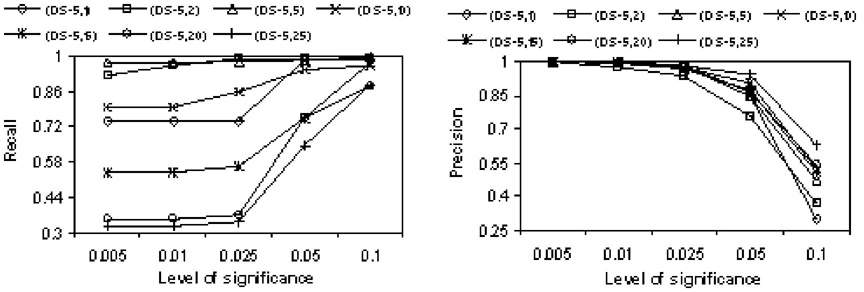
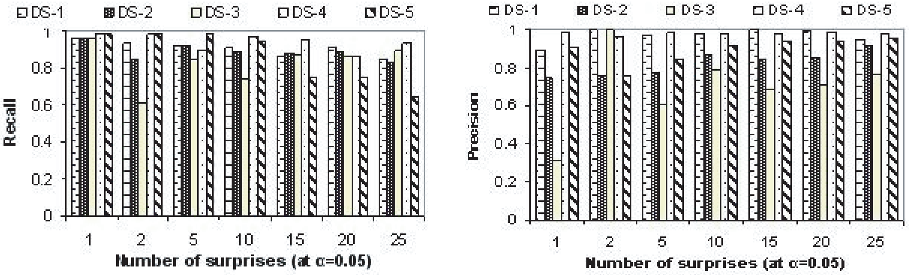


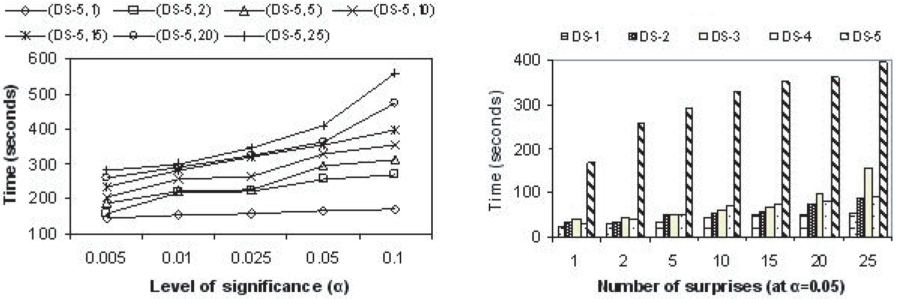
Figure 7. (a) recall (on left) and (b) precision (on right) for five datasets at α = 0.05



the number of surprises was set to 20 or 25, as expected, the recall was relatively smaller at α equal to 0.005 and 0.01. This happened because the impact of aggregating 20 to 25 surprises with variability in the degree of surprise results in low skewness. However, these surprises later were detected when α was increased to 0.05 or 0.1, thus ensuring a high recall. Noticeably, the algorithm achieved a precision of 1 when α was 0.005, because it discovered only very highly skewed patterns and did not allow any low skewed patterns to qualify as surprises. Figure 7 shows the recall and precision for all five datasets at α set at 0.05. Again, we observe that DISNAR achieved high recall and precision values for different datasets. Complete results on recall and precision for additional four datasets are included in Appendix A.

In Figure 6, we observe that the recall increases with a lower precision for higher αs. This is attributed to low critical skewness with an increase in α. The algorithm detects the low skewed patterns, which are not significant otherwise at smaller αs. Discovery of these less-evident surprises results in a high recall. However, relaxing critical skewness to a lower value also discovers some false surprises, because some nodes with relatively small skewness also qualify as surprises, resulting in a smaller precision. A practical implication of this variation in α is that some surprises are apparent, while others are not. Therefore, we allow users to dynamically adjust α to pursue either high- or low-skewed surprises.

