

Short Paper

Rules Generation From the Decision Tree

DING-AN CHIANG, WEI CHEN^{*}, YI-FAN WANG⁺ AND LAIN-JINN HWANG

*Department of Information Engineering
Tamkang University*

Tanshui, Taipei, Taiwan 251, R.O.C.

E-mail:chiang@cs.tku.edu.tw

^{}National Taipei College of Nursing*

Taipei, Taiwan 112, R.O.C.

E-mail:wei@ntcn.edu.tw

⁺Chang Gung Institute of Nursing

Taoyuan, Taiwan 333, R.O.C.

To avoid checking unnecessary or irrelevant conditions of rules, the irrelevant values problem of the decision tree is addressed. We propose an algorithm to remove irrelevant conditions of rules in the process of converting the decision tree to rules according to the semantics of the decision tree. Since our algorithm depends only on the semantics of the decision tree, our algorithm can be integrated into any existing tree-construction algorithm with negligible increase in computational cost concerning that of constructing the decision tree. Moreover, as a side effect, the resultant rules are less likely to suffer from missing branches problem.

Keywords: decision tree, irrelevant values problem, missing branches problem, classification rules, learning for examples

1. INTRODUCTION

Classification is one of the key data mining technique and has been applied to numerous applications. Until now, a number of different classification techniques have been proposed [2-4] and the induction of decision trees is a well-known approach [8]. A decision tree is a representation of a decision procedure for determining the class of a given instance [18] and it can be constructed by the non-incremental tree-induction algorithm [8, 11-13] or the incremental tree-induction algorithm [2, 17, 18]. The non-incremental tree-induction algorithm, such as ID3, has been studied by many authors. This kind of algorithm infers a decision tree once, based on the entire training data. On the other hand, the incremental tree-induction algorithm, such as ID4 [2], ID5 [17], and ID5R [18], revises the current decision tree, if necessary, in response to each newly observed training data. However, as pointed out by P. Utgoff [18], ID4 builds the same

Received May 16, 1998; revised June 10, 1999, November 18, 1999, & February 25, 2000;
accepted April 20, 2000

Communicated by Jhing-Fa Wang.

tree as the basic ID3 only when there is an attribute at each non-leaf node that is clearly the best choice with its E-score. ID5R is a successor of ID5 algorithm and it is guaranteed to build the same decision tree as ID3 for the same given training data. Since these algorithms are ID3-like algorithms, we only consider ID3 in this paper.

ID3 builds a decision tree by selecting the best test attribute by examining their information gain. Attribute with maximum information gain is selected to be the root of the decision tree. Then, the same procedure is operated on each branch to induce the remaining levels of the decision tree until all examples in a leaf belong to the same class. ID5R was developed by P. Utgoff [18], and is an incremental tree-induction algorithm. ID5R uses pull-up technique to reconstruct the tree so that the desired attribute is at the root of the tree or subtree without re-examining the training instances. The main difference between the structure of the ID3 tree and that of the ID5R tree is at the node of the decision tree. The node of the ID3 tree does not contain any information about the training data; on the other hand, the node of the ID5R tree contains all necessary information for the training set. For example, as shown in Figure 1, the leaf node of the ID5R tree contains a class name and the set of instance descriptions at the node belonging to the class, and the leaf node of the ID3 tree only contains a class name that the branch belongs to. Moreover, the non-leaf node (or decision node) of the ID5R tree also contains (a) an attribute test, with a branch to another decision tree for each possible value of the attribute, the positive and negative counts for each possible value of the attribute, and (b) the set of non-test attributes at the node, each with positive and negative counts for each possible value of the attributes [18].

For the tree-induction algorithms, such as ID3 and ID5R, both trees generated by these algorithms may have over-specialization problem because when an attribute is selected for branching out a node, the decision tree creates a branch for each value of that appear in the training data without considering whether the value is relevant to the classification. Fayyad has proposed two famous algorithms, GID3 and GID3*, to solve

eyes	hair	height	class
brown	blond	short	N
brown	dark	tall	N
blue	blond	tall	P
blue	dark	tall	N
blue	dark	short	N
blue	red	tall	P
brown	blond	tall	N
blue	blond	short	P

the over-specialization problem of the decision tree constructed using ID3 [5-7]. These two algorithms overcome not only the irrelevant values problem but also the missing branches problem. However, the problem of these algorithms is that some branches in the GID3 tree or the GID3* tree may be longer than that in the ID3 tree. Consequently, more attributes need to be examined in the process of making decision using the GID3 tree or the GID3* tree. We will discuss this problem in section 3.

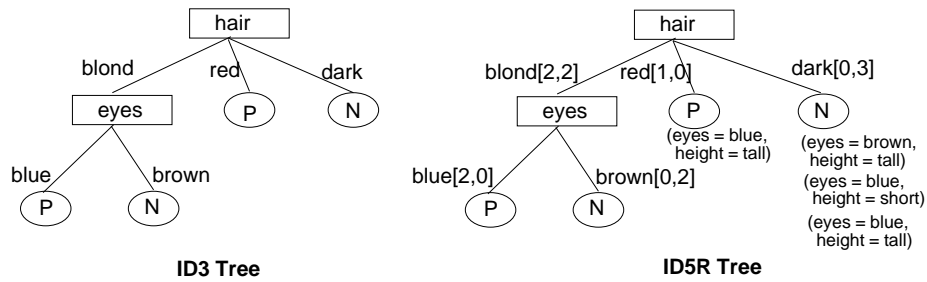


Fig. 1. Two decision trees for a simple training set.

As pointed out by J. R. Quinlan [14], large decision trees are difficult to understand because each node has a specific context established by the outcomes of tests at antecedent nodes and the structure of the decision tree may cause individual sub-concepts to be fragmented. Rewriting the tree to a collection of rules, one for each leaf in the tree, would not result in anything much simpler than the tree, since there would be one rule for every leaf. However, since the antecedents of a rule may contain irrelevant conditions, the rule can be generalized by deleting these superfluous conditions without affecting its correctness. Consequently, the generalized rules are simpler than the tree. In this paper, we provide another view to solve over-specialization problem of the decision tree without the problem in the GID3 tree or the GID3* tree. We eliminate irrelevant values during the process of converting the decision tree to rules according to the semantics of the decision tree, so that the irrelevant conditions will not appear in the resultant rules. Unlike Quinlan’s approach [13, 14], since our algorithm depends only on the semantics of the decision tree, our algorithm has clear computational advantages and can be integrated into any existing decision tree system easily. Moreover, since irrelevant conditions are removed from the resultant rules, the resultant rules are more general than those represented by the decision tree. As a side effect, the resultant rules are less likely to suffer from missing branches problem.

In this paper, the over-specialization problem of the ID3 tree is introduced in section 2. The problem of GID3 and GID3* algorithms are briefly introduced in section 3. The idea of the virtual branches in the decision tree is discussed in section 4. An efficient algorithm for removing irrelevant values is given in section 5. The empirical evaluation of our algorithm is discussed in section 6. The concluded remarks are stated in section 7.

2. PROBLEM STATEMENTS

As pointed out in [5], the irrelevant values problem and the missing branches problem are two causes of over-specialization of the decision tree. The ID3 tree and ID5R tree have the irrelevant values problem because when an attribute is selected for branching out a node, both trees create a branch for each value of that attribute appear in the training data. Since some values of that attribute may not be relevant to the classification, the rules of the decision tree may have irrelevant conditions, which demands irrelevant information to be supplied [4, 5, 14].

The missing branches problem of the decision tree is due to the fact that some of the reduced subsets at the non-leaf nodes do not necessarily contain examples of every possible value of the branching attribute [5]. Consequently, the decision tree may fail to classify some instances. Now, we introduce the irrelevant values problem and the missing branches problem by the following example.

Example 1. Let us consider the ID3 tree in Fig. 2. According to [14], rewriting the tree to a collection rules, the conjunction of the condition attribute values on the branches is involved in the antecedent of the rule, and the target attribute value (class) in the leaf node is involved in the consequent of the rule. In this example, the rules of the tree are:

1. $a_3 \rightarrow c_1$,
2. $a_1 \wedge b_1 \rightarrow c_1$,
3. $a_1 \wedge b_2 \rightarrow c_2$,
4. $a_2 \wedge b_1 \rightarrow c_1$,
5. $a_2 \wedge b_2 \rightarrow c_3$.

As shown in Fig. 2, since the attribute B is not in the branch Br , we assume that all branches with respect to attribute A and attribute B implied by Br are $a_3 \wedge b_j \rightarrow c_1$, where $b_j \in \text{dom}(B)$ and $\text{dom}(B)$ is called the domain of attribute B . Moreover, since $a_1 \wedge b_1 \rightarrow c_1$, $a_2 \wedge b_1 \rightarrow c_1$, and $a_3 \wedge b_1 \rightarrow c_1$ are the rules of the tree, we can conclude that the value a_1 is an irrelevant condition in the rule $a_1 \wedge b_1 \rightarrow c_1$, and the value a_2 is also an irrelevant condition in the rule $a_2 \wedge b_1 \rightarrow c_1$. When we remove the irrelevant values from these two rules, the resultant rules, which are simpler than the rules with irrelevant condition, are:

1. $a_3 \rightarrow c_1$,
2. $b_1 \rightarrow c_1$,
3. $a_1 \wedge b_2 \rightarrow c_2$,
4. $a_2 \wedge b_2 \rightarrow c_3$.

Let us re-consider the decision tree in Fig. 2. Let the testing data e be (a_4, b_1) . Since data e is not in the training set, the corresponding branch with respect to this data is missing in the tree. Therefore, the ID3 tree fails to classify e . However, after converting the decision tree to rules, this testing data, e , can be classified by rule $b_1 \rightarrow c_1$. Accordingly, the missing branches problem can be avoided, as a side effect, in the process of removing irrelevant conditions of rules. \square

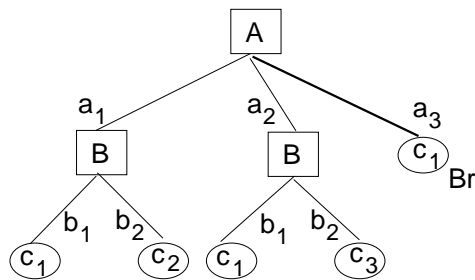


Fig. 2. An ID3 decision tree with irrelevant values.

3. GID3 AND GID3*

To overcome the over-specialization problem of the decision tree, Fayyad has proposed two algorithms, GID3 and GID3*, to solve this problem. GID3 and GID3* solve the irrelevant values problem at attribute phantomization step before attribute selection. In these two algorithms, irrelevant values are merged to a single value *DEFAULT*, and the other values remain unchanged.

The difference between GID3 and GID3* is only in the attribute phantomization step. In GID3, attribute phantomization is achieved evaluating the *Gain* of each attribute-value pair separately. The maximum *Gain* multiplied by the user specified tolerance, TL, gives a threshold gain. All attribute-value pairs whose *Gains* are not less than the threshold gain are considered relevant and are passed; otherwise, they are irrelevant and are mapped to *DEFAULT*. The problem of GID3 is that TL is specified by users. To provide a well-defined algorithm, user does not need to specify TL in GID3*. GID3* uses the *Tear* of the attribute-value pair to determine whether the attribute value is irrelevant and is grouped with other values under the *DEFAULT* value.

Although the irrelevant values problem has been overcome by GID3 and GID3*, some branches in the GID3 tree or GID3* tree may be longer than that in the ID3 tree. In other words, more attributes need to be examined to classify the instances by the GID3 tree or the GID3* tree than that using the ID3 tree in some cases. For example, Fig. 3 illustrates how an GID3* tree generates such problem for the given training set. As shown in Fig. 3, to classify the instances *e1*, *e5*, *e7*, and *e9* of the given data, we have to examine the value of “SiO₂ flow” attribute in the GID3* tree. However, we do not need to consider this attribute in the ID3 tree.

	Temp	Selectivity	line width	SiO ₂ flow	Class
<i>e1</i>	100	normal	normal	low	HP
<i>e2</i>	102	low	low	low	LF
<i>e3</i>	105	normal	high	v. high	LP
<i>e4</i>	110	high	high	high	LP
<i>e5</i>	100	high	low	v. low	LF
<i>e6</i>	101	low	normal	normal	LF
<i>e7</i>	115	normal	normal	normal	HP
<i>e8</i>	112	high	high	high	LP
<i>e9</i>	111	normal	low	low	HP

4. THE VIRTUAL BRANCHES

Let *A* be a set of attributes $\{A_1, \dots, A_n\}$, *C* be a set of classes $\{C_1, \dots, C_s\}$, and the set of possible values for an attribute *A_j* is denoted by *dom*(*A_j*). To represent a decision tree by a set of branches, the branch *Br* of the decision tree can be represented as the form $(Br[A_1], \dots, Br[A_n], C_i)$ or $(Br[A_1, \dots, A_n], C_i)$, where *Br*[*A_j*] is the branch value out of an attribute *A_j* in the branch *Br*. Moreover, according to [13], this branch can be easily converted into the following rule:

$$Br[A_1] \wedge \dots \wedge Br[A_n] \rightarrow C_i.$$

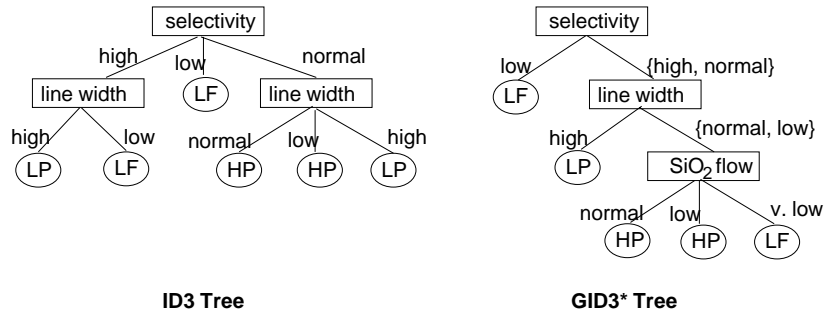


Fig. 3. Two decision trees for a simple training set.

For the ID3 tree, not all attributes A_j will be the nodes of some branches Br in the decision tree, that is, the corresponding values $Br[A_j]$ may be missing in the branches. When the corresponding values $Br[A_j]$ are missing in the branches, we say these attributes are irrelevant attributes with respect to Br . According to the semantics of irrelevant attributes, we know that the missing value $Br[A_j]$ can be replaced by any values from the same domain without affecting the correctness of the corresponding rule of Br . Therefore, when we consider these irrelevant attributes, there are many rules implied by Br . These implied rules are called virtual branches of the decision tree. We explain this observation by the following definition:

Definition 1. Let $Br = (Br[A_1, \dots, A_{j-1}], C_i)$ be a branch in the decision tree. Then, the virtual branches with respect to attributes A_1, \dots, A_n implied by Br are:

$$\{ (Br[A_1, \dots, A_{j-1}], a_{jr}, \dots, a_{ns}, C_i) \mid a_{jr}, \dots, a_{ns} \in \text{dom}(A_j, \dots, A_n) \},$$

where $\text{dom}(A_j, \dots, A_n) = \text{dom}(A_j) \times \dots \times \text{dom}(A_n)$.

For easy explanation how to identify irrelevant values of a branch in a decision tree, we further define the following definition.

Definition 2. Let Br and Br' be two different branches in a decision tree, where $Br = (Br[A_1, \dots, A_k], C_{i1})$. Then Br is in conflict with Br' with respect to attributes A_1, \dots, A_k , iff $(Br[A_1, \dots, A_k], C_{i2})$ is a part of virtual branch of Br' and $C_{i1} \neq C_{i2}$.

According to the semantics of irrelevant values, a value, $Br[A_j]$, is an irrelevance of a rule, this value can be deleted or replaced by any value from the same domain value without affecting the correctness of the rule. Therefore, based on the idea of virtual branches, we can trivially do combinatorial explosion in the number of comparisons to all the branches to identify irrelevant values of a branch. Consequently, we can remove irrelevant values from the corresponding rule of the branch. The following theorem presents a naive algorithm to identify irrelevant values of a branch.

Theorem 1. Let $Br = (Br[A_1, \dots, A_k], C_{i1})$ be a branch in a tree. When Br is not in conflict with any other branch with respect to attributes $A_1, \dots, A_{j-1}, A_{j+1}, \dots, A_k$ in the

decision tree, the branch value $Br[A_j]$ is an irrelevant value of the branch; otherwise, $Br[A_j]$ is not an irrelevant value of the branch.

Proof: (\Rightarrow) Let Br is not in conflict with any other branch with respect to attributes $A_1, \dots, A_{j-1}, A_{j+1}, \dots, A_k$ in the decision tree. Assume that $Br[A_j]$ is not an irrelevant value in Br . When Br is not in conflict with any other branch with respect to attributes $A_1, \dots, A_{j-1}, A_{j+1}, \dots, A_k$ in the decision tree, $Br[A_1] \wedge \dots \wedge Br[A_{j-1}] \wedge Br[A_{j+1}] \wedge \dots \wedge Br[A_k] \rightarrow C_{i1}$ can be a rule of the decision tree. Consequently, $Br[A_j]$ can be deleted from Br without affecting the correctness of the corresponding rule. That is, $Br[A_j]$ is an irrelevant value in Br .

(\Leftarrow) Let $Br[A_j]$ is an irrelevant value in Br . Assume that Br is in conflict with at least one branch with respect to attributes $A_1, \dots, A_{j-1}, A_{j+1}, \dots, A_k$ in the decision tree. According to the semantics of irrelevant values, when the value $Br[A_j]$ is an irrelevant value in Br , $Br[A_1] \wedge \dots \wedge Br[A_{j-1}] \wedge Br[A_{j+1}] \wedge \dots \wedge Br[A_k] \rightarrow C_{i1}$ is a rule of the decision tree, and $Br[A_1] \wedge \dots \wedge Br[A_{j-1}] \wedge Br[A_{j+1}] \wedge \dots \wedge Br[A_k] \rightarrow C_{i2}$ must not be a part of rule implied by other branch in the tree with respect to attributes $A_1, \dots, A_{j-1}, A_{j+1}, \dots, A_k$, where $C_{i1} \neq C_{i2}$. In other words, by definition 2, Br is not in conflict with any other branch with respect to attributes $A_1, \dots, A_{j-1}, A_{j+1}, \dots, A_k$ in the decision tree; otherwise, it is a contradiction. \square

To identify all irrelevant values of a branch, we can recursively apply Theorem 1 until all branch values have been checked.

Example 2. Let us consider the decision trees in Fig. 4, and we want to identify irrelevant values of each branch in the trees. Some attributes are missing in some branches. To identify irrelevant values of a branch, by Theorem 1, we have to consider the virtual branches of each branch with respect to different attributes in the tree. By Definition 1, the virtual branches of each branch with respect to attributes A, B , and D in the tree are:

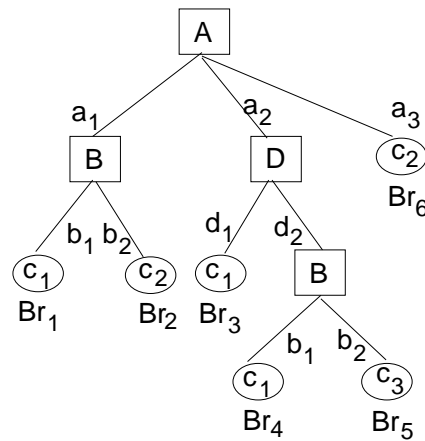


Fig. 4. An ID3 decision tree.

- $Br_1 = \{ (a_1, b_1, d_i, c_1) \mid d_i \in \text{dom}(D) \}$.
- $Br_2 = \{ (a_1, b_2, d_i, c_2) \mid d_i \in \text{dom}(D) \}$.
- $Br_3 = \{ (a_2, b_i, d_1, c_1) \mid b_i \in \text{dom}(B) \}$.
- $Br_4 = \{ (a_2, b_1, d_2, c_1) \}$.
- $Br_5 = \{ (a_2, b_2, d_2, c_3) \}$.
- $Br_6 = \{ (a_3, b_i, d_j, c_2) \mid b_i \in \text{dom}(B) \wedge d_j \in \text{dom}(D) \}$.

Now, let us consider the branch Br_1 . To identify whether $Br_1[A]$ is an irrelevant value in Br_1 , by Theorem 1, we have to check whether Br_1 is in conflict with any other branch with respect to attribute B in the decision tree. Since Br_1 is in conflict with Br_6 , by Theorem 1, $Br_1[A]$ is not an irrelevant value in Br_1 . By the same way, we find out that $Br_i[A]$ and $Br_i[B]$ are not irrelevant values in Br_i , where $i = 1 \dots 6$. Finally, let us consider the branch value $Br_i[D]$ in the ID3 tree. Since Br_4 is not in conflict with other branches with respect to attributes A and B , by Theorem 1, $Br_4[D]$, d_2 , is an irrelevant value in Br_4 . After removing the irrelevant values from the rules, the corresponding rules of this ID3 tree are:

- (1) $a_1 \wedge b_1 \rightarrow c_1$,
- (2) $a_1 \wedge b_2 \rightarrow c_2$,
- (3) $a_2 \wedge d_1 \rightarrow c_1$,
- (4) $a_2 \wedge b_1 \rightarrow c_1$,
- (5) $a_2 \wedge d_2 \wedge b_2 \rightarrow c_3$,
- (6) $a_3 \rightarrow c_2$. □

To identify all irrelevant values of a branch, we can recursively apply Theorem 1 until all branch values have been checked. As shown in example 2, however, the process of identifying irrelevant values by Theorem 1 is very time-consuming because we have to consider not only all the virtual branches of each branch in the decision tree, but also all combinations in the branches with respect to different attributes. Therefore, to enable users to focus on only relevant conditions of the classification rules, we have to provide a feasible method to solve the irrelevant values problem for a complex decision tree.

5. AN EFFICIENT ALGORITHM

Let $Br = (Br[A_1, \dots, A_{j-1}], Br[P], Br[A_{j+1}, \dots, A_k], C_i)$ be the branch of the decision tree, i.e., Br is through node P of the decision tree. According to Theorem 1, to identify whether a branch value $Br[P]$ is an irrelevant value, we have to check all the virtual branches of the decision tree. However, according to the semantics of the decision tree, when a branch Br' is not through P , the virtual branches of Br must not conflict with Br' because the decision tree creates a branch for each value of the non-leaf node attribute. As a result, identifying the virtual branches of a branch is similar to identify common branches or parts of branches of the decision tree. We prove this by Theorem 2.

Theorem 2. Let $Br = (Br[A_1, \dots, A_{j-1}], Br[P], Br[A_{j+1}, \dots, A_k], C_j)$ be a branch in a decision tree. For all other branches in the subtree of node P , when Br is not in conflict with these branches with respect to attributes A_{j+1}, \dots, A_k , the branch value $Br[P]$ is an irrelevant value in Br ; otherwise, $Br[P]$ is not an irrelevant value of the branch.

Proof: Let $Br = (Br[A_1, \dots, A_{j-1}], Br[P], Br[A_{j+1}, \dots, A_k], C_j)$. Then, for the branches, Br' , are not through P , Br must not conflict with these branches because the decision tree creates a branch for each value of the non-leaf attribute. That is, $\exists k', Br[A_{k'}] \neq Br[A_k]$, where $1 \leq k' \leq j-1$. Consequently, by Theorem 1, we do not need to consider these branches in the process to identify whether $Br[P]$ is an irrelevant value. Moreover, for all other branches in the subtree of node P , these branches, Br' , are through P , therefore, $Br[A_{k'}]$ must be equal to $Br[A_k]$, where $k = 1 \dots j-1$. Therefore, by Theorem 1, when Br is not in conflict with these branches with respect to attributes A_{j+1}, \dots, A_k , the branch value $Br[P]$ is an irrelevant value in the branch Br ; otherwise, $Br[P]$ is not an irrelevant value of the branch. \square

Moreover, after $Br[P]$ has been checking, all other branches in the subtree of node P are useless for the following process to identify the other irrelevant values of Br . Therefore, Theorem 2 still contains some redundant computations. We prove this by the following theorem.

Theorem 3. Let $Br = (Br[A_1, \dots, A_{j-1}], Br[P], Br[A_{j+1}, \dots, A_k], C_j)$ be a branch in a decision tree. When the branch value $Br[P]$ has been identified whether it is an irrelevant value by Theorem 2, all other branches in the subtree of P are useless for the following process to identify the irrelevant values of Br .

Proof: Let Br and Br' be two branches through P in the decision tree, respectively. When $Br[P]$ is an irrelevant values, Br must not conflict with Br' by Theorem 2. Moreover, when $Br[P]$ is not an irrelevant value, $Br[P]$ must not be equal to that of $Br'[P]$ because the decision tree creates a branch for each value of the non-leaf node attribute. Therefore, Br will never conflict with Br' . In other words, Br' is useless for the following computations to identify irrelevant values of Br . \square

According to Theorem 2, for each selected node P , to identify whether the branch value of node P is an irrelevant value of a branch, Br , we need only to consider the branches in the subtree of node P , and by Theorem 3, after checking whether the branch value of node P is an irrelevant value, all of these branches can be ignored for the following process to identify irrelevant values of Br . In other words, by Theorem 2 and Theorem 3, to identify all the irrelevant values of a branch, we need to check all the branches in the decision tree only once. Therefore, in the process of identifying the irrelevant values of a branch Br , we can assume that one of values $Br[P]$ is an irrelevant value and recursively applies Theorem 2 until the node P of the branch is the root of the decision tree. However, as indicated in Section 4, the process of identifying whether two branches are in conflict with each other is very time-consuming. Therefore, we have to provide an efficient method to solve this problem. The following theorem pro-

vides an algorithm to do this job so that the computation time of removing the irