Discovering Surprising Instances of Simpson's Paradox in Hierarchical Multidimensional Data

Carem C. Fabris
CPGEI, CEFET-PR
Av. Sete de Setembro, 3165
Curitiba-PR, 80215-901, Brazil
caremf@uol.com.br

Alex A. Freitas
Computing Laboratory
University of Kent
Canterbury, CT2 7NF, UK
A.A.Freitas@kent.ac.uk

Abstract

This paper focuses on the discovery of surprising, unexpected patterns, based on a data mining method that consists of detecting instances of Simpson's paradox. By its very nature, instances of this Paradox tend to be surprising to the user. Previous work in the literature has proposed an algorithm for discovering instances of that paradox, but it addressed only "flat" data stored in a single relation. This work proposes a novel algorithm that considerably extends that previous work, by discovering instances of Simpson's paradox in hierarchical multidimensional data – the kind of data typically found in data warehouse and OLAP environments. Hence, the proposed algorithm can be regarded as integrating the areas of data mining and data warehousing, by using an adapted data mining technique to discover surprising patterns from data warehouse and OLAP environments.

Keywords: Simpson's paradox, data mining, data warehouse, OLAP, multidimensional data.

INTRODUCTION

In general data mining consists of discovering interesting hidden patterns or previously unknown relationships in data. However, the question of what properties the discovered patterns should have, in order to be considered "interesting", is still an open problem.

The majority of data mining algorithms focus on the discovery of accurate patterns. This is particularly the case in the three data mining tasks most investigated in the literature, namely classification, clustering and association (Fayyad et al., 1996).

Another criterion that is also used quite often to evaluate discovered patterns (though not nearly so often as accuracy) is comprehensibility. Pattern comprehensibility is important in order to allow the user to validate and interpret discovered patterns, giving the user an insight that can be effectively used to make intelligent decisions.

There is, however, another criterion to evaluate discovered patterns that has been relatively less explored in the literature, namely the *surprisingness* of discovered patterns. First of all, it should be noted that accuracy and comprehensibility do not imply surprisingness, and discovering surprising patterns seems more difficult than discovering accurate and comprehensible patterns. As a simple, classic example of this point, consider the following rule, which could be discovered from a hypothetical medical database: "IF (patient is pregnant) THEN (patient is female)." Clearly, this rule is highly accurate. It is also highly comprehensible – i.e., it is very short and simple, easy to be interpreted, referring to attribute values whose meaning are very well-known. However, this rule is not surprising at all, representing an obvious fact, and so it is useless to the user.

The challenge addressed in this paper is to discover surprising patterns in data. There has been some work addressing this challenge, mainly in the context of the classification or association tasks (Liu & Hsu, 1996; Silberschatz & Tuzhilin, 1996; Suzuki, 1997; Liu et al., 1997; Padmanabhan & Tuzhilin, 1998; Suzuki & Kodratoff, 1998; Freitas, 1998; Dong & Li, 1998; Sahar, 2002; Carvalho et al., 2003; Ohsaki et al., 2004; Romao et al., 2004). However, there are two major differences between those projects and the work presented in this paper.

First, this work does not address the classification or association tasks. Rather, it focuses on the detection of instances of Simpson’s paradox, which will be explained in section 2. In terms of the data mining tasks described in (Fayyad et al., 1996), the detection of Simpson’s paradox seems more closely related to the task of deviation detection, although it differs from most deviation detection techniques in the sense that it does not require the specification of what is a “normal” relationship from which deviations should be found. Hence, when detecting instances of Simpson’s paradox the system has more autonomy. This is a significant difference between this work and typical projects on deviation detection, such as (Matheus et al., 1996).

Second, instead of trying to select the most surprising rules out of many discovered rules – as it is done in most of the previously mentioned projects on rule surprisingness – the system presented in this paper was designed specifically for discovering only surprising patterns. As will be argued in Section 2, Simpson’s paradox is surprising because in general the user cannot find an explanation for the “paradox”. In other words, the kind of pattern represented by an instance of Simpson’s paradox is, by its very nature, a *surprising* pattern to most users.

There is a previous work alerting for the pitfalls associated with Simpson's paradox in the context of data mining (Glymour et al., 1997). However, this paper follows a very different research direction consisting of exploiting the surprisingness of that paradox, making the detection of Simpson's paradox the central goal of a data mining algorithm explicitly designed to discover surprising patterns. This research direction was suggested by (Freitas, 1998), which proposed an algorithm for detecting instances of Simpson's paradox. However, that work did not implement the proposed algorithm. Later the algorithm was implemented and evaluated in public domain data sets by (Fabris & Freitas, 1999). However, those two projects involved only "flat" data sets, with no hierarchical dimensions, where each data set was represented by a single relation (corresponding to the universal relation of a relational database system).

This work proposes a significant extension to those previous projects, by extending the previous Simpson's paradox-discovery algorithm to a very different kind of data set, namely data cubes stored in a star format with hierarchical dimensions, as typically found in data warehouse (Kimball & Ross, 2002) and OLAP environments (Thomsen, 2002). The extended algorithm is then evaluated in a real-world data cube with 5 hierarchical dimensions.

Hence, this work obtains an *integration between data mining and data warehouse*, in the sense that it proposes to adapt a data mining method for discovering surprising patterns to the typical data warehouse/OLAP environment of hierarchical multidimensional data. From a data mining viewpoint, this has the benefit of making the data mining method in question more widely applicable, considering the increasing popularity of hierarchical multidimensional data. From an OLAP viewpoint, this has the benefit of extending the functionality and increasing the degree of autonomy of typical OLAP tools.

The remainder of this paper is organized as follows. Section 2 presents a review of Simpson’s paradox. Section 3 briefly describes the above-mentioned algorithm for detecting instances of Simpson’s paradox in “flat” data stored in a single relation. Section 4 proposes the above-mentioned novel extension of that algorithm, designed for mining hierarchical multidimensional data. Section 5 presents computational results. Finally, Section 6 concludes the paper.

SIMPSON’S PARADOX

This section presents a brief review of Simpson’s paradox, based on (Fabris & Freitas, 1999). See also (Wagner, 1982; Newson, 1991) for further discussions about this kind of paradox.

Let a population be partitioned into two mutually exclusive and exhaustive populations, denoted Pop_1 and Pop_2, according to the value of a given binary attribute, denoted *1stPartAtt* (First Partitioning Attribute). Let each of the populations Pop_1 and Pop_2 be further partitioned, in parallel, according to the values of a given categorical attribute, denoted *2ndPartAtt* (Second Partitioning Attribute), with m values in its domain. This will partition Pop_1 and Pop_2 into m subpopulations, denoted $Pop_{11},\dots Pop_{1m}$ and $Pop_{21},\dots,Pop_{2m}$.

Let G be a binary goal attribute, which takes on a value indicating whether or not a given situation of interest has occurred in a population. We are interested in analyzing the variation of the probability of the situation of interest across the above-mentioned partitions, i.e., investigating the relationship between the values of the partitioning attributes and the probability of the situation of interest. In order to do this analysis, let us use the following notation:

- G_1 and G_2 denote the attribute G value representing the situation of interest in each of the respective populations Pop_1 and Pop_2;
- $Pr(G_1)$ and $Pr(G_2)$ denote the probability that the situation of interest has occurred in populations Pop_1 and Pop_2, respectively;
- G_{ij} denotes the attribute G value representing the situation of interest in the subpopulation Pop_{ij}, where i is the id of the value of *1stPartAtt* (i=1,2) and j is the id of the value of *2ndPartAtt* (j=1,...,m);
- $Pr(G_{ij})$ denotes the probability that the situation of interest has occurred in the subpopulation Pop_{ij}, i=1,2 and j=1,...,m.

Formally, Simpson's paradox (Simpson, 1951) occurs when either of the following two logical expressions is satisfied:

(a) $Pr(G_1) > Pr(G_2)$ and $Pr(G_{1j}) \leq Pr(G_{2j})$ for j=1,...,m;
or,
(b) $Pr(G_1) < Pr(G_2)$ and $Pr(G_{1j}) \geq Pr(G_{2j})$ for j=1,...,m.

Expression (a) means that the paradox occurs when, although the probability of the situation of interest is higher in Pop_1 than in Pop_2, in *each* of the categories produced by *2ndPartAtt* the probability of the situation of interest in Pop_1 is lower than or equal to its value in Pop_2. The paradox also occurs in the dual situation represented by expression (b).

One of the earliest real-life examples of this paradox occurred in a comparison of tuberculosis deaths in New York City and Richmond, Virginia, during the year 1910, as shown in Table 1, where the situation of interest measured by attribute G was the *occurrence of death* in a tuberculosis case;

the *1stPartAtt* was *city*; and the *2ndPartAtt* was *racial category*. As can be observed in the table, overall the tuberculosis mortality rate of New York was lower than Richmond's one. However, the opposite was observed when the data was further partitioned according to two racial categories: white and non-white. In both the white and non-white categories, Richmond had a lower mortality rate.

It should be noted that an instance of Simpson's paradox has an explanation. In the example of Table 1, this explanation is as follows. As shown in the table, taking a look at the disaggregated data (partitioned by both city and racial category), the mortality rates of non-white people (0.56% and 0.33%) were in general higher than the mortality rates of white people (0.18% and 0.16%) in both cities. The opposite conclusion is reached by taking a look at the aggregated data (partitioned only by city) because the proportion of non-white people in Richmond was 36.6% (46,733 / 127,682), a value much larger than the proportion of non-white people in New York, which was only 1.9% (91,709 / 4,766,883). In other words, the paradox occurred because the two racial categories white and non-white had different mortality rates and these two categories were present in considerably different proportions in the cities of New York and Richmond.

Table 1: Simpson's paradox in data about tuberculosis deaths.

	New York		**Richmond**	
Total Population	4766883		127682	
No. of deaths	8878		286	
Percentage	0.19%		0.22%	
	New York		**Richmond**	
	white	**non-w.**	**white**	**non-w.**
Total Population	4675174	91709	80895	46733
No. of deaths	8365	513	131	155
Percentage	0.18%	0.56%	0.16%	0.33%

The existence of such explanations, which are consistent with the data, have led some statisticians to point out that Simpson’s paradox is not really a paradox (De Groot & Schervish, 2002). This issue depends, of course, on how one defines a paradox. According to Rescher (2001): “... *a paradox arises when a set of individually plausible propositions is collectively inconsistent. And the inconsistency at issue must be real rather than merely seeming*" (p. 6). In Table 1 the aggregated data is associated with the proposition that the mortality rate was higher in Richmond than in New York; and the disaggregated data is associated with the proposition that in both racial categories the mortality rate was higher in New York than in Richmond. The inconsistency between these two individually-plausible propositions can be explained in a way that is consistent with the observed data, as explained earlier, so that the inconsistency is “merely seeming” rather than real. This leads us to the conclusion that Simpson’s paradox is not really a paradox, according to Rescher’s definition.

We emphasize, however, that although Simpson’s paradox is not really a paradox in a strict technical sense, it is still a surprising kind of pattern. Furthermore, it does look like a paradox for most data mining *users*, who are not trained statisticians or data analysts. Even statisticians who do not consider Simpson’s paradox as a real paradox admit that it is a surprising result for someone who has not studied it before (De Groot & Schervish, 2002).

DISCOVERING INSTANCES OF SIMPSON’S PARADOX IN “FLAT”, SINGLE-RELATION DATA

This section briefly describes an algorithm for discovering instances of Simpson's paradox in "flat" data , stored in a single relation (corresponding to the universal relation of a database system), which is the typical kind of data mined by most data mining algorithms. The algorithm is described in the pseudocode of Algorithm 1 – adapted from (Freitas, 1998). The input for the algorithm is a list L_G of user-defined binary goal attributes, each of them indicating whether or not a given situation of interest (to the user) has occurred. The algorithm then identifies all the binary attributes (potential "first partitioning attributes") in the data and puts them into the list L_1. It also identifies all categorical attributes (potential "second partitioning attributes") in the data and puts them into the list L_2. The algorithm enforces the constraint that any goal attribute included in L_G cannot be included in L_1 nor in L_2.

The output of Algorithm 1 consists of all instances of Simpson's paradox found by the algorithm. These instances can be ranked in decreasing degree of surprisingness to the user, so that a user with little available time can focus on the analyses of only the most surprising instances of the paradox.

```
INPUT: list of user-defined goal attributes, denoted $L_G$
BEGIN
(1)  identify attributes that can be used as 1stPartAtt and put them in list $L_1$
(2)  identify attributes that can be used as 2ndPartAtt and put them in list $L_2$
(3)  FOR EACH goal attribute $G$ in $L_G$
(4)      FOR EACH first partitioning attribute $A$ in $L_1$
(5)          partition population into $Pop_1$ and $Pop_2$ according to values of $A$
(6)          $Pr(G_1) = Pr(G=\text{"yes"}|A=1)$
(7)          $Pr(G_2) = Pr(G=\text{"yes"}|A=2)$
(8)          FOR EACH second partitioning attribute $B$ in $L_2$ such that $B \neq A$
(9)              FOR  $i$=1,2
(10)                 partition $Pop_i$ into $m$ new populations $Pop_{i1}$ ... $Pop_{im}$,
                     according to the values of $B$
(11)                 FOR $j$=1,...,$m$
(12)                     $Pr(G_{ij}) = Pr(G=\text{"yes"}|A=i,B=j)$
(13)             IF ( $Pr(G_1) > Pr(G_2)$  AND  $Pr(G_{1j}) \leq Pr(G_{2j})$, $j$=1,...,$m$ )
                 OR ( $Pr(G_1) < Pr(G_2)$  AND  $Pr(G_{1j}) \geq Pr(G_{2j})$, $j$=1,...,$m$ )
```

```
(14)            OUTPUT the occurrence of the paradox to the user
END
```

Algorithm 1: Search for instances of Simpson’s paradox in flat data

The degree of surprisingness of a given instance of the paradox is estimated by a measure of the “magnitude” of the paradox, as proposed by (Fabris & Freitas, 1999). The basic idea is that the larger the magnitude of the paradox, the more the probability of the situation of interest given the first partitioning attribute differs from the probability of the situation of interest given both the first and the second partitioning attributes, and so the larger the estimated degree of surprisingness for the user.

Formally, the “magnitude” *M* of an instance of the paradox is given by the following formula:

$$M = (M1 + M2) / 2, \qquad (1)$$

where *M1* measures by how much the probability of the situation of interest increases (decreases) from Pop_1 to Pop_2 after the first partition of the data and *M2* measures by how much that probability decreases (increases) from Pop_1 to Pop_2 for each of the categories of *2ndPartAtt* after the second partition of the data.

The measures *M1* and *M2* are as defined in formulas (2) and (3), where *m* is the number of values in the domain of *2ndPartAtt*.

$$M1 = |Pr(G_1) - Pr(G_2)| / \max(Pr(G_1), Pr(G_2)) \qquad (2)$$

$$M2 = \sum_{j=1}^{m} \left(|Pr(G_{1j}) - Pr(G_{2j})| / \max(Pr(G_{1j}), Pr(G_{2j})) \right) / m \qquad (3)$$

In formula (2) the numerator $|Pr(G_1) - Pr(G_2)|$ is simply the absolute value of the difference between the probabilities of the situation of interest in Pop_1 and Pop_2, after the first partitioning of the data. Similarly, in formula (3) the term $|Pr(G_{1j}) - Pr(G_{2j})|$ is the absolute value of the difference between the probabilities of the situation of interest in Pop_{1j} and Pop_{2j}, considering the *j*-th value of *2ndPartAtt*. In both formulas the probability difference terms are divided by the maximum value between the two corresponding probabilities, and in formula (3) the result of this division is averaged over the *m* values of *2ndPartAtt*.

The intuition behind the division of the probability difference terms by their corresponding maximum value is twofold. First, it produces a normalized value between 0 and 1 for both *M1* and *M2*. As a result, the value of *M* in formula (1) is also normalized between 0 and 1, facilitating its interpretation by the user. Second, it takes into account the relative value of the differences between probabilities. To see this point, consider two scenarios in the comparison of two probabilities of the situation of interest. In the first scenario the probabilities are 0.01 and 0.02, whereas in the second scenario the probabilities are 0.49 and 0.50. The absolute value of the difference is 0.01 in both scenarios. However, in the first scenario the relative difference between the two probabilities is larger than in the second scenario – since dividing these differences by the corresponding maximum of the two probabilities we get 0.5 in the first scenario and 0.0002 in the second scenario. Therefore, the first scenario would be associated with a larger magnitude, reflecting the fact that the greater relative difference in the observed probabilities is estimated to be somewhat more surprising to the user.

This kind of normalization –i.e., the division of the probability difference terms by their corresponding maximum value – is particularly important when analyzing data where the situation of interest is a rare event (i.e., it has a very small probability of occurrence) and even a minor probability difference can still be surprising to the user, due to the strategic importance of the rare situation of interest. This is the case, for instance, in the data represented by Table 1, where the number of deaths is a very small fraction of the numbers of individuals in the populations and even a minor difference in the probability of occurrence of this situation of interest is potentially very interesting to the user. In such cases, the just-described normalization is very useful in amplifying the degree of magnitude of the paradox, because without such a normalization the paradox's magnitude would be unduly very low. This normalization avoids this problem by considering the relative (rather than the absolute) value of the probability differences.

To see this point, and also as an example of the use of formulas (1), (2) and (3) to compute the magnitude of an instance of Simpson's paradox, let us consider the paradox instance shown in Table 1, where $Pr(G_1) = 0.0019$, $Pr(G_2) = 0.0022$, $Pr(G_{11}) = 0.0018$, $Pr(G_{21}) = 0.0016$, $Pr(G_{12}) = 0.0056$, $Pr(G_{22}) = 0.0033$ and $m = 2$. Using formulas (2) and (3) we get $M1 = 0.1364$ and $M2 = 0.2609$, and using formula (1) we finally get the measure of magnitude $M = 0.1987$. This indicates that this is a reasonably surprising instance of the paradox. If we did not use that above-described normalization (i.e. if in formulas (2) and (3) the probability differences were not divided by their maximum value), we would get $M1 = 0.0003$, $M2 = 0.00155$ and $M = 0.000775$, which would be an extremely low value for the magnitude of the paradox – an undesirable result.

Finally, it should be noted that the magnitude of an instance of the paradox is just a data-driven, objective estimate of the (ultimately subjective) degree of surprisingness of that pattern to the user. A precise measure of that degree of surprisingness would ideally take into account the user's previous knowledge, but this kind of user-driven, subjective approach would of course considerably reduce the autonomy and generality of the proposed algorithm for detecting instances of Simpson’s paradox. The proposed data-driven approach for estimating the surprisingness of a paradox instance, although not perfect, has the twofold advantage that it is very generic and can be directly applied without requiring the user to spend her/his precious time to specify her/his previous knowledge in the application domain – which would be difficult to formalize, anyway. In the proposed approach the user just has to specify the list of goal attributes with their corresponding situation of interest. This task is much less time consuming and much simpler than capturing and formalizing the user’s previous knowledge.

Analysis of the Computational Time Complexity of the Algorithm

Intuitively, Algorithm 1 seems a time consuming algorithm, since it has several nested loops. In order to formalize this intuition and analyze how the algorithm scales up to large data sets, we now present an analysis of the computational complexity of the algorithm.

The initialization steps of the algorithm – lines (1) and (2) – involve only checking the data type of each attribute, to identify binary and categorical attributes. Assuming this simple metadata information is readily available in the database – a safe assumption in general – each of those two steps takes a time on the order of K, denoted $O(K)$, where K is the number of attributes (or “dimensions”) in the data being mined.

On the other hand, lines (3)–(13) are much more time consuming, involving the computation of conditional probabilities. Lines (3), (4), (8), (9), (11) represent just the beginning of an iteration of a FOR loop. At each iteration of those loops, the corresponding line of course takes only $O(1)$ to initialize or update the value of the loop variable (G, A, B, i, or j, respectively). To be precise, line (8) includes an additional operation, the comparison ($B \neq A$), but this also takes just $O(1)$.