Figure 8. Execution time for (a) DS-5 as a function of α (on left) and (b) five datasets at α = 0.05 (on right)



Execution Time

Figure 8(a) shows the execution time to discover different sets of surprises for DS-5 as a function of α. The graph suggests an increase in execution time for higher αs. This is expected, since at higher αs, low value of the critical skewness results in a larger number of lattice nodes being examined. However, even for the largest dataset DS-5, DISNAR discovered the surprises within a reasonable processing time. For instance, it took only 4.28 minutes to discover 334 rules with a number of surprises equal to 10 and α set at 0.05. These rules were identified from a total of 1,035,893 lattice nodes, resulting in a processing time of 0.248 seconds per thousand nodes. The execution time reached a maximum 9.316 minutes when 653 surprises were discovered for DS-5 at a peak α value of 0.1, resulting in a processing time of only 0.5396 seconds per thousand nodes.

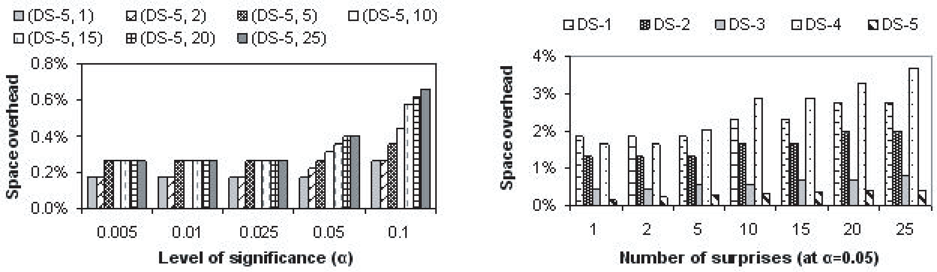
Effective pruning of nodes, a feature of the DISNAR algorithm, plays an important role in the total execution time. When the number of surprise rules increased from 215 to 738 (a factor of 3.43) for DS-5 at α set at 0.05, the execution time increased by only by a factor of 1.35 from 258 to 396 seconds. DISNAR's knowledge discovery process begins at the root nodes. Subsequent nodes are selected for further traversal, only if they satisfy the minimum significant skewness. A node with insignificant skewness immediately is pruned from the discovery process. Hence, even if the number of surprises increases in the dataset, it does not linearly affect the number of searchable nodes in the lattice. The execution time graphs for the other four datasets are included in Appendix B.

We also compare the execution time for different datasets at α set at 0.05. As shown in Figure 8(b), the execution time for datasets other than DS-5 is much lower than for DS-5. For DS-1, it took less than 1.67 minutes to discover 258 rules. It took 1.70 minutes to discover 516 rules for DS-4. In comparison, it took 6.07 minutes to discover 519 rules for DS-5. Although DS4 and DS5 discovered about the same number of rules, they had a huge difference in execution time because of the variation in their respective search spaces. While 516 surprises were discovered from 131,759 lattice nodes for DS-4, a total of 1,035,893 nodes were searched to detect 519 rules for the larger dataset DS-5.

Space Overhead

The Sk-navigation rules are stored in an Oracle database using a relational schema as suggested in Kumar, Gangopadhyay, and Karabatis (2005). Each rule contains the details on its antecedent, consequent, measure of skewness, α, the parent rule, and the interestingness. We compare the storage space (in kilobytes) required for storing Sk-navigation rules with the dataset size to compute the percent of space

Figure 9. Storage cost for (a) DS-5 as a function of α (on left) and (b) five datasets at α = 0.05 (on right)



overhead. Figure 9(a) shows the effect of change in α on space overhead for DS-5 at different number of surprises. Although the overhead was expected to increase at higher αs, it ranged only from 0.18% to 0.66%. It reached a peak 0.66% when the largest number of rules (1,034 rules) was discovered with 25 surprises. Thus, compared to the dataset size, a modest storage space is required to store the highly meaningful set of Sk-navigation rules. The space overhead graphs for other four datasets are included in Appendix C.

In Figure 9(b), we compare the space overhead for different datasets at α = 0.05. The graph shows a decrease in percentage space overhead for larger datasets. For instance, DS-1 and DS-4 are the two smallest datasets (Table 1) that also exhibit a higher space overhead. On the other hand, the largest dataset DS-5 resulted in the lowest space overhead. It is understandable, because the number of discovered Sk-navigation rules is not linearly dependent on the size of dataset, and the marginal rate of rule generation decreases as the dataset size increases. DS-5 was approximately 11 times larger than DS-1, but the respective number of discovered rules was 502 and 258 at 25 surprises with α set at 0.05. Clearly the number of rules did not increase in direct proportion to the size of datasets.

The Crash Data Analysis

We also tested DISNAR on a vehicle crash dataset obtained from the Maryland Department of Transportation (Bapna & Gangopadhyay, 2005). The dataset contained 534,941 crash records ranging from the years 1993 to 2000, in which an individual crash record detailed the location of the crash, the day and time, the severity of the crash, and the crash cost. DISNAR discovered 372 expected and 52 unexpected Sk-navigation rules within a total of 2.06 minutes. Manual examination of the rules confirmed their surprising properties. Some hidden surprises were not comprehendible by simple examination of the large dataset but were discovered by DISNAR as unexpected rules. For example, the crash cost for both *Severity = possible injury* and *Day_of_Week = Tuesday* were low, but the algorithm discovered an unexpectedly high crash cost in Howard County on Tuesdays for crashes resulting in possible injuries, which was not noticeable at higher levels of aggregation.

CONCLUSION AND FUTURE WORK

In this article, we presented the DISNAR algorithm in order to discover hidden surprises in data cube lattices. The algorithm assists end users to navigate through data cubes that otherwise would be difficult to steer due to the combinatorial explosion in the number of possible navigation paths. This navigation capability was conceptualized in terms of the discovery of hidden surprises in the dataset. Three complementary and independent means to navigate a data cube were used: the level of significance as measured by the skewness between the parent node and a child node, interestingness of surprises, and the ability to handle multiple facts. A prototype system implemented the DISNAR algorithm and was tested on simulated grocery datasets and a real-world crash dataset. The algorithm discovered hidden surprises for different grocery datasets with a high recall and precision within a reasonable execution time. It also discovered hidden surprises in the crash dataset. The effectiveness of pruning nodes, based on the test of significance of skewness, led to a considerable decrease in the number of nodes that were examined for traversal. Thus, not only is the traversal of the data cube possible in reasonable execution time, but the Sk-navigation rules also were stored with slight overhead.

We are extending our work on the concept of navigation to deal with scenarios in which possibly a large number of Sk-navigation rules are generated. Specifically, we are working on an axis-shift concept in order to determine interestingness of navigation paths, which ranks the paths based on the distance metrics of Sk-navigation rules and then provides a comparison matrix to study the usefulness of navigation paths. This would allow users simply to navigate along selective paths on which highly extreme surprises are present. Furthermore, we are implementing additional exploration capabilities in our rule-driven system in which users can view all possible navigation paths (the children and further generations until the lowest level) from an Sk-navigation rule in a tree structure and then delve directly to a rule of interest. Another enhancement in our prototype would allow users to view the corresponding transaction set for the rule sorted in such a way that highly skewed transactions are displayed clearly at the top.

Association rule mining and OLAP traditionally have been viewed from different perspectives in order to aid in the knowledge acquisition from data warehouses. While our approach deals with navigation paths in OLAP environments, it produces rules that are presented in a manner similar to association rules. We currently are examining kurtosis-based enhancements to the algorithm in order to generate all types of rules, including rules that are commonly present in the data set. Thus, our approach may bridge the gap between OLAP and data mining.

REFERENCES

Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining associations between sets of items in large databases. In *Proceedings of the ACM SIGMOD*.

Aumann, Y., & Lindell, Y. (1999). A statistical theory for quantitative association rules. In *Proceedings of the ACM SIGKDD.*

Bapna, S., & Gangopadhyay, A. (2005). A Web-based GIS for analyzing commercial motor vehicle crashes. *Information Resources Management Journal, 18*(3), 1–12.

Bauer, A., Hümmer, W., et al. (2000). *An alternative relational OLAP modeling approach*. Proceedings of the DAWAK.

Bi, Z., Faloutsos, C., & Korn, F. (2001). *The dgx distribution for mining massive, skewed data*.

Boulicaut, J., Marcel, P., & Rigotti, C. (1999). Query driven knowledge discovery in multidimensional data. In *Proceedings of the ACM DOLAP.*

Bouzeghoub, M., Fabret, F., & Matulovic, M. (1999). Modeling data warehouse refreshment process as a workflow application. In *Proceedings of the International Workshop on Design and Management of Data Warehouses, DMDW*, Heidelberg, Germany.

Casali, A., Cicchetti, R., & Lakhal, L. (2003). Cube lattices: A framework for multidimensional data mining. In *Proceedings of the SDM*.

Chan, C.Y., & Ioannidis, Y.E. (1998). Bitmap index design and evaluation. In *Proceedings of the ACM SIGMOD.*

Chen, Q. (1999). *Mining exceptions and quantitative association rules in OLAP data cubes*. Simon Fraser University.

Chen, Y., Dehne, F., Eavis, T., & Rau-Chaplin, A. (2006). Improved data partitioning for building large ROLAP data cubes in parallel. *International Journal of Data Warehousing and Mining, 2*(1), 1–26.

Choong, Y.-W., Laurent, D., & Marcel, P. (2001). Computing appropriate representations for multidimensional data. In *Proceedings of the ACM 4th International Workshop on Data Warehousing and OLAP*.

D'Agostino, R., & Stephens, M. (1986). *Goodness-of-fit techniques*. New York: M. Dekker.

Deshpande, P.M., Ramasamy, K., Shukla, A., & Naughton, J.F. (1998). *Caching multidimensional queries using chunks*. Proceedings of the ACM SIGMOD, Seattle, Washington, USA.

Fabris, C.C., & Freitas, A.A. (2006). Discovering surprising instances of Simpson's paradox in hierarchical multidimensional data. *International Journal of Data Warehousing and Mining, 2*(1), 27–49.

Fu, L. (2005). Novel efficient classifiers based on data cube. *International Journal of Data Warehousing and Mining, 1*(3), 15–27.

Ghoting, A., Otey, M., & Parthasarathy, S. (2004). *LOADED: Link-based outlier and anomaly detection in evolving data sets*. ICDM.

Golfarelli, M., Maio, D., & Rizzi, S. (2000). Applying vertical fragmentation techniques in logical design of multidimensional databases. In *Proceedings of the DAWAK*.

Gupta, A., Mumick, I.S., Rao, J., & Ross, K.A. (2001). Adapting materialized views after redefinitions: Techniques and a performance study [Special issue on Data Warehousing]. *Information Systems*.

Gupta, H., Harinarayan, V., Rajaraman, A., & Ullman, J. (1997). Index selection for OLAP. In *Proceedings of the ICDE*.

Gupta, H., & Mumick, I.S. (1999). Selection of views to materialize under a maintenance-time constraint. In *Proceedings of the International Conference on Database Theory*, Jerusalem, Israel.

Han, J. (1998). Towards on-line analytical mining in large databases. In *Proceedings of the ACM SIGMOD*.

Han, J., Chee, S., & Chiang, J. (1998). Issues for on-line analytical mining of data warehouses. In *Proceedings of SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery, DMKD*, Seattle, Washington, USA.

Hurtado, C., & Mendelzon, A. (2001). Reasoning about summarizability in heterogeneous multidimensional schemas. In *Proceedings of the ICDT*.

Inmon, W.H. (1996). *Building the data warehouse*. New York: John Wiley & Sons.

Kaser, O., & Lemire, D. (2003). Attribute value reordering for efficient hybrid OLAP. In *Proceedings of the DOLAP*, New Orleans, Louisiana.

Kim, W.Y., Lee, Y.K., & Han, J. (2004). CC-Mine: Efficient mining of confidence-closed correlated patterns. In *Proceedings of the PAKDD*.

Kimball, R. (2002). *The data warehouse toolkit*.

Klosgen, W. (2002). Subgroup discovery. In W. Klosgen, & J.M. Zytkow (Eds.), *Handbook of data mining and knowledge discovery* (pp. 354–361). New York, Oxford University Press.

Kumar, N., Gangopadhyay, A., & Karabatis, G. (2005). Supporting mobile decision making with association rules and multi-layered caching. *Decision Support Systems*.

Lemire, D. (2002). Wavelet-based relative prefix sum methods for range sum queries in data cubes. In *Proceedings of the CASCON*, Toronto, Canada.

Maniatis, A., Vassiliadis, P., Skiadopoulos, S., Vassiliou, Y., Mavrogonatos, G., & Michalarias, I. (2005). A presentation model & non-traditional visualization for OLAP. *International Journal of Data Warehousing and Mining, 1*(1), 1–36.

Marcel, P. (1999). Modeling and querying multidimensional databases: An overview.

Networking and Information Systems Journal, 2(5–6), 515–548.

Mendelzon, A., & Vaisman, A. (2000). Temporal queries in OLAP. In *Proceedings of the VLDB*, Cairo, Egypt.

Park, C.-S., Kim, M.H., & Lee, Y.-J. (2001). Rewriting OLAP queries using materialized views and dimension hierarchies in data warehouses. In *Proceedings of the ICDE*.

Poon, C.K. (2003). Dynamic orthogonal range queries in OLAP. *Theoretical Computer Science, 296,* 487–510.

Sarawagi, S. (1997). Indexing OLAP data. *IEEE Data Engineering Bulletin*.

Sarawagi, S. (1999). Explaining differences in multidimensional aggregates. In *Proceedings of the VLDB*. Morgan Kaufmann.

Sarawagi, S. (2000). User-adaptive exploration of multidimensional data. In *Proceedings of the VLDB*.

Sarawagi, S., Agrawal, R., & Megiddo, N. (1998). Discovery-driven exploration of OLAP data cubes. In *Proceedings of the International Conference on Extending Database Technology*.

Theodoratos, D., & Sellis, T. (1999a). Designing data warehouses. In *Proceedings of the Data and Knowledge Engineering, DKE*.

Theodoratos, D., & Sellis, T. (1999b). Dynamic data warehouse design. In *Proceedings of the International Conference on Data Warehousing and Knowledge Discovery, DaWaK*, Florence, Italy.

Tjioe, H.C., & Taniar, D. (2005). Mining association rules in data warehouses. *International Journal of Data Warehousing and Mining, 1*(3), 28–62.

Vaisman, A., & Mendelzon, A. (2001). A temporal query language for OLAP: Implementation and a case study. In *Proceedings of the DBPL*, Rome, Italy.

Xiong, H., Tan, P.-N., & Kumar, V. (2003). Mining strong affinity association patterns in data sets with skewed support distribution. In *Proceedings of the ICDM.*

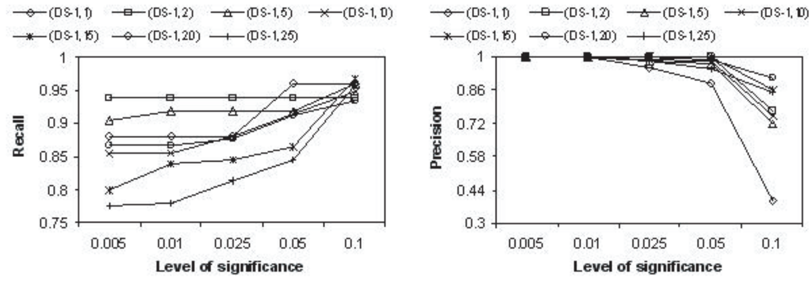
Yang, J., Karlapalem, K., & Li, Q. (1997). Algorithms for materialized view design in data warehousing environment. In *Proceedings of the VLDB*.

Yang, J., & Widom, J. (2000). Making temporal views self-maintainable for data warehousing. In *Proceedings of the 7th International Conference on Extending Database Technology*, Konstanz, Germany.

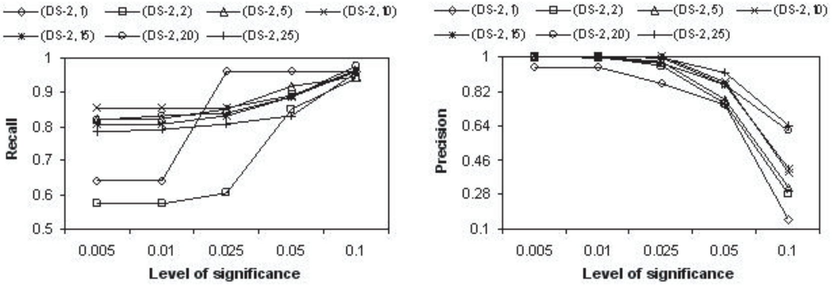
Zhang, J., Bloedorn, E., Rosen, L., & Venese, D. (2004). Learning rules from highly unbalanced data sets. In *Proceedings of the ICDM*.

APPENDIX A. RECALL AND PRECISION

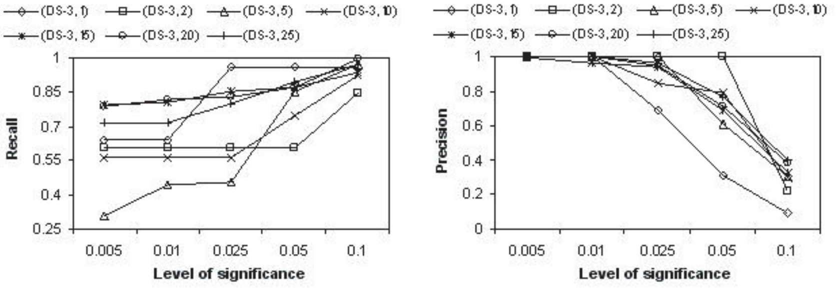
Dataset-1



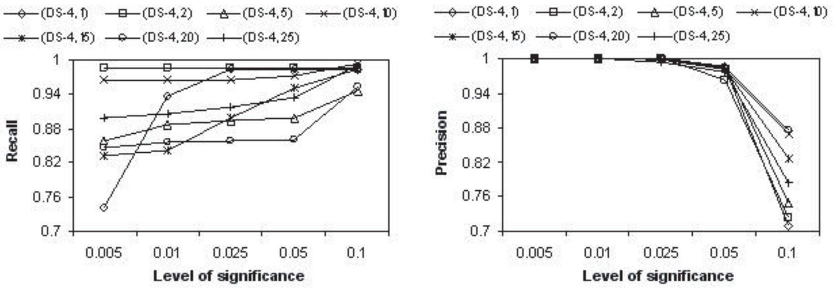
Dataset-2



Dataset-3

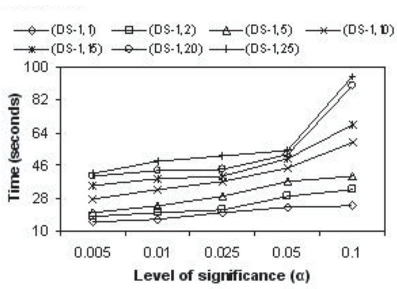


Dataset-4

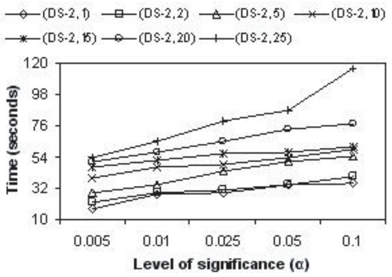


APPENDIX B. EXECUTION TIME

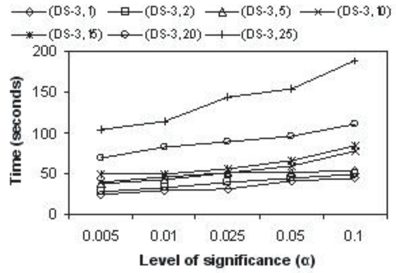
Dataset-1



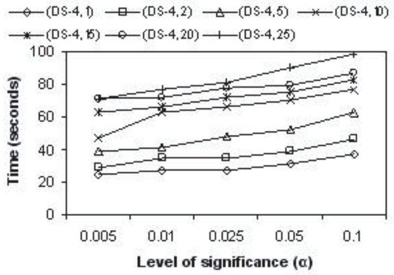
Dataset-2



Dataset-3

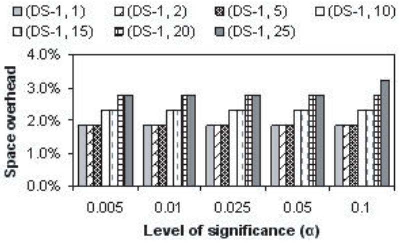


Dataset-4

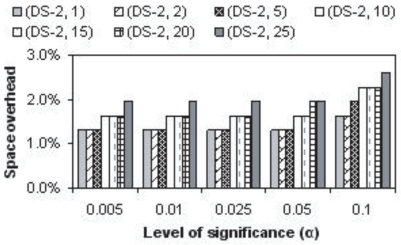


APPENDIX C. SPACE OVERHEAD

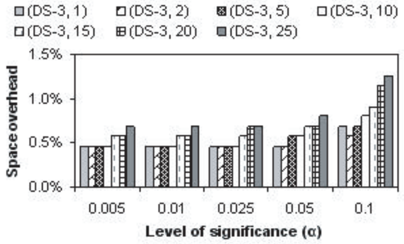
Dataset-1



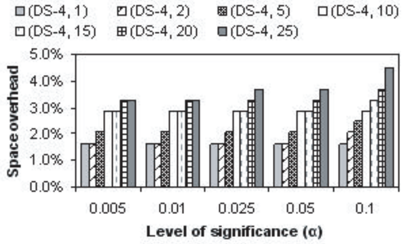
Dataset-2



Dataset-3



Dataset-4



Navin Kumar is a PhD candidate at the Department of Information Systems at the University of Maryland, Baltimore County. He received his MS in information systems from the University of Maryland, Baltimore County, and BTech(Honors) in industrial engineering and management from Indian Institute of Technology, Kharagpur. His main research interests include data warehousing and mining, knowledge discovery, and mobile commerce.

Aryya Gangopadhyay (gangopad@umbc.edu) is an associate professor of information systems at the University of Maryland, Baltimore County (UMBC). He has a PhD in computer information systems from Rutgers University. His research interests include decision support using data warehousing and mining, and database applications in geographic information systems and healthcare informatics. He has co-authored and edited three books, many book chapters, and numerous papers in journals.

Dr. George Karabatis is an assistant professor of information systems at the University of Maryland, Baltimore County (UMBC). He holds degrees in Computer Science (PhD and MS) and mathematics (BS). He is pursuing research on various aspects of information technology related to databases systems, applications for wireless handheld devices. Prior to his current appointment he was a research scientist at Telcordia Technologies (formerly Bellcore) where he led several telecommunications projects involving database related technologies. His work has been published in journals, conference proceedings and book chapters.

Sanjay Bapna is an associate professor of information science and systems at Morgan State University and has published articles in the areas of data mining, intelligent transportation systems, geographic information systems, and decision support systems. He has received numerous external funded grants in the area of ITS. His current research interests are in the area of privacy and anonymity.

Zhiyuan Chen received a PhD in computer science from the Cornell University in 2002. Presently, he is an assistant professor in the information systems department at University of Maryland, Baltimore County. His research interests include XML and semi-structured data, privacy-preserving data mining, data integration, automatic database tuning, and database compression.