Efficient Rule-Based Attribute-Oriented Induction for Data Mining *

David W. Cheung[†] **H.Y. Hwang**[‡] **Ada W. Fu**[‡] **Jiawei Han**[§]

[†] Department of Computer Science and Information Systems, The University of Hong Kong, Hong Kong.
Email: dcheung@csis.hku.hk.

[‡] Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong.
Email: adafu@cs.cuhk.hk.

[§] School of Computing Science, Simon Fraser University, Canada. Email: han@cs.sfu.ca.

Abstract. *Data mining has become an important technique which has tremendous potential in many commercial and industrial applications. Attribute-oriented induction is a powerful mining technique and has been successfully implemented in the data mining system DBMiner [17]. However, its induction capability is limited by the unconditional concept generalization. In this paper, we extend the concept generalization to rule-based concept hierarchy, which enhances greatly its induction power. When previously proposed induction algorithm is applied to the more general rule-based case, a problem of induction anomaly occurs which impacts its efficiency. We have developed an efficient algorithm to facilitate induction on the rule-based case which can avoid the anomaly. Performance studies have shown that the algorithm is superior than a previously proposed algorithm based on backtracking.*

Keywords: data mining, knowledge discovery in databases, rule-based concept generalization, rule-based concept hierarchy, attribute-oriented induction, inductive learning, learning and adaptive systems.

1 Introduction

Data mining (also known as *Knowledge Discovery in Databases*) is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data [12]. Over the past twenty years, huge amounts of data have been collected and managed in relational databases by industrial, commercial or public organizations. The growth in size and number of existing databases has far exceeded the human abilities to analyze such data with available technologies. This has created a need and a challenge for extracting knowledge from these databases. Many large corporations are investing into the data mining tools, and in many cases, they are coupled with the *data warehousing* technology to become an integrated system to support management decision and business planning [27]. Within the research community, this problem of *data mining* has been touted as one of the many great challenges [10, 12, 25]. Researches have been performed with different approaches to tackle this problem [2, 3, 4, 8, 9, 15, 16, 18, 19, 21, 28].

*The research of the authors were supported in part by RGC (the Hong Kong Research Grants Council) grant HKU 286/95E. Research of the fourth author was supported in part by grants from the Natural Sciences and Engineering Research Council of Canada the Centre for Systems Science of Simon Fraser University.

In the previous studies [16], a **Basic Attribute-Oriented Induction** (**Basic AO Induction**) method has been developed for knowledge discovery in relational databases. AO induction can discover many different types of rules. A representative type is the characteristic rule. For example, from a database of computer science students, it can discover a characteristic rule such as "if x is a computer science student, then there is a 45% chance that he is a foreign student and his GPA is excellent". Note that the concepts of "foreign student" and "excellent GPA" do not exist in the database. Instead, only lower level information such as "birthplace" and GPA value are stored there. An important feature of AO induction is that it can generalize the values of the tuples in a relation to higher level concepts, and subsequently merge those tuples that have become identical into generalized tuples. An important observation is that each one of these resulted generalized tuples certainly reflects some common characteristics of the group of tuples in the original relation from which it is generated.

Basic AO induction has been implemented in the mining system **DBMiner**, (whose prototype was called **DBLearn**) [16, 17, 19]. Besides AO induction, DBMiner also has incorporated many interesting mining techniques including mining multiple-level knowledge, meta-rule guided mining, and can discover many different types of rules, patterns, trends and deviations [17]. AO induction, not only can be applied to relation databases, but also performed on unconventional databases such as spatial, object-oriented, and deductive databases [19].

The engine for concept generalization in basic AO induction is the *concept ascension.* It relies on a *concept tree* or *lattice* to represent the background knowledge for generalization. [22]. However, concept tree and lattice have their limitation in terms of background knowledge representation. In order to further enhance the capability of AO induction, there is a need to replace them by a more general concept hierarchy. In this paper, the work has been focused on the development of a *rule-based concept hierarchy* to support a more general concept generalization. Rule-based concept generalization was first studied in [7]. In general, concepts in a concept tree or lattice are generalized to higher level concepts *unconditionally.* For example, in a concept tree defined for the attribute *GPA* of a student database, a *3.6 GPA* (in a 4 points system) can be generalized to a higher level concept, perhaps to the concept *excellent.* This generalization depends only on the GPA value but not any other information (or attribute) of a student. However, in some institutions, they may want to apply different rules to different types of students. The same *3.6 GPA* may only deserve a *good*, *if the student is a graduate*; and it may be *excellent*, *if the student is an undergraduate.* This suggests that a more general concept generalization scheme should be *conditional* or *rule-based.*

In a **Rule-Based Concept Graph**, a concept can be generalized to more than one higher level concept, and rules are used to determine which generalization path should be taken. To support AO induction on a rule-based concept graph, we have extended the basic AO induction to **Rule-Based Attribute-Oriented Induction**. However, if the technique of induction on a concept tree is applied directly to a concept graph, a problem of **induction anomaly** would occur. In [7], a "backtracking" technique was proposed to solve this problem. It is designed based on *generalized relation* which is proposed originally in the basic AO induction. The backtracking algorithm has an $O(n \log n)$ complexity, where n is the number of tuples in the induction domain. In [23], a more efficient technique for induction has been proposed. In this paper, we apply the technique to rule-based concept graph, and propose an algorithm for rule-based induction whose complexity is improved to $O(n)$. The algorithm has avoided to use the data structure of generalized relation. Instead, it uses a **multi-dimensional data cube** [20] or a **generalized-attribute tree** depending on the sparseness of the data distribution. Extensive performance studies on the algorithm have been done and the results show that it is more efficient than the backtracking algorithm based on generalization relation.

$\{0.0 - 1.99\} \rightarrow poor$
$\{2.0 - 2.99\} \rightarrow average$
$\{3.0 - 3.49\} \rightarrow good$
$\{3.5 - 4.0\} \rightarrow excellent$
$\{poor, average\} \rightarrow weak$
$\{good, excellent\} \rightarrow strong$
$\{weak, strong\} \rightarrow ANY(GPA)$

Figure 1: Concept tree table entries for a university student database

The paper is organized as follows. The primitives of knowledge discovery and the principle of basic AO induction are briefly reviewed in Section 2. The general notions of rule-based concept generalization and rule-based concept hierarchy are discussed in Section 3. The model of rule-based AO induction is defined in Section 4. A new technique of using path relation and data cube instead of generalized relation to facilitate rule-based AO induction is discussed in Section 5. An efficient rule-based AO induction algorithm is presented in Section 6. Section 7 is the performance studies. Discussions and conclusions are in Sections 8 and 9.

2 Basic Attribute-Oriented Induction

The purpose of AO induction is to discover rules from relations. The primary technique used is concept generalization. In a database such as a university student relation with the schema

$$Student(Name, Status, Sex, Major, Age, Birthplace, GPA),$$

the values of attributes like Status, Age, Birthplace, GPA can be generalized according to a concept hierarchy. For example, GPA between 0.0 and 1.99 can be generalized to "poor", those between 2.0 and 2.99 to "average", other values to "good" or "excellent". After this process, many records would have the same values except on those un-generalizable attributes such as Name. By merging the records which have the same generalized values, important characteristics of the data can be captured in the generalized tuples and rules can be generated from them. Basic AO induction is proposed along this approach. *Task-relevant data, background knowledge*, and *expected representation of learning results* are the three primitives that specify a learning task in basic AO induction [16, 19].