Lines (5), (6) and (7) can be performed in a single scan of all the tuples of the data being mined. This involves updating, on the fly, counters for the number of tuples (records) satisfying each of the conditions ($A = 1$), ($G =$ "*yes*",$A = 1$), ($A = 2$), ($G =$ "*yes*",$A = 2$). These counters are, of course, initialized with 0. Each time a tuple is read, its values for attributes G and A are checked and the appropriate counters are incremented accordingly. For instance, suppose the first tuple has values ($A = 1$) and ($G =$ "*no*"). After reading this tuple, the counter for the condition ($A = 1$) would be incremented to take the value 1, whereas the other three counters would remain with the value 0. After reading all the tuples and performing all the appropriate counter updates, the final values of those counters are directly used to compute the conditional probabilities in steps (6) and (7). Therefore, recalling that the conditional probabilities have to be computed for each iteration of the FOR loops in lines (3) and (4), the computation of lines (5), (6), (7) is performed in time $O(N \times |L_G| \times |L_1|)$, where N is the number of tuples in the data being mined, $|L_G|$ is the cardinality of (number of attributes in) list L_G and $|L_1|$ is the cardinality of list L_1.

The computation of conditional probabilities associated with lines (10) and (12) can be performed in a similar way, with another single scan of all the tuples of the data being mined. The main difference is that now there are more counters to be updated, i.e., the algorithm now needs to keep track of counters for the conditions ($A = 1, B = j$), ($G =$ "*yes*",$A = 1,B = j$), ($A = 2, B = j$), ($G =$ "*yes*",$A = 2,B = j$); for all values j ($j = 1 \ldots m$) of the attribute B, where m is the number of values of *B*. (For now we assume *m* has the same value for all attributes, for the sake of simplicity. We discuss the case where *m* varies across attributes later.) Therefore, recalling that the conditional probabilities of line (12) have to be computed for each iteration of the FOR loops in lines (3), (4), (8), (9) and (11), the computation of lines (10) and (12) is performed in time $O(N \times |L_G| \times |L_1| \times |L_2| \times m)$. Note that the value 2 associated with the two iterations of the FOR loop specified by line (9) – where $i = 1,2$ – does not appear in this expression, since it is a constant.

Finally, lines (13) and (14) involve using the previously computed conditional probabilities to detect instances of the paradox and report them to the user. Line (14) represents only the output of the algorithm, rather than its processing, and therefore can be ignored in the analysis of the computational complexity of the algorithm. Hence, we focus on line (13). This line contains an implicit FOR loop, associated with the expression "$j = 1,\ldots,m$". (This loop was not explicitly indicated in Algorithm 1 for the sake of simplicity in the description of the pseudocode.) Hence, the time taken by each iteration of line (13) is $O(m)$. Recalling that this line has to be computed for each iteration of the FOR loops in lines (3), (4) and (8), the computation of line (13) takes time $O(|L_G| \times |L_1| \times |L_2| \times m)$.

Therefore, the computational time taken by the algorithm as a whole is given by the summation of the following factors:

- $O(K)$ for lines (1), (2);
- $O(N \times |L_G| \times |L_1|)$ for lines (5), (6), (7) of the FOR loops in lines (3), (4);
- $O(N \times |L_G| \times |L_1| \times |L_2| \times m)$ for lines (10), (12) of the FOR loops in lines (3),

  (4), (8), (9), (11);

- $O(|L_G| \times |L_1| \times |L_2| \times m)$ for line (13) of the FOR loops in lines (3), (4), (8).

The second and fourth above factors can be ignored since they are dominated by the third one, so that the total computational time of the algorithm is given by: $O(K + N \times |L_G| \times |L_1| \times |L_2| \times m)$.

In practical applications the value of N, the number of tuples, tends to be much larger than the value of K, the number of categorical attributes in the data, so that this computational complexity can be simplified to:

$$O(N \times |L_G| \times |L_1| \times |L_2| \times m) .$$

It is also useful to discuss how the cardinalities of L_G, L_1, L_2 are related with K. In the worst-case scenario, each of those three lists would have a number of attributes close to K, the maximum possible value for those cardinalities. In this case the algorithm would have a time complexity of $O(N \times K^3 \times m)$. However, this is an unlikely scenario, because the list L_G includes only goal attributes containing a situation of interest to the user and the list L_1 can contain only binary attributes. Therefore, it is quite possible that the cardinalities $|L_G|$ and $|L_1|$ will be considerably smaller than the value of K.

In the best-case scenario where the user specifies a small set of goal attributes and the number of binary attributes is also small, $|L_G|$ and $|L_1|$ would be small constants, and so the above formula for the time complexity of the algorithm would collapse to $O(N \times K \times m)$.

To summarize, with respect to K, the time complexity of Algorithm 1 can vary from linear to cubic, depending on the sizes of $|L_G|$ and $|L_1|$. We stress that the time complexity does not depend on the total number of attributes, but rather just on the number of categorical attributes, K – since continuous attributes are ignored by the algorithm. In any case, the time complexity is linear with respect to *m*, the number of values per categorical attribute, and also linear with respect to *N*, the number of tuples.

Finally, a couple of remarks must be made about the analysis carried out to derive the above formula. First, this analysis was based on a straightforward implementation of the pseudocode described in Algorithm 1. This implementation has the disadvantage that two scans of all the tuples are required, one scan to compute the conditional probabilities of lines (6), (7) and another scan to compute the probabilities of line (12). This can be improved by a somewhat more complex implementation of the algorithm, which performs a single scan of all the tuples, updating all the counters necessary to compute all those probabilities on the fly, as each tuple is read. This reduces the number of scans of all tuples to 1, but it does not change the computational time complexity of the algorithm.

Second, the above analysis assumed that *m*, the number of values of a categorical attribute, is constant for every attribute *B* in list L_2. In practice this is unlikely to be the case. In the most likely scenario of *m* varying across different attributes in L_2, the above formula for computational time complexity can be interpreted in two different ways, depending on the value assigned to *m*. If *m* is assigned the *average* number of values per attribute, computed over all attributes in L_2, the above formula can be interpreted as the average-case time complexity of the algorithm. Alternatively, if *m* is assigned the *largest* number of values per attribute, out of all attributes in L_2, the above formula can be interpreted as the worst-case time complexity of the algorithm.

DISCOVERING INSTANCES OF SIMPSON’S PARADOX IN HIERARCHICAL MULTI-DIMENSIONAL DATA

The discussion of the previous section assumed that all attributes are “flat”, i.e. they have no hierarchy. This is a common situation when mining data extracted from a relational database. In this section, however, we are interested in mining a different kind of data, namely hierarchical multidimensional data (Kimball & Ross, 2002; Thomsen, 2002). The existence of hierarchical dimensions in data cubes introduces new opportunities and requirements for modifying the discovery and the evaluation of the degree of surprisingness of instances of Simpson’s paradox.

Suppose we are analyzing data about sales of a product, involving combinations of values of two attributes, say address, with hierarchy store $\rightarrow$ city $\rightarrow$ state, and time, with hierarchy day $\rightarrow$ month $\rightarrow$ year. Consider an instance of the paradox involving a combination of states and years (highest hierarchical level of both attributes). Should the degree of surprisingness of that instance be computed in the same way as for the combinations of stores and days (lowest hierarchical level of both attributes)? What about combinations of stores and years (mixed hierarchical levels)?

In order to address this kind of question, we modify the original computation of the degree of surprisingness of instances of the paradox (discussed in the previous Section) to take into account a correction factor based on the hierarchical levels of the two attribute values being analyzed.

The first step of this modification consists of deciding whether the algorithm should favor the discovery of paradoxes involving attribute values at higher or lower hierarchical levels of the attributes being analyzed. We have chosen to favor higher hierarchical levels, which is a bias consistent with one of the basic goals of data mining, namely the discovery of comprehensible patterns. In general, the higher the hierarchical level of an attribute, the higher its associated level of abstraction and the smaller the number of values belonging to the domain of the attribute at that level. A smaller number of attribute values facilitates the interpretation of an instance of the paradox by the user, because there are fewer cells in a table representing an instance of the paradox.

An analogy can be made here with the discovery of IF-THEN rules in the classification task of data mining. It is a well-established practice to favor the discovery of shorter rules, having fewer attribute values in their antecedent, because short rules are in general considered more easily interpretable by the user than long rules (Witten & Frank, 2000; Quinlan, 1993).

In other words, both favoring shorter classification rules and favoring paradox instances at higher hierarchical levels aim at the same broad objective, namely reducing the size of the discovered patterns, therefore facilitating their interpretation by the user.

The above mentioned correction factor (hereafter denoted *C*), based on the hierarchical levels of the two partitioning attributes characterizing the paradox, is computed by formula (4):

$$C = \sqrt{\ (\ (nh_{max} - h_{avg})\ /\ h_{max}\ )}, \qquad (4)$$

where $nh_{max} = max(nh_A, nh_B)$ and $h_{avg} = (h_A + h_B) / 2$, where nh_A and nh_B are the numbers of hierarchical levels of attributes *A* (the first partitioning attribute) and *B* (the second partitioning attribute), and h_A and h_B are the indices of the hierarchical levels of attributes *A* and *B* characterizing the paradox. For a given attribute *A* (*B*), its index h_A (h_B) varies from 0 (the root level) to $nh_A - 1$ ($nh_B - 1$).

Finally, the degree of Surprisingness (*S*) of an instance of Simpson's paradox is computed by formula (5):

$$S = M \times C, \quad (5)$$

where *M* is the magnitude of the paradox, computed by formula (1), and *C* is the correction factor taking into account the hierarchical levels of the two partitioning attributes characterizing the paradox, computed by formula (4).

One can see that formula (4) favors the discovery of paradox instances at higher hierarchical levels of attributes *A* and *B* by considering the following example. Suppose that attributes *A* and *B* have respectively 5 and 3 hierarchical levels – i.e., $nh_A = 5$ and $nh_B = 3$ – and that an instance of Simpson's paradox is characterized by the hierarchical level 1 of attribute *A* and by hierarchical level 1 of attribute *B* – i.e., $h_A = 1$ and $h_B = 1$ – i.e, attribute values at a high hierarchical level (close to the root level). Then $C = 0.89$. Now suppose instead that another instance of the paradox is characterized by $h_A = 4$ and $h_B = 2$ – i.e, attribute values at a low hierarchical level. Then $C = 0.63$. In other words, attribute values at lower hierarchical levels are more penalized, since they have a smaller value of the correction factor, and so lead to a smaller value of *S* in formula (5).

Note that formula (4) returns its maximum value, 1, only when the values of attributes *A* and *B* associated with the paradox instances are at the root levels in their hierarchies, i.e., when $h_A = h_B = 0$. This is an intuitive result, since $C = 1$ means no penalty will be applied to attribute values at the root level – which is the "best" hierarchical level in the sense of having the smallest number of attribute values among all hierarchical levels.

In any case, there is nothing magical about formula (4). This formula was chosen in this work because it is a simple formula with the desired effect of favoring paradox instances at higher levels of an attribute hierarchy and it has empirically performed well in our preliminary experiments. However, other formulas that are biased towards favoring higher-level hierarchical values could be used instead. Of course, it would also be easy to use a very different kind of formula implementing a different kind of bias, if the user wished so, since the issue of how the correction factor is computed is orthogonal to the execution of the algorithm for detecting instances of Simpson's paradox.

Note that in formula (5), if we remove the above-discussed correction factor considering hierarchical levels, we obtain the basic equation $S = M$. The motivation for measuring the surprisingness of a paradox instance by its magnitude was explained in section 3. Formula (5) is simply extending that basic equation to the more complex case of hierarchical multidimensional data, introducing the correction factor to favor the discovery of paradox instances involving higher-level attribute values – which tend to be paradox instances more easily interpretable by the user, as discussed earlier.

Our new algorithm for discovering instances of Simpson's paradox in hierarchical multidimensional data is presented in Algorithm 2. Similarly to Algorithm 1, the input for the Algorithm 2 is a list L_G of user-defined binary goal attributes, each of them indicating whether or not a given situation of interest has occurred. There are some basic restrictions that must be put on the creation of the lists L_1 and L_2, as follows.

First, note that list L_1 (the list of candidate *First Partitioning Attributes*) contains pairs of the form $<A, h_A>$, standing for attribute A and hierarchical level h_A of that attribute. A given pair $<A, h_A>$ can be included in list L_1 only if the hierarchical level h_A contains two categorical values, which can then be used to divide the data into two populations. Second, all attributes in L_2 (the list of candidate *Second Partitioning Attributes*) must be categorical, so that they can be used to further divide the data into *m* subpopulations. For each attribute in L_2, all of its hierarchical levels can be used to discover instances of Simpson's paradox. In other words, list L_2 will contain all pairs of the form $<B, h_B>$ where attribute *B* is categorical, regardless of the number of categorical values in hierarchical level h_B, as can be seen in Algorithm 2.

```
INPUT: list of user-defined goal attributes, denoted $L_G$
BEGIN
(1)  identify all pairs <attribute, hierarchical level> that can be used
       as 1stPartAtt and put them in list $L_1$;
(2)  identify attributes that can be used as 2ndPartAtt and put all the
       corresponding pairs <attribute, hierarchical level> in list $L_2$;
(3)  FOR EACH goal attribute $G$ in $L_G$
(4)      FOR EACH pair of attribute $A$ and its corresponding
           hierarchical level $h_A$ in $L_1$:
(5)          partition population into $Pop_1$ and $Pop_2$, according
               to values of $A$ in hierarchical level $h_A$;
(6)          $Pr(G_1) = Pr(G=\text{"yes"}|A=1)$;
(7)          $Pr(G_2) = Pr(G=\text{"yes"}|A=2)$;
(8)          FOR EACH pair of attribute $B$ and its corresponding
               hierarchical level $h_B$ in $L_2$ such that $A \neq B$
(9)          FOR  $i$=1,2
(10)                 partition $Pop_i$ into $m$ new populations
                       $Pop_{i1}$ ... $Pop_{im}$, according to the values
                       of $B$ in hierarchical level $h_B$;
(11)                 FOR $j$=1,...,$m$
(12)                     $Pr(G_{ij}) = Pr(G=\text{"yes"}|A=i,B=j)$;
(13)         IF ($Pr(G_1) > Pr(G_2)$ AND $Pr(G_{1j}) \leq Pr(G_{2j})$, $j$=1,...,$m$)
             OR ($Pr(G_1) < Pr(G_2)$ AND $Pr(G_{1j}) \geq Pr(G_{2j})$, $j$=1,...,$m$ )
(14)             OUTPUT the instance of the paradox to the user;
END
```

Algorithm 2: Discovering instances of Simpson's paradox in hierarchical multidimensional data

Analysis of Computational Time Complexity

The analysis of the computational time complexity of Algorithm 2 is similar to the analysis of Algorithm 1, presented earlier. Hence, in the current subsection we present a relatively summarized version of this kind of analysis, focusing on the parts of the pseudocode in Algorithm 2 that were not present in the Algorithm 1, i.e, the parts referring to the handling of hierarchical dimensions.

Line (1) of Algorithm 2 takes a time on the order of $O(K \times nh_A)$, since the algorithm has to check, for each of the K attributes, whether or not each of its nh_A hierarchical levels contains just two categorical values. Line (2) takes $O(K \times nh_B)$ by a similar argument, since every possible pair of $<B, h_B>$ has to be put in L_2.

Lines (5), (6) and (7) can be performed in a single scan of all the tuples of the data being mined by updating the appropriate counters on the fly, as discussed in the analysis of Algorithm 1. Hence, the time complexity of lines (5), (6), (7) is given by the formula $O(N \times |L_G| \times |HL_1|)$, where N is the number of tuples (records) in the data being mined, $|L_G|$ is the number of *attributes* in list L_G and $|HL_1|$ is the number of *attribute-value pairs* in the hierarchical list L_1.

The computation of conditional probabilities associated with lines (10) and (12) of Algorithm 2 can also be performed using the same approach as described earlier for Algorithm 1. Therefore, the computation of lines (10) and (12) of Algorithm 2 takes time $O(N \times |L_G| \times |HL_1| \times |HL_2| \times m)$, where $|HL_2|$ is the number of *attribute-value pairs* in the hierarchical list L_2 and m is the number of values of attribute B at each hierarchical level. (For now we assume m has the same value for all hierarchical levels, for the sake of simplicity. We discuss the case where m varies across hierarchical levels later.)

The analysis of the time taken by lines 13 and 14 is, again, similar to the analysis of the corresponding lines of Algorithm 1 discussed earlier.1. Hence, the time taken by these lines is $O(|L_G| \times |HL_1| \times |HL_2| \times m)$.

Finally, doing the same simplifications that were done in the analysis of Algorithm 1, the computational time taken by the algorithm as a whole is given by:

$$O(N \times |L_G| \times |HL_1| \times |HL_2| \times m) .$$

Superficially, this computational time complexity is similar to its counterpart for Algorithm 1 derived earlier, viz: $O(N \times |L_G| \times |L_1| \times |L_2| \times m)$. An important difference is that the time complexity of Algorithm 2 involves $|HL_1|$ and $|HL_2|$, rather than $|L_1|$ and $|L_2|$ for Algorithm 1. Hence, it is useful to re-express the former notation in terms of the latter, for a direct comparison between the time complexities of the two algorithms. $|HL_1|$ and $|HL_2|$ can be straightforwardly expressed by $|L_1| \times nh_A$ and $|L_2| \times nh_B$, respectively. Hence, the computational time complexity of Algorithm 2 can be expressed as:

$$O(N \times |L_G| \times |L_1| \times nh_A \times |L_2| \times nh_B \times m)$$

which is clearly considerably higher than the time complexity of Algorithm 1.

In the case of very large data sets, the computational time taken by Algorithm 2 can be significantly reduced by using parallel processing. However, this topic is beyond the scope of this paper, and the interested reader is referred to (Freitas & Lavington, 1998) for a review of parallel data mining techniques.

So far we have assumed that the number of hierarchical levels nh_A is the same for every attribute A in L_1, the number of hierarchical levels nh_B is the same for every attribute B in L_2 and every categorical attribute B has the same number of values, m, in all of its hierarchical levels. In practice, the values of these three variables will vary across attributes. Hence, the previous formula for the time complexity of Algorithm 2 can be interpreted in two different ways. If these three variables are assigned their corresponding *average* value per attribute, the previous formula can be interpreted as the average-case time complexity of the algorithm. Alternatively, if these three variables are assigned their corresponding *largest* value per attribute, the previous formula can be interpreted as the worst-case time complexity of the algorithm.

COMPUTATIONAL RESULTS

Hereafter the algorithm described in the previous section will be called HSPD (Hierarchical Simpson's Paradox Discovery). We have applied the HSPD algorithm to a real-world data cube containing insurance data. The original data was stored in a star-scheme format in disk, containing a fact table with 51332 records and 9 dimension tables. For the purposes of our experiments the data was loaded into main-memory arrays, to make the execution of the algorithm more efficient.

For our experiments, we have manually selected five dimensions, which seem to be the dimensions more promising for the discovery of interesting patterns. The selected dimensions were Claimant, Covered_Item, Event_Time, Insured_Party, and Policy. Figure 1 shows, for each of these dimensions, which were the attributes used in our experiments. Attribute-value hierarchies are indicated by nesting the names of the attribute levels. For instance, the dimension Claimant has attributes Gender, Claimant Type and Age Group with just one hierarchical level each, and an attribute with three hierarchical levels: State $\rightarrow$ Count $\rightarrow$ City. The number of the hierarchical level shown between brackets ranges from 0 for the root (highest) level to *nh* – 1 for the deepest level, where *nh* is the number of levels in the hierarchy of the attribute.

```
CLAIMANT
 ├─ Gender          (level 0)
 ├─ State           (level 0)
 │    └── County          (level 1)
 │              ├── City        (level 2)
 ├─ Claimant type (level 0)
 ├─ Age_Group   (level 0)
 │
COVERED_ITEM
 ├─ Covered_Item_Type    (level 0)
 ├─ Covered_Item_Description    (level 0)
 │
EVENT_TIME
 ├─ Week_Number    (level 0)
 ├─ Year        (level 0)
 │    └── Quarter     (level 1)
 │         ├── Month_Name    (level 2)
 ├─ Week_Day_Name   (level 0)
 ├─ Fiscal_Year      (level 0)
 │      ├── Fiscal_Quarter   (level 1)
 │
INSURED_PARTY
 ├─ Gender     (level 0)
 ├─ State       (level 0)
 │    └── County       (level 1)
 │            ├── City       (level 2)
 ├─ Age_Group     (level 0)
 │
POLICY
 ├─ Risk_Grade (level 0)
```

Figure 1: Description of the dimensions of the data cube used in our experiments

Mining the five dimensions shown in Figure 1, the HSPD algorithm discovered in total 15 instances of Simpson’s paradox, with surprisingness degrees – measured by formula (5) – varying from 0.036 to 0.476. We report the four most surprising instances of the Paradox in Tables 2 through 5. The degree of surprisingness of these instances varies from 0.426 to 0.476.

Table 2: 1stPartAttr = Claimant’s state; 2ndPartAttr = Insured party’s district
Situation of Interest: (Claimant’s type = third party)
Surprisingness degree: 0.461

Claimant' state = PA				Claimant' state = DE			
Insured party's district	total pop.	claimant = third party	%	Insured party's district	total pop.	claimant = third party	%
Bucks	9186	2532	27.6	Bucks	0	0	0
Philadel.	12476	3948	31.6	Philadel.	0	0	0
NewCast.	42	42	100.0	NewCast.	3954	1320	33.4
Montgom	11564	3456	29.9	Montgom.	0	0	0
Delaware	7458	2464	33.0	Delaware	0	0	0
Chester	6652	2328	35.0	Chester	0	0	0
TOTAL	47378	17770	31.2	TOTAL	3954	1320	33.4

Table 3: 1stPartAttr=Insured party's gender; 2ndPartAttr=Claimant's gender
Situation of Interest: (Insured party's state = DE)
Surprisingness degree: 0.476

Insured party's gender = female				Insured party's gender = male			
Claimant's gender	total pop.	Insured party's state= DE	%	Claimant's gender	total pop.	Insured party's state= DE	%
female	24774	2150	8.7	female	234	42	17.9
male	84	0	0	male	26240	1804	6.9
TOTAL	24858	2150	8.6	TOTAL	26474	1846	7.0

Table 4: 1stPartAttr = Claimant's state; 2ndPartAttr = Insured party's state
Situation of Interest: (Claimant's type = third party)
Surprisingness degree: 0.450

Claimant' state = PA				Claimant' state = DE			
Insured party's state	total population	claimant = third party	%	Insured party's state	total population	claimant = third party	%
PA	47336	14728	31.1	PA	0	0	0
DE	42	42	100.0	DE	3954	1320	33.4
TOTAL	47378	14770	31.2	TOTAL	3954	1320	33.4

Table 5: 1stPartAttt = Insured party's state; 2ndPartAttr = Claimant's state
Situation of Interest: (Insured party's sex = male)
Surprisingness degree: 0.426

Insured party's state = PA				Insured party's state = DE			
Claimant's state	total pop.	Insured party's sex = male	%	Claimant's state	total pop.	Insured party's sex = male	%
PA	47336	24628	52.0	PA	42	42	100
DE	0	0	0.0	DE	3954	1804	45.6
TOTAL	47336	24628	52.0	TOTAL	3996	1846	46.2

In the title of each table we indicate the first partitioning attribute, the second partitioning attribute, the situation of interest and the surprisingness degree of the corresponding instance of the paradox. In addition, these tables have the following structure. The first row indicates the two values of the *First Partitioning Attribute* used to partition the data into subpopulations Pop_1 and Pop_2. Those two values divide the table into two parts. Each of those parts is further divided into four columns, whose meaning is as follows. The first column indicates the values of the *Second Partitioning Attribute*. In this column each row, starting from the third row of the table, specifies one value of that attribute. In the second column each cell contains the number of records having the corresponding values of the *First* and *Second Partitioning Attributes* used to specify the position of the cell. In the third column each cell contains the number of records that not only have the corresponding values of *First* and *Second Partitioning Attributes* but also have the situation of interest (value of the goal attribute) indicated in the header of this column (in the second row of the table). Finally, in the fourth column each cell contains the percentage of records with the situation of interest for the corresponding values of *First* and *Second Partitioning Attributes*.

As shown in Tables 2 through 5, all the four most surprising discovered instances of the paradox involved a combination of partitioning attributes of two dimensions, namely Claimant and Insured Party. In addition, in three out of those four tables, the partitioning attributes were the addresses of the Claimant and the Insured Party, indicating that there is a surprising relationship between the addresses of these two agents. This surprising relationship holds for two different situations of interest, namely “Claimant type = third party” (Tables 2 and 4), and “Insured party’s sex = male” (Table 5). In Tables 2, 4 and 5, in general the instances of the paradox are associated with the addresses of Claimant and Insured Party at the hierarchical level of state. (The exception is that in Table 2 the address of the Insured Party is at the hierarchical level of district.)

Note that the fact that instances of Simpson's paradox were discovered from hierarchical multidimensional data offers the user a possibility that is not available when instances of the paradox are discovered from a single "flat" relation. In the former case, the user can use conventional OLAP operators, such as drill-down, to further analyze the discovered instances of the paradox, observing the data at a lower level of abstraction (a deeper hierarchical level). Actually, the result of a drill-down on an instance of the paradox might even have been already reported to the user, if the paradox also occurred in the data associated with the drill-down. For instance, Table 2 is actually a drill-down of Table 4. Looking at Table 4, one can see that in general the claimant's state is the same as the insured party's state. However, there are some exceptions. More precisely, there are 42 cases where the claimant's state is PA but the insured party's state is DE. Observing the drilled-down data in Table 2 one can see that all those 42 cases occur when the insured party's district is New Castle.

Hence, instances of the paradox discovered in hierarchical multidimensional data not only represent surprising patterns by themselves, but also have the nice "side effect" of naturally suggesting potentially-interesting drill-down directions for the user – therefore, in some sense, increasing the functionality of OLAP tools.

CONCLUSIONS AND FUTURE RESEARCH

Previous work in the literature introduced an algorithm that discovers surprising instances of Simpson’s paradox in data based on the relational model, assuming that all the data was stored in a single universal relation. In this paper we have extended that algorithm to cope with hierarchical multidimensional data, stored in a star scheme. Hence, this work obtains an *integration between data mining and data warehouse/OLAP*, which is beneficial for both areas.

We emphasize that the algorithm proposed in this paper was designed specifically for discovering surprising patterns. By contrast, a number of data mining algorithms in the literature were designed for initially discovering a large number of patterns and then passing them through a filter, to try to select the most surprising (or interesting) patterns. It is also important to notice that many measures of “interestingness” proposed in the literature focus on measuring some kind of statistical correlation or another predictive accuracy-related criterion, without actually trying to estimate the *degree of surprisingness of discovered patterns to the user*. This is the case, for instance, with the 21 measures of rule interestingness discussed by Tan et al. (2002). There are, of course, several measures of rule surprisingness (mentioned in the Introduction), but this work focuses on the discovery of a very different kind of surprising pattern, as explained in the Introduction.

We believe that Simpson’s paradox offers good opportunities for future research in data mining, since the usefulness of discovering Simpson’s paradox instances has been underexplored in the literature. Discovered instances of the paradox are potentially useful for helping to solve other kinds of data mining problems.

As one example, we can envisage the following application of Simpson’s paradox discovery in prediction-rule discovery. Consider that hidden instances of Simpson’s paradox can fool a greedy data mining algorithm, making it to misinterpret a given relationship between some attributes (Glymour et al., 1997). E.g., greedy decision tree induction and rule induction algorithms can select an attribute or attribute value that seems to have a certain relationship with a given class, when in reality the true relationship – taking into account attribute interactions (Freitas, 2001) – is the reverse of the apparent one. If instances of Simpson’s paradox have been previously discovered, in principle the decision tree or rule induction algorithm could be given information about those discovered instances of the paradox, in the form of “background knowledge”. Once the algorithm has been properly modified to take into account this kind of background knowledge, it would not be fooled by those instances of the paradox – i.e., it would not choose the wrong attribute or attribute value to add to a prediction rule, because it would know that the true relationship is the reverse of the apparent one. This seems an interesting research direction.

Another research direction consists of devising more efficient algorithms for discovering Simpson’s paradox instances, perhaps by exploiting background knowledge to reduce the size of the search space.

ACKNOWLEDGMENT

We are very grateful to Kurt E. Allebach for making available to us the hierarchical multi-dimensional data set used in our experiments. We also thank the anonymous reviewers for their valuable comments, which significantly contributed to the final, improved version of this paper.

REFERENCES

Carvalho, D.R., Freitas, A.A. and Ebecken, N.F.F. (2003). A critical review of rule surprisingness measures. In: N.F.F. Ebecken, C.A. Brebbia, A. Zanasi (Eds.) *Proceedings of Data Mining IV (4th International Conference on Data Mining)*, pp. 545-555. Southampton, UK: WIT Press.

De Groot, M.H. and Schervish, M.J. (2002). *Probability and Statistics*. 3rd Ed. New York: Addison-Wesley.

Dong, G. and Li, J. (1998). Interestingness of discovered association rules in terms of neighborhood-based unexpectedness. *Research and Development in Knowledge Discovery & Data Mining (Proceedings of the 2nd Pacific-Asian Conference, PAKDD-98). Lecture Notes in Artificial Intelligence 1394*, pp. 72-86. Berlin: Springer-Verlag.

Fabris, C.C. and Freitas, A.A (1999). Discovering surprising patterns by detecting instances of Simpson's paradox. In: M. Bramer, A. Macintosh, and F. Coenen (Eds.) *Research and Development in Intelligent Systems XVI (Proceedings of ES-99, the 19th International Conference on Knowledge Based Systems and Applied Artificial Intelligence)*, pp. 148-160. Berlin: Springer-Verlag.

Fayyad, U.M., Piatetsky-Shapiro, G. and Smyth, P. (1996). From data mining to knowledge discovery: an overview. In: Fayyad, U.M. et al (Eds.) *Advances in Knowledge Discovery and Data Mining*, pp. 1-34. Menlo Park, CA, USA: AAAI/MIT.

Freitas, A.A. (1998). On objective measures of rule surprisingness. *Principles of Data Mining and Knowledge Discovery: Proceedings of the 2nd European Symposium (PKDD'98). Lecture Notes in Artificial Intelligence 1510*, pp. 1-9. Berlin: Springer-Verlag.

Freitas, A.A. (2001). Understanding the crucial role of attribute interaction in data mining. *Artificial Intelligence Review 16(3), pp. 177-199*.

Freitas, A.A. and Lavington, S.H. (1998). *Mining Very Large Databases with Parallel Processing*. Amsterdam: Kluwer.

Glymour, C., Madigan, D., Pregibon, D. and Smyth, P. (1997). Statistical themes and lessons for data mining. *Data Mining and Knowledge Discovery 1(1)*, pp. 11-28.

Kimball, R. and Ross, M. (2002). *The Datawarehouse Toolkit: the complete guide to multidimensional modeling*. 2nd Ed. New York: John Wiley & Sons.

Liu, B. and Hsu, W. (1996). Post-analysis of learned rules. *Proceedings of the 1996 National Conference of the American Association for Artificial Intelligence (AAAI-96)*, pp. 828-834. Menlo Park, CA, USA: AAAI Press.

Liu, B., Hsu, W. and Chen, S. (1997). Using general impressions to analyze discovered classification rules. *Proceedings of the 3rd International Conference on Knowledge Discovery & Data Mining*, pp. 31-36. Menlo Park, CA, USA: AAAI Press.

Matheus, C.J.; Piatetsky-Shapiro, G. and McNeil, D. (1996). Selecting and reporting what is interesting: the KEFIR application to health data. In: Fayyad, U.M. et al. (Eds.) *Advances in Knowledge Discovery and Data Mining*, pp. 495-516. Menlo Park, CA, USA: AAAI/MIT Press.

Newson, G. (1991). Simpson´s paradox revisited. *The Mathematical Gazette 75(473)*, pp. 290-293.

Ohsaki, M., Kitaguchi, S., Okamoto, K., Yokoi, H., and Yamaguchi, T. (2004). Evaluation of rule interestingness measures with a clinical dataset on hepatitis. *Knowledge Discovery in Databases: Proceedings of PKDD-2004, Lecture Notes in Artificial Intelligence 3202*, pp. 362-373. Berlin: Springer-Verlag.

Padmanabhan, B. and Tuzhilin, A. (1998) A belief-driven method for discovering unexpected patterns. *Proceedings of the 4th International Conference on Knowledge Discovery & Data Mining (KDD-98)*, pp. 94-100. Menlo Park, CA, USA: AAAI Press.

Quinlan, J.R. (1993). *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann.

Rescher, N. (2001). *Paradoxes: their roots, range and resolution*. New York: Open Court.

Romao, W., Freitas, A.A. and Gimenes, I.M.S. (2004). Discovering interesting knowledge from a science and technology database with a genetic algorithm. *Applied Soft Computing 4(2)*, pp. 121-137.

Sahar, S. (2002). On incorporating subjective interestingness into the mining process. *Proc. 2002 IEEE International Conference on Data Mining*. New York: IEEE Press.

Silberschatz, A. and Tuzhilin, A. (1996). What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge & Data Engineering*, 8(6), pp. 970-974.

Simpson, E.H. (1951). The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society, Series B, 13,* pp. 238-241.

Suzuki, E. (1997). Autonomous discovery of reliable exception rules. *Proceedings of the 3rd International Conference on Knowledge Discovery & Data Mining*, pp. 259-262. Menlo Park, CA, USA: AAAI Press.

Suzuki, E. and Kodratoff, Y. (1998). Discovery of surprising exception rules based on intensity of implication. *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98)*. *Lecture Notes in Artificial Intelligence, 1510*, pp. 10-18. Berlin: Springer-Verlag.

Tan, P.N., Kumar, V. and Srivastava, J. (2002). Selecting the right interestingness measure for association patterns. *Proceedings of the ACM SIGKDD 2002 International Conference on Knowledge Discovery and Data Mining (KDD-2002)*. New York: ACM Press.

Thomsen, E. (2002). *OLAP Solutions : building multi-dimensional systems,* 2nd Ed. New York: Wiley.

Wagner, C.H. (1982). Simpson's paradox in real life. *The American Statistician*, *36(1),* pp. 46-48.

Witten, I.H. and Frank, E. (2000). *Data Mining: practical machine learning tools with Java implementations*. San Francisco, CA, USA: Morgan Kaufmann.