2.1 Primitives in AO Induction

The first primitive is the **task-relevant data**. A database usually stores a large amount of data, of which only a portion may be relevant to a specific induction task. A query that specifies an induction task can be used to collect the task-relevant set of data from a database as the domain of an induction. In AO induction, the retrieved task-relevant tuples are stored in a table called **initial relation**.

The second primitive is the **background knowledge**. In AO induction, background knowledge is necessary to support generalization, and it is represented by concept hierarchies. Concept hierarchies could be supplied by domain experts. As have been pointed out, generalization is the key engine in induction.

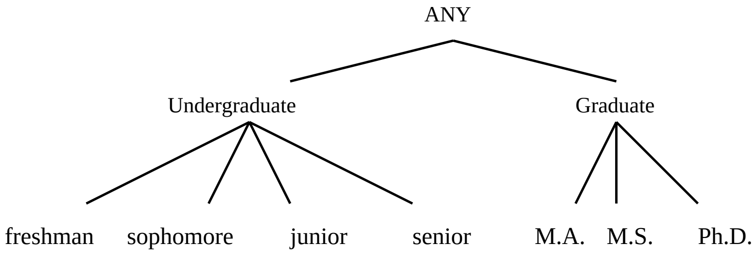

Figure 2: A concept tree for *Status*

Therefore, the structure and representation power of the concept hierarchies is an important issue in AO induction.

Concepts hierarchies in a particular domain are often organized as a multi-level taxonomy in which concepts are partially ordered according to a general-to-specific ordering. The most general concept is the null description (described by a reserved word "ANY"), and the most specific concepts corresponds to the low level data values in the database [22]. The simplest hierarchy is **concept tree** in which a node can only be generalized to one higher level node at each step.

Example 1 Consider a a typical university student database with a schema

$$Student(Name, Status, Sex, Major, Age, Birthplace, GPA),$$

Part of the corresponding *concept tree table* is shown in Figure 5, where $A \rightarrow B$ indicates that B is a *generalization* of the members of A.

An example of a concept tree on the attribute *status* is shown in Figure 6. □

Student records in the university relation can be generalized following the paths in the above concept trees. For example, the status value of all students can be generalized to "undergraduate" or "graduate".

Note that the generalization should be performed iterative from lower levels to higher levels together with the merging of tuples which have the same generalized values. The generalization should stop once the generalized tuples resulted from the merging have reached a reasonable level in the concept trees. Otherwise, the resulted tuples could be over generalized and the rules generated subsequently would have no practical usage.

The third primitive is the **representation of learning results**. The generalized tuples at the end will be used to generate rules by converting them to logic formula. This follows the fact that a tuple in a relation can always be viewed as a logic formula in conjunctive normal form, and a relation can be characterized by a large set of disjunctions of such conjunctive forms [13, 26]. Thus, both the data for learning and the rules discovered can be represented in either relational form (tuples) or first-order predicate calculus. For example, if one of the generalized tuples resulted from the generalization and merging of the computer science student records in the university database is (graduate(Status), NorthAmerican(Birthplace), good(GPA)), then the following rule in predicate calculus can be generated:

$$cs\_student(x) \rightarrow$$
$$(Birthplace(x) \in NorthAmerica \wedge Status(x) \in graduate \wedge GPA(x) \in good).$$

Note that the above rule is only one of the many rules that would be generated, and it is not quantified yet. We will explain how quantification of the rules is done in AO induction. Many kinds of rules, such as *characteristic rules*, and *discriminant rules* can be discovered by induction processes [15]. A **characteristic rule** is an assertion that characterizes a concept satisfied by all or most of the examples in the class targeted by a learning process. For example, the symptoms of a specific disease can be summarized by a characteristic rule. A **discriminant rule** is an assertion which discriminates a concept of the class being learned from other classes. For example, to distinguish one disease from others, a discriminant rule should summarize the symptoms that discriminate this disease from others.

2.2 Concept Generalization

The most important mechanisms in AO Induction are concept generalization and rule creation. Generalization is performed on all the tuples in the initial relation with respect to the concept hierarchy. All values of an attribute in the relation are generalized to the same higher level. A selected set of attributes of the tuples in the relation are generalized to possibly different higher levels synchronously and redundant tuples are merged to become generalized tuples. The resulted relation containing these generalized tuples is called a **generalized relation**, and it is smaller than the initial relation. In other words, a generalized relation is a relation which consists of a set of generalized attributes storing generalized values of the corresponding attributes in the original relation.

Even though, a generalized relation is smaller than the initial relation; however it may still contain too many tuples, and is not practical to convert them to rules. Therefore, some principles are required to guide the generalization to do further reduction. An attribute in a generalized relation is at a **desirable level** if it contains only a small number of distinct values in the relation. A user of the mining system can specify a small integer as a **desirable attribute threshold** to control the number of distinct values of an attribute. An attribute is at the **minimum desirable level** if it would contain more distinct values than a defined desirable attribute threshold when generalized to a level lower than the current one [16]. The minimum desirable level for an attribute can also be specified explicitly by users or experts.

A special generalized relation R' of an original relation R is the **prime relation** [19] of R if every attribute in R' is at the minimum desirable level. The first step of AO induction is to generalize the tuples in the initial relation to proper concept levels such that the resulted relation becomes the prime relation. The prime relation has a useful characteristic of containing minimal distinct values for each attribute. However, it may still has many tuples, and would not be suitable for rule generation. Therefore, AO induction will generalize and reduce the prime relation further until the final relation can satisfy the user's expectation in terms of rule generation. This generalization can be done repetitively in order to generate rules at different concept levels so that a user can find out the most suitable levels and rules. This is the technique of **progressive generalization (roll-up)** [19]. If the rules discovered in a level is found to be too general, then re-generalization to some lower levels can be performed and this technique is called **progressive specialization (drill-down)** [19]. DBMiner has implemented the roll-up and drill-down techniques to support user to explore different generalization paths until the resulted relation and rules so created can satisfy his expectation.

A discovery system can also quantify a rule generated from a generalized tuple by registering the number of tuples from the initial relation, which are generalized to the generalized tuple, as a special attribute *count* in the final relation. The attribute *count* carries database statistics to higher level concept rules, supports pruning scattered data and searching for substantially weighted rules. A set of basic principles for AO induction related to the above discussion has been proposed in [15, 16].

3 Rule-Based Concept Generalization

In Basic AO Induction, the key component to facilitate concept generalization is the concept tree. Its generalization is *unconditional* and has limited generality. From this point on, we will focus our investigation on the **rule-based concept generalization** which is a more general scheme. Concepts are partially ordered in a concept hierarchy by **levels** from specific (lower level) to general (higher level). Generalization is achieved by ascending concepts along the paths of a concept hierarchy. In general, a concept can ascend via more than one path. A **generalization rule** can be assigned to a path in a concept hierarchy to determine whether a concept can be generalized along that path.

For example, the generalization of GPA in Figure 5 could depend not only on the GPA of a student but also his *status*. A GPA could be a *good* GPA for an *undergrad* but a *poor* one for a *graduate*. For example, the rules to categorize a GPA in (2.0 - 2.49) may be defined by the following two *conditional* generalization rules:

If a student's GPA is in the range (2.0 - 2.49) and he is an undergrad, then it is an average GPA.

If a student's GPA is in the range (2.0 - 2.49) and he is a grad, then it is a poor GPA.

Concept hierarchies whose paths have associated generalization rules are called **rule-based concept hierarchies**. Concept hierarchies can be balanced or unbalanced. Unbalanced hierarchy can always be converted to a balanced one. For easy of discussion, we will assume all hierarchies are balanced. Also, similar to concept tree, we will assume that the concepts in a hierarchy are partially ordered into levels such that lower level concepts will be generalized to the next higher level concepts and the concepts converge to the null concept "ANY" in the top level (root). (The two notions of "concept" on the concept hierarchy and "generalized attribute value" are equivalent, depending on the context, we will use the two notions interchangeably.) In the following, three types of concept generalization, their corresponding generalization rules and concept hierarchies are classified and discussed.

3.1 Unconditional Concept Generalization

This is the simplest type of concept generalization. The rules associated with these hierarchies are the unconditional IS-A type rules. A concept is generalized to a higher level concept because of the subsumption relationship indicated in the concept hierarchy. This type of hierarchies support concept climbing generalization. The most popular unconditional concept generalization are performed on *concept tree* and *lattice*. The hierarchies represented in both Figure 5 and Figure 6 belong to this type.

3.2 Deductive Rule Generalization

In this type of generalization, the rule associated with a generalization path is a deduction rule. For example, the deduction rule: *if a student's GPA is in the range (2.0 - 2.49) and he is a grad, then it is a poor GPA,* can be associated with the path from the concept $GPA \in (2.0 - 2.49)$ to the concept *poor* in the GPA hierarchy.

This type of rules is *conditional* and can only be applied to generalize a concept if the corresponding condition can be satisfied. A deduction generalization rule has the following form:

$A(x) \wedge B(x) \longrightarrow C(x)$.

For a tuple x, concept (attribute value) A can be generalized to concept C if condition B can be satisfied by x.

The condition $B(x)$ can be a simple predicate or a general logic formula. In the simplest case, it can be a predicate involving a single attribute. A concept hierarchy associated with deduction generalization rules is called **deduction-rule-based concept graph**. This structure is suitable for induction in a database that supports deduction.

3.3 Computational Rule Generalization

The rules for this type of generalization are computational rules. Each rule is represented by a condition which is value-based and can be evaluated against an attribute or a tuple or the database by performing some computation. The truth value of the condition would then determine whether a concept can be generalized via the path.

For example, in the concept hierarchy for a spatial database, there may be three generalization paths from *regional_spatial_data* to the concepts of *small_region*, *medium_size_region* and *large_region*. Conditions like "$region\_size \leq SMALL\_REGION\_SIZE$", "$region\_size > SMALL\_REGION\_SIZE \wedge region\_size < LARGE\_REGION\_SIZE$", and "$region\_size \geq LARGE\_REGION\_SIZE$" can be assigned to these paths respectively. The conditions depend on the computation of the value of *region_size* from the *regional_spatial_data*. In general, computation rules may involve sophisticated algorithms or methods which are difficult to be represented as deduction rules.

The hierarchy with associated computation rules is called **computation-based concept graph**. This type of hierarchy is suitable for induction in databases that involve a lot of numerical data, e.g., spatial databases and statistical databases.

3.4 Hybrid Rule-Based Concept Generalization

A hierarchy can have paths associated with all the above three different types of rules. This type of hierarchy is called **hybrid rule-based concept graph**. It has a powerful representation capability and is suitable for many kinds of applications. For example, in many spatial databases, some generalization paths are computation bound and are controlled by computation rules. While some symbolic attributes can be generalized by deduction rules. And many simple attributes can be generalized by unconditional IS-A rules.

In the scope of database induction, the same technique can be used on all the different types of rule-based hierarchies. Therefore, in the rest of this paper, we will use deduction-rule-based concept graph as a typical concept hierarchy.

4 Rule-Based Attribute-Oriented Induction

In order to discuss the technique of AO induction in the rule-based case. We first define a general model for rule-based AO induction. A rule-based AO induction system is defined by five components (DB, CH, DS, KR, t_a). DB is the underlying extensional database. CH is a set of rule-based concept hierarchies associated with the attributes in DB. We assume these hierarchies are deduction-rule-based concept graphs. DS is a deduction system supporting the concept generalization. The generalization rules in CH, together with some other deduction rules form the core of DS. In the simple case, DS may consist of only the rules in CH. KR is a knowledge representation scheme for the learned result. It can be any one of the popular schemes : predicate calculus, frames, semantic nets, production rules, etc. Following the approach in basic AO induction, we assume KR in the induction system is first-order predicate calculus. The last component t_a is the *desirable attribute threshold* defined in the basic induction. Note that all these five components are input to the rule-based deduction system. Output of the system is the rules discovered from the database.

The generalization and rule creation processes in rule-based induction is fundamentally the same as that in the basic induction. However, an attribute value could be generalized to different higher level concepts depending on the concept graph. As a consequence, the techniques in basic induction have to be modified to solve the induction problem in this case. We will describe in this section the frame work of the rule-based induction, and explain the induction anomaly problem which occurs in this case.

Same as the basic induction, the first step of rule-based induction is to generalize and reduce the initial relation to the prime relation. The minimum desirable level can be found in a scan of the initial relation. Once the minimum desirable levels have been determined, the initial relation can be generalized to these levels in a second scan, and the result would be the prime relation. This step of inducing the prime relation is basically the same as that of the basic induction, except that the induction is performed on a general concept graph rather than a restricted concept tree.

Once the prime relation is found, selected attributes are to be generalized further and the generalization-comparison-merge process will be repeated to perform roll-up or drill-down. (Some selected attributes can also be removed before the generalization starts.) In the basic induction, attributes in a prime relation can always be generalized further by concept tree ascension because the generalization is based only on the current generalized attribute values. However, this may not be the case in the rule-based induction because the application of a rule on the prime relation may require additional information not available in the prime relation. This phenomenon is called induction anomaly. Following are some cases that will cause this anomaly to happen.

(1) A rule may depend on an attribute which has been removed.

(2) A rule may depend on an attribute whose concept level in the prime relation has been generalized too high to match the condition of the rule.

(3) A rule may depend on a condition which can only be evaluated against the initial relation, e.g., the number-of-tuples in the relation.

In the following, an example will be used to illustrate the rule-based induction on a concept graph and the associated induction anomaly.

Example 2 This example is based on the induction in Example 1. We will enhance the concept tree there to a rule-based concept graph to explain the rule-based induction.

The database DB is the same student database in Example 1, and the mining task is the same which is to discover the characteristic rules for CS students. The modification is in the concept hierarchy CH. The unconditional rules for the attribute GPA in Figure 5 are replaced by the set of deduction rules in Figure 7. And the concept tree for GPA is enhanced to the rule-based concept graph in Figure 8 which has been labeled with the corresponding deduction rules in Figure 7. For example, the GPA in the range $(3.5 - 3.79)$

$R_1 : \{0.0 - 1.99\} \rightarrow poor$
$R_2 : \{2.0 - 2.49\} \wedge \{graduate\} \rightarrow poor$
$R_3 : \{2.0 - 2.49\} \wedge \{undergraduate\} \rightarrow average$
$R_4 : \{2.5 - 2.99\} \rightarrow average$
$R_5 : \{3.0 - 3.49\} \rightarrow good$
$R_6 : \{3.5 - 3.79\} \wedge \{graduate\} \rightarrow good$
$R_7 : \{3.5 - 3.79\} \wedge \{undergraduate\} \rightarrow excellent$
$R_8 : \{3.8 - 4.0\} \rightarrow excellent$
$R_9 : \{poor\} \rightarrow weak$
$R_{10} : \{average\} \wedge \{senior, graduate\} \rightarrow weak$
$R_{11} : \{average\} \wedge \{freshman, sophomore, junior\} \rightarrow strong$
$R_{12} : \{good\} \rightarrow strong$
$R_{13} : \{excellent\} \rightarrow strong.$

Figure 3: Conditional Generalization Rules for GPA

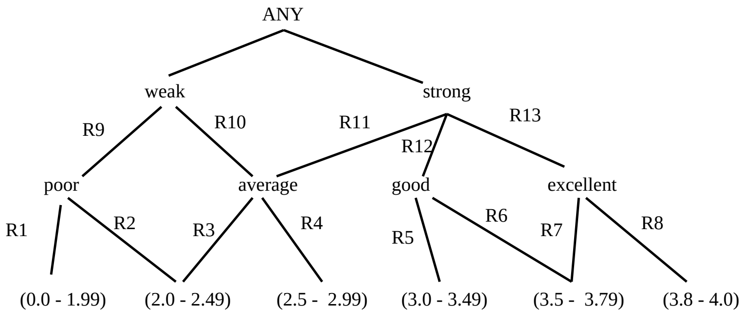

Figure 4: A rule-based concept graph for *GPA*

would no longer be generalized to "excellent" only, as would be the case if the concept tree in Figure 5 is being followed. Instead, it will be checked against the two rules R_6 and R_7 in Figure 7. If it is the GPA of a graduate student, then it will be generalized to "good"; otherwise, it must be an undergraduate, and will be generalized to "excellent".

Suppose the tuples in the initial relation in Example 1 has been generalized according to the rule-based concept graph in Figure 8. After comparison and merging, the prime relation resulted is the one in Table 1 [1]. In performing further generalization in Table 1, some rules which reference no other information than the generalized attribute values, such as R_9, R_{12} and R_{13} in Figure 7, can be applied directly to the prime relation to further generalize the GPA attribute. For example, the GPA value "good" in the second tuple can be generalized to "strong" according to R_{12}, and the value "poor" in the fifth row to "weak" according to R_9. However, for the GPA value "average" in the first tuple, it cannot be decided with the information in the prime relation which one of the two rules R_{10} and R_{11} should be applied. If the student is either a senior or graduate, then R_{10} should be used to generalize the GPA to "weak"; otherwise, it should be generalized to "strong". However, the status information (*freshman/sophomore/junior/senior*) has been lost during the previous generalization, and is not available in the prime relation. In fact, the 40

[1]The comparison and merging technique used here follows that proposed in [15]

Status	Sex	Age	GPA	Count
undergrad	M	16_25	average	40
undergrad	M	16_25	good	20
undergrad	F	16_25	excellent	10
...	...	...	...	...
grad	M	25_30	poor	6
grad	M	25_30	good	4
grad	F	25_30	excellent	4

Table 1: A Prime Relation from the Rule-Based Generalization.

student tuples in the initial relation which are generalized and merged into the first tuple may have all the different status. Therefore, if further generalization is performed, its value "average" will be generalized to both "weak" and "strong", and the tuple will be split into two generalized tuples. □

It is clear from Example 2 that further generalization from a prime relation may have difficulty for rule-based induction. Therefore, the generalization technique has to be modified to suit the rule-based case. In Section 5, we will describe a new method of using **path relation** instead of generalized relation to solve the induction anomaly problem.

5 Path Relation

Since any generalization could introduce induction anomaly into the generalized relation, any further generalization in the rule-based case has to be started again from the initial relation which has the full information. However, re-applying the deduction rules all over again on the initial relation is costly and wasteful. All the deduction that have been done previously in the generation of the prime relation is wasted, and has to be redo. In order to solve this problem, we propose to use a **path relation** to capture the generalization result from one application of the rules on the initial relation such that the result can be reused in all subsequent generalization.

An attribute value may be generalized to the root via multiple possible paths on a concept graph. However, for the attribute value of a given tuple in the initial relation, it can only be generalized via a unique path to the root. Each one of the multiple paths on which an attribute value can be generalized is a **generalization path**. Since the concepts on the graph are partially ordered, there are only a finite number of distinct generalization paths from the bottom level. In general, the number of generalization paths of an attribute should be small. Before an induction starts, a preprocessing is used to identify and label the generalization paths of all the attributes. For example, the generalization paths of the concept graph of GPA in Figure 8 are identified and labeled in Figure 9.

For every attribute value of a tuple in the initial relation, its generalization path can be identified by generalizing the tuple to the root. Therefore, each tuple in the initial relation is associated with a tuple of generalization paths. In a scan of the initial relation, every tuple can be transformed into a tuple of ids of the associated generalization paths. The result of the transformation is the **path relation** of the initial relation. It is important to observe that the path relation has captured completely the generalization result of the

$Path_1 : \{0.0 - 1.99\} \rightarrow poor \rightarrow weak \rightarrow ANY$
$Path_2 : \{2.0 - 2.49\} \rightarrow poor \rightarrow weak \rightarrow ANY$
$Path_3 : \{2.0 - 2.49\} \rightarrow average \rightarrow weak \rightarrow ANY$
$Path_4 : \{2.0 - 2.49\} \rightarrow average \rightarrow strong \rightarrow ANY$
$Path_5 : \{2.5 - 2.99\} \rightarrow average \rightarrow weak \rightarrow ANY$
$Path_6 : \{2.5 - 2.99\} \rightarrow average \rightarrow strong \rightarrow ANY$
$Path_7 : \{3.0 - 3.49\} \rightarrow good \rightarrow strong \rightarrow ANY$
$Path_8 : \{3.5 - 3.79\} \rightarrow good \rightarrow strong \rightarrow ANY$
$Path_9 : \{3.5 - 3.79\} \rightarrow excellent \rightarrow strong \rightarrow ANY$
$Path_{10} : \{3.8 - 4.0\} \rightarrow excellent \rightarrow strong \rightarrow ANY$

Figure 5: Generalization Paths for GPA

initial relation at all levels. In other words, given the generalization paths of a tuple, the generalized values of a tuple can be determined easily from the concept graph without redoing any deduction. Furthermore, the set of generalization paths on which some attribute values in the initial relation are generalized to the root can be determined during the generation of the path relation. By checking the number of distinct attribute values (concepts) on each level in the concept graph through which some paths found above have traversed, the minimum desirable level can be found. It can be concluded at this point that the path relation is an effective structure for capturing the generalization result in the rule-based case. By using it, the repetitive generalization required by roll-up and drill-down can be done in an efficient way without the problem introduced by the induction anomaly.

Name	Status	Sex	Age	GPA
J. Wong	freshman	M	18	3.2
C. Chan	freshman	F	19	2.8
D. Zhang	senior	M	21	2.7
...	...	...	...	...
A. Deng	senior	F	21	3.3
C. Ma	M.A.	M	28	2.3
E. Liu	senior	M	22	2.2
A. Chan	sophomore	M	19	2.4

Table 2: Initial Relation from the student database.

Example 3 Assume that the initial relation of the induction in Example 2 is the one in Table 2. Its path relation can be generated in one scan by using the generalization rules in Figure 7 and the path ids specified in Figure 9. (Figure 7 only has the rules for GPA, the rules of the other attributes are simple.) Table 3 is the path relation of Table 2. □

Another issue in rule-based generalization is the cyclic dependency. A generalization rule may introduce dependency between attributes. The generalization of an attribute value may depend on that of another

Status path id	Sex path id	Age path id	GPA path id
1	1	1	7
1	2	1	6
4	1	1	5
...	...	...	...
4	2	1	7
5	1	2	2
4	1	1	3
2	1	1	4

Table 3: Path Relation from the Initial Relation.

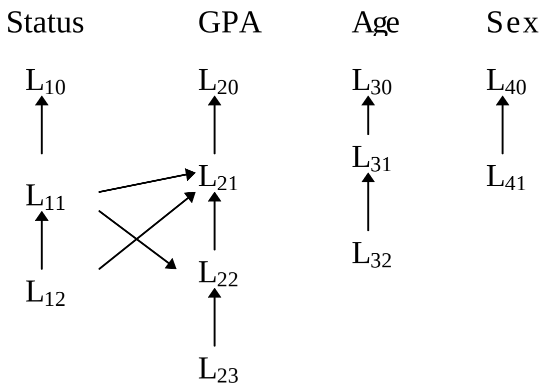

Figure 6: Generalization Dependency Graph

attribute. If the dependency is cyclic, it could introduce deadlock in the generalization process. In order to prevent cyclic dependency, rule-based induction creates a **generalization dependency graph** from the generalization rules and prevents deadlock by ensuring the graph is acyclic. The nodes in the generalization dependency graph are the levels of each attributes in the concept graph. (In the rest of the paper, we have adopted the convention of numbering the top level (root) of a concept graph level 0, and to increase the numbering from top to bottom. In other word, the bottom level would have the highest level number.) We use L_{ij} to denote the node associated with level j of an attribute A_i. In the dependency graph, there is an edge from every lower level node L_{ik} to the next higher level node $L_{i(k-1)}$ for each attribute A_i. Also, if the generalization of an attribute A_i from L_{ik} to level $L_{i(k-1)}$ depends on another attribute A_j in level L_{jm}, $(i \neq j)$, then there is an edge from L_{jm} to $L_{i(k-1)}$. In Figure 10, we have the generalization dependency graph of the generalization rules in Example 2. For example, the edges from L_{22} to L_{21}, from L_{12} to L_{21}, and from L_{11} to L_{21} are introduced by the rule R_{10}. If the dependency graph is acyclic, a *generalization order* of the concepts can be derived from a partial ordering of the nodes. Following this order, every attribute values of a tuple can be generalized to any level in the concept graphs. For example, a generalization order of the graph in Figure 10 is $(L_{12}, L_{11}, L_{23}, L_{22}, L_{21}, L_{10}, L_{20}, L_{32}, L_{31}, L_{30}, L_{41}, L_{40})$. Moreover, any tuple in the initial relation in Table 2 can be generalization to the root following this order.

5.1 Data Structure for Generalization

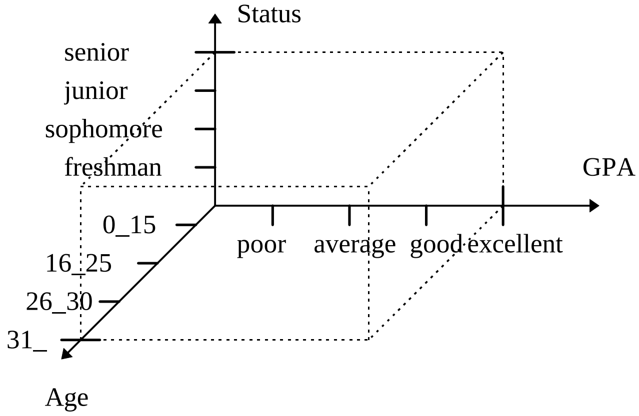

Figure 7: A multi-dimensional data cube

In the prototype of the DBMiner system (called DBLearn), the data structure of generalization relation is used to store the intermediate results. Both generalized tuples and their associated aggregate values such as "count" are stored in relation tables. However, it is discovered that generalized relation is not the most efficient structure to support insertion of new generalized tuples and comparison of identical generalized tuples. Furthermore, the prime relation may not have enough information to support further generalization in the rule-based case, which makes it an inappropriate choice for storing intermediate results. To facilitate rule-based induction, we propose to use either a **multi-dimensional data cube** or a **generalized-attribute tree**.

A data cube is a multi-dimensional array, as shown in Figure 11, in which each dimension represents a generalized attribute and each cell stores the value of some aggregate attributes, such as "count" or "sum". For example, the data cube in Figure 11 can store the generalization result of the initial relation in Table 2 to levels 2,2,1 of the attributes Status, GPA and Age. (Please refer to the generalized attributes of the levels in Figure 10.) Let v be a vector of desirable levels of a set of attributes, and the initial relation is required to be generalized to the levels in v. During the generalization, for every tuple p in the initial relation, the generalized attribute values of p with respect to the levels in v can be derived from its path ids in the path relation and the concept graphs. Let p_g be the tuple of these generalized attribute values. In order to record the count of p and update the aggregate attribute values, p_g is used as an index to a cell in the data cube. The count and the aggregate attribute values of p are recorded in this cell. For example, the count of the tuple (J.Wong, freshman, M, 18, 3.2) in the initial relation in Table 2 would be recorded in the cell whose index is the generalized tuple (undergraduate, M, 16_25, good).

Many works have been published on how to build data cubes [1, 6, 14]. In particular, how to compute data cubes storing aggregated values efficiently from raw data in a database. In our case, we only need to use a data cube as a data structure to store the "counts", i.e., number of tuples that have been generalized into a higher level tuple. Therefore, the details of how to compute a cube on aggregations from a base cube has no relevancy. For the AO induction algorithm, the cube is practically a multi-dimensional array.

In [17], the data cube has been compared with the generalized relation. It costs less to produce a data cube, and its performance is better except when the data is extremely sparse. In that case, the data cube may waste some storage. A more space efficient option is to use a b-tree type data structure. We propose to use a b-tree called **generalized-attribute tree** to store the count and aggregate attribute values. In this approach, the generalized tuple p_g will be used as an index to a node in the generalized-attribute tree and the count and aggregate values will be stored in the corresponding node. According to the experience in the DBMiner system, data cube is very efficient as long as the percentage of occupied cells is reasonably dense. Therefore, in rule-based induction, data cube is the favorable data structure; and the generalized-attribute tree should be used only in case the sparseness is extremely high.

6 An Efficient Rule-Based AO Induction Algorithm

We have shown that in the case of rule-based induction, it is more efficient to capture the generalization results in the path relation, and to use the data cube to store the intermediate results. In the following, we present the **path relation algorithm** for rule-based induction.

Algorithm 1 Path Relation Algorithm for Rule-Based Induction

Input : A task specification is input into a Rule-Based AO Induction System (DB, CH, DS, KR, t_a)
/* the initial relation R, whose attributes are A_i, $(1 \leq i \leq n)$, is retrieved from DB */

Method:

Step One: Inducing the prime relation R_{pm} from R

1: transform R into the path relation R_p;

2: compute the minimum desirable level L_i for each A_i $(1 \leq i \leq n)$;

3: create a data cube C with respect to the levels L_i $(1 \leq i \leq n)$;

4: scan R_p; for each tuple $p \in R_p$, compute the generalized tuple p_g with respect to the levels L_i, $(1 \leq i \leq n)$; update the count and aggregate values in the cell indexed by p_g;

5: convert C into the prime relation R_{pm};

Step Two: Perform progressive generalization to create rules

1: Select a set of attributes A_j and corresponding desirable levels L_j for further generalization;

2: create a data cube C for the attributes A_j with respect to the levels L_j;

3: scan the path relation R_p; compute the generalized tuple p_g for each tuple $p \in R_p$ with respect to the desirable levels L_j; update the count and aggregate values in the cell indexed by p_g;

4: convert all non-empty cells of C into rules.

Termination condition : Step Two will be repeated until the rules generated satisfy the discovery goal [2]. □

Explanation of the algorithm: In the first step, the path relation R_p is first created from the initial relation. After that, the minimum desirable levels L_i, $(1 \leq i \leq n)$, are computed by scanning R_p once, and a data cube C is created for the generalized attributes at levels L_i. In 4, R_p is scanned again and every tuple in R_p is generalized to levels L_i. For every p in R_p, its generalized tuple p_g is used as an index to

[2]The meaning of "discovery goal" follows those defined in [15], and will be discussed in the following paragraph.

locate a cell in C in which the count and other aggregate values are updated. At the end of the first step, the non-empty cells in C are converted to tuples of the prime relation R_{pm}.

The second step is the progressive generalization, it is repeated until the rules generated satisfy the discovery goal. There are two ways to define the discovery goal. In [15], a threshold was defined to control the number of rules generated. Once the number of rules generated is reduced beyond the given threshold, then the goal has been reached. Another way is to allow the generalization to go through an interactive and iterative process until the user is satisfied with the rules generated. In other words, no pre-defined threshold would be given, but the goal is reached when the user is comfortable with the rules generated. This is compatible with the roll-up and drill-down approach used in many data mining systems.

The details of step two in the algorithm is the following. In the beginning of each iteration, a set of attributes A_j and levels L_j are selected for generalization, and a data cube C is created with respect to the levels L_j. Following that, the tuples in R_p are generalized to the levels L_j in the same way as that in the first step. After all corresponding cells in the data cube have been updated, the non-empty cells are converted into rules. This can be repeated until the number of rules reached a pre-defined threshold or the user is satisfied with the rules generated.

In the above algorithm, if the rules discovered are too general and in a level which is too high, then the generalization can be redone to a lower level. Hence, it can perform not only progressive generalization, but also progressive specialization.

Example 4 We extend Example 2 here to provide a complete walk through of Algorithm 1. The task is to discover the characteristic rules from the computer science students in a university database. The initial relation has been extracted from the database, and it is presented in Table 2. In the following, we will describe the execution of Algorithm 1 on Table 2 in detail.

Input : The input are the same as those described in Example 2. The database D is the database of computer science students. The concept hierarchy CH is the one described in Figure 7. The initial relation R is the one in Table 2.

Step One: Inducing the prime relation R_{pm} from R

1: Scan R and generalize each tuple in R to the root with respect to the concept graph (Figure 7) to identify the associated path ids. For example, the 2.8 GPA of the second student in Table 2 can be generalized to "average" by R_4, and then to "strong" by R_{11} (Figure 7). (Note that R_{11} is applied instead of R_{10}, because the student is a "freshman".) Therefore, the path for the GPA attribute associated with this tuple is $Path_6$ (Figure 9), and the associated path id tuple is the second one in Table 3. Following this mechanism, every tuple in R is transformed into a tuple of path ids, and R is transformed into the path relation R_p in Table 3.

2: Compute the minimum desirable levels for each attribute in Table 2 by checking the number of concepts at each level through which some generalization paths identified in the previous step have traversed.

3: Create a data cube C with respect to the minimum desirable levels found.

4: Scan the path relation R_p (Table 3). For each tuple $p \in R_p$, by using the corresponding path ids, find out the generalized values p_g of p on the concept graphs with respect to the minimum levels. For example, assume the minimum level for GPA is found to be level 2, since the path id for

Status	Age	GPA	Count
undergrad	16_25	weak	10
undergrad	16_25	strong	60
grad	25_30	weak	6
grad	25_30	strong	8

Table 4: A Final Relation from the Rule-Based Generalization.

GPA of the first tuple in R_p (Table 3) is "7", it can be identified from $Path_7$ (Figure 9) that the generalized value at level 2 is "good". Once p_g is found, update the count and aggregate values in the cell indexed by p_g. For example, the first tuple of Table 3 is generalized to (undergrad, M, 16_25, average), the count in the corresponding cell is updated.

5: For every nonempty cell in C, a corresponding tuple can be created in a generalized relation, and the result is the prime relation R_{pm} in Table 1. For example, the count in the cell in C indexed by (undergrad, M, 16_25, average) has count equal to 40 and it is converted to the first tuple in R_{pm} (Table 1).

Step Two: Perform progressive generalization to create rules

1: In order to perform further generalization on the prime relation R_{pm} (Table 1), the attributes Sex and Birthplace are removed and GPA is generalized one level higher from level 2 to level 1.

2: A data cube C is created for the remaining attributes Status, Age and GPA with respect to the new levels. (In fact, only GPA is moved one level higher.)

3: Scan the path relation R_p to compute the generalized tuple p_g for each tuple $p \in R_p$ with respect to the new levels, and update the count and aggregate values in the cell.

4: Convert all non-empty cells of C into a generalized relation, and the result is the final relation in Table 4. If the final relation can satisfy the user's expectation, it is then converted to the following rule :

$$\begin{array}{l}
\forall(x) cs_student(x) \rightarrow \\
(Status(x) \in undergraduate \wedge (16 \leq Age(x) \leq 25) \wedge GPA(x) \in weak)[10.4\%] \vee \\
(Status(x) \in undergraduate \wedge (16 \leq Age(x) \leq 25) \wedge GPA(x) \in strong)[62.5\%] \vee \\
(Status(x) \in graduate \wedge (25 \leq Age(x) \leq 30) \wedge GPA(x) \in weak)[6.25\%] \vee \\
(Status(x) \in graduate \wedge (25 \leq Age(x) \leq 30) \wedge GPA(x) \in strong)[20.8\%]
\end{array}$$

□

Let us analyze the complexity of the path relation algorithm. The cost of the algorithm can be decomposed into the cost of induction and that of deduction. The deduction portion is the one time cost of generalizing the attribute values to the root in building the path relation. The induction portion covers those spent on inducing the prime and the final relations. The induction portion of the algorithm is very efficient, which will be discussed in the following theorem. However, the cost of the deduction portion will depend on the complexity of the rules and the efficiency of the deduction system DS. A general deductive database system may consist of complex rules involving multiple levels of deduction, recursion, negation, aggregation, etc. and thus may not exist an efficient algorithm to evaluate such rules [26]. However, the deduction rules in the algorithm are the conditional rules associated with a concept graph, which in most cases, are very simple conditional rules. Therefore, it is practical to assume each generalization in the deduction process is bounded by a constant in the analysis of the algorithm. When the concept graphs involve more complex deduction rules, the complexity of the algorithm will depend on the complexity of the deduction system.

The following theorem shows the complexity of the path relation algorithm under the assumption of the bounded cost of the deduction processes.

Theorem 1 *If the cost of generalizing an attribute to any level is bounded, the complexity of the path relation algorithm for rule-based induction is $O(n)$, where n is the number of tuples in the initial data relation.*

Proof. In the first step, the initial relation and the path relation are scanned once in steps 1 and 4. The time to access a cell in the data cube is constant, therefore, the complexity of this step is $O(n)$.

In the subsequent progressive generalization, assume that the number of rounds of generalization is k, which is much smaller than n. In each round, the path relation will be scanned once only. Therefore, the complexity is bounded by $n \times k$. Adding the costs of the two steps together, the complexity of the entire induction process is $O(n)$. □

7 Performance Study

Our analysis in Section 6 has shown that the complexity of the relation path algorithm is $O(n)$, which is as good as that of the algorithms proposed for the more restricted non-rule-based case. Moreover, the path relation algorithm proposed here is more efficient than a previously proposed backtracking algorithm [7] which has a complexity of $O(n \log n)$. To confirm this analysis, an experiment has been conducted to compare the performance between the path relation algorithm and the backtracking algorithm.

There are two main differences between the two algorithms. (1) The generalization in the backtracking algorithm uses generalized relation as the data structure. (2) All further generalization after the prime relation has been generated is based on the information in the prime relation. As has been explained, the prime relation will introduce the induction anomaly in rule-based induction. Because of that, the generalized tuples in the prime relation have to be backtracked to the initial relation and split according to the multiple possible generalization paths in the further generalization. The backtracking and splitting has to be performed in every round of progressive generalization. This impacts the performance of the backtracking algorithm when comparing with the path relation algorithm. The path relation algorithm is more efficient because the path relation has captured all the necessary induction information in the path ids and its tuples can be generalized to any level in the rule-based case.

For comparison purpose, both algorithms were implemented and executed on a synthesized student database similar to the one in Example 1. The records in the database have the following attributes { Name, Status, Sex, Age, GPA}. The records are generated such that values in each attribute are random within the range of possible values and satisfy some conditions. The following conditions are observed so that the data will contain some interesting patterns rather than being completely random.

1. Graduate students are at least 22 years old.

2. Ph.D. students are at least 25 years old.

3. Graduate students' GPAs are at least 3.00.

For each attribute, there is a corresponding rule-based concept graph same as that in Example 1. In order to compare the two algorithms, we assume the attributes Status, Sex, Age, GPA are generalized to level 1,1,1,3 respectively in the prime relation, (the levels are those indicated in Figure 10). Following that, GPA is further generalized to level 2 and then level 1. Hence the the generalization order is:

$1,1,1,3 \rightarrow 1,1,1,2 \rightarrow 1,1,1,1 \rightarrow 1,1,0,1$.

Number of of records	No. of tuples in final relation	Path relation Alg	Backtracking Alg. (case 1)	Backtracking Alg. (case 2)
10000	6	192	4028	24033
10000	12	191	4028	24033
10000	24	190	1969	11972
1000	6	21	395	2396
1000	12	20	395	2396
1000	24	19	197	1196

Figure 8: Number of pages read

Since the cost in the algorithms is dominated by the scanning in the database, the metric in the comparison is the number of pages required to read and write.

For the path relation algorithm, the initial relation is first transformed into the path relation, and the algorithm use the path relation to generalize the records. For each generalization step, we use a data cube to store the counts and aggregate values of the generalized tuples. The number of pages written is much smaller than the number of pages read as the path relation contains only the path id for the data of each attribute and is much smaller than the initial relation.

For the backtracking algorithm, the initial relation is used to form the prime relation and a virtual attribute is added to the initial relation to record the linkages between the tuples in the initial and prime relations. (Please see [7] for details). In the generalization of the attribute GPA from level 3 to level 2, the induction anomaly occurs. The initial relation with the virtual attribute is read and the selected attribute GPA is generalized and form the enhanced-prime relation by merging the tuples. The enhanced-prime relation has the virtual attribute values to support splitting of the prime relation for the further generalization. Two cases are considered for the implementation of the enhanced-prime relation. In the first case the enhanced-prime relation is stored in the main memory. In the second case, the enhanced-prime relation is stored in the secondary storage and the pages read and written are counted accordingly.

The results of the comparison are shown in Figures 12 and 13. It can be seen from the two figures that the number of pages read and written by the path relation algorithm is much smaller. And the reduction ratio between the two algorithms is comparable to the ratio between n and $n \log n$, where n is the number of records, which matches our analysis in the previous section. This has clearly demonstrated that the path relation algorithm is more efficient. As an example, the final relation for generalizing 10000 records with the final relation threshold set to 8 is shown in Figure 14.

Number of of records	No. of tuples in final relation	Path relation Alg	Backtracking Alg. (case 1)	Backtracking Alg. (case 2)
10000	6	40	259	261
10000	12	40	259	261
10000	24	40	219	220
1000	6	4	26	28
1000	12	4	26	28
1000	24	4	22	23

Figure 9: Number of pages written

Status	Sex	GPA	Count
Undergraduate	M	Weak	1545
Undergraduate	M	Strong	1326
Undergraduate	F	Weak	1574
Undergraduate	F	Strong	1266
Graduate	M	Strong	2167
Graduate	F	Strong	2122

Figure 10: Final relation

8 Discussion

The path relation has an important role in the rule-based induction proposed above. If the initial relation is very large, it may not be cost effective to create another large table as the path relation. A better alternative would be to encode and store the path ids in the initial relation. Also, it is useful to note that it is not necessary to encode all the paths in a concept graph together. For example, in Figure 8, for a given GPA value, only one bit at each level is sufficient to record the generalization path of the GPA value. Therefore, in order not to create another large relation (path relation), generalization paths can be tagged to tuples in the initial relation with a minimal amount of space.

On the other hand, if it is feasible to create a path relation, then another interesting observation is that the path relation can be reduced without lossing of any information. In the generation of the path relation, the path id tuples generated are stored in the same order as that in the initial relation and redundant path id tuples are not compared and merged. Hence, the size of both relations are equal. If the initial relation has m relevant attributes A_i, $(1 \leq i \leq m)$, then the number of distinct tuples in the path relation would be less than $\rho = \prod_i^m \rho_i$, where ρ_i, $(1 \leq i \leq m)$, is the number of distinct generalization paths in the concept graph of attribute A_i. In some cases, ρ would be much smaller than the size of the initial relation, and it would be beneficial to merge the redundant path id tuples in the path relation and record their counts there. By using this compact path relation, the cost of induction will be reduced significantly in the progressive generalization.

In [17], it is suggested that mining characterization rules should be multiple-level. And the techniques of progressive generalization and specialization should be used. The techniques rely on a *minimally generalized relation* in which concepts are generalized to some minimal levels such that it can be used to perform drill-down and roll-up from any level. The compact path relation discussed above has all necessary information to support any generalization and its size is smaller than the initial relation. Therefore, it can be a good candidate of minimally generalized relation for performing rule-based progressive generalization and specialization. In fact, because of the induction anomaly problem, it would not be efficient to use other generalized relation as a minimally generalized relation.

Many large organizations have started to use data warehouse to store valuable data from different information sources. In some cases, the data could come from 20 to 30 legacy sources. In this type of warehouse, the number of attributes of an object could be in the order of hundreds. One such example is the data warehouses in large international banking corporations. In order to perform rule-based induction on the data in large warehouses, the path relation algorithm is an excellent choice because many warehousing systems have already implemented the data cube technique for on-line analytical processing (OLAP) and decision support.

9 Conclusion

A rule-based attribute-oriented induction technique is proposed and studied in this paper. The technique extends the previously developed basic attribute-oriented induction method and make the method more generally applicable to databases containing both data and rules. Also, the given concept hierarchies may contain unconditional and conditional rules, which enlarges the application domain and handles more sophisticated situations.

Rule-based concept graph extends substantially the representation power of the background knowledge in an induction system. It also has converted the discovery system into an integrated deduction system and may eventually lead to the integration of a discovery system into an intelligent and cooperative information system [5].

This study has developed a general model of rule-based database induction and extended the basic AO Induction to the rule-based AO induction which handles induction on a rule-based concept graph. A new technique of using path relation and data cube to replace the initial relation and the generalized relations has been proposed to facilitate a more efficient induction. The proposed path relation algorithm has an improved complexity of $O(n)$ which is faster than the previously proposed rule-based induction algorithms. The performance studies have also confirmed that the path relation algorithm is more efficient than the "backtracking" algorithm.

A data mining system DBMiner has been developed for interactive mining of multiple-level knowledge in large databases [17]. The system has implemented a wide spectrum of data mining techniques, including generalization, characterization, association, classification, and prediction. Our proposed rule-based induction can extend the mining capability of DBMiner. In particular, a rule-based progressive generalization and specialization could be very useful in mining multiple-level knowledge. Furthermore, unconditional and rule-based induction, including both generalization and specialization, in multidatabases or data warehouses with large number of views are among the interesting topics in data mining for future research.

References

[1] S. Agrawal, R. Agrawal, P.M. Deshpande, A. Gupta, J.F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. In *Proceedings of the International Conference on Very Large Databases*, pages 506-521, Bombay, India, September 1996.

[2] R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami. An interval classifier for database mining applications. *Proc. 18th Int. Conf. Very Large Data Bases*, pp. 560–573, Vancouver, Canada, August 1992.

[3] R. Agrawal, T. Imielinski, and A. Swami. Database mining: A performance perspective. *IEEE Trans. on Knowledge and Data Engineering*, Vol. 5, pp. 914–925, 1993.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proc. 1994 Int. Conf. Very Large Data Bases*, pp. 487–499, Santiago, Chile, September 1994.

[5] M.L. Brodie and S. Ceri. On Intelligent and Cooperative Information Systems: A Workshop Summary. *International Journal of Intelligent and Cooperative Information Systems*, V1 N2:233–248, 1992.

[6] S. Chaudhuri et al. An Overview of Data Warehousing and OLAP Technology. ACM-SIGMOD Record, Vol. 26 No.1 P.65-74, March 1997

[7] D.W. Cheung, A.W. Fu, and J. Han. Knowledge discovery in databases: A rule-based attribute-oriented approach. *Proc. 1994 Int. Symp. on Methodologies for Intelligent Systems*, pp. 164–173, Charlotte, North Carolina, October 1994.

[8] D.W. Cheung, J. Han, V.T. Ng, and C.Y. Wong. Maintenance of Discovered Association Rules in Large Databases: An Incremental Updating Technique. *Proc. 1996 IEEE Int. Conf. on Data Engineering*, New Orleans, Louisiana , February 1996.

[9] D.W. Cheung, J. Han, V.T. Ng, A.W. Fu, and Y. Fu. A Fast Distributed Algorithm for Mining Association Rules. *Proc. Fourth International Conference on Parallel and Distributed Information System (PDIS-96)*, Miami Beach, Florida, December 1996.

[10] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining.* AAAI/MIT Press, 1995.

[11] D. Fisher. Improving inference through conceptual clustering. In *Proc. 1987 AAAI Conf.*, pages 461–465, Seattle, Washington, July 1987.

[12] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases: An overview. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 1–27. AAAI/MIT Press, 1991.

[13] H. Gallaire, J. Minker, and J. Nicolas. Logic and databases: A deductive approach. *ACM Comput. Surv.*, 16:153–185, 1984.

[14] J. Gray, A. Bosworth, A. Layman, and H. Piramish. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total. In *Proceeding of the 12th Intl. Conference on Data Engineering*, pages 152-159, New Orleans, February 1996.

[15] J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases: An attribute-oriented approach. In *Proc. 18th Int. Conf. Very Large Data Bases*, pages 547–559, Vancouver, Canada, August 1992.

[16] J. Han, Y. Cai, and N. Cercone. Data-driven discovery of quantitative rules in relational databases. *IEEE Trans. Knowledge and Data Engineering*, 5:29–40, 1993.

[17] J. Han, Y. Fu, W. Wang, J. Chiang, W. Gong, K. Koperski, D. Li, Y. Lu, A. Rajan, N. Stefanovic, B. Xia, O. R. Zaiane. DBMiner: A System for Mining Knowledge in Large Relational Databases. In *Proc. 1996 Int'l Conf. on Data Mining and Knowledge Discovery (KDD'96)*, Portland, Oregon, August 1996.

[18] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. *Proc. 1995 Int. Conf. Very Large Data Bases*, Zurich, Switzerland, pp. 420–431, Sept. 1995.

[19] J. Han and Y. Fu. Exploration of the power of attribute-oriented induction in data mining. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 399–421. AAAI/MIT Press, 1995.

[20] V. Harinarayan, A. Rajaraman and J. Ullman. Implementing Data Cubes Efficiently. *Proc. 1996 ACM-SIGMOD Int. Conf. Management of Data*, Montreal, Canada, April, 1996.

[21] K. A. Kaufman, R. S. Michalski, and L. Kerschberg. Mining for knowledge in databases: Goals and general description of the INLEN system. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 449–462. AAAI/MIT Press, 1991.

[22] D. Haussler. Learning conjuctive concepts in structural domains. In *Proc. 1987 AAAI Conf.*, pages 466–470, Seattle, Washington, July 1987.

[23] H.Y. Hwang and A.W. Fu. Efficient Algorithms for Attribute-Oriented Induction. In *Proc. 1995 1st Int. Conf. on Knowledge Discovery and Data Mining (KDD'95)*, Montreal, Canada, Aug., 1995.

[24] R. S. Michalski. A theory and methodology of inductive learning. In Michalski et. al., editor, *Machine Learning: An Artificial Intelligence Approach, Vol. 1*, pages 83–134. Morgan Kaufmann, 1983.

[25] A. Silberschatz, M. Stonebraker, and J. Ullman. Database Research: Achievements and Opportunities Into the 21st Century. *Report of an NSF Workshop on the Future of Database Systems Research*, May, 1995.

[26] J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Vols. 1 & 2.* Computer Science Press, 1989.

[27] J. Widom. Research Problems in Data Warehousing. In *Proc. 1995 4th Int. Conf. on Information and Knowledge Management (CIKM)*, November, 1995.

[28] J. Zytkow and J. Baker. Interactive mining of regularities in databases. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 31–54. AAAI/MIT Press, 1991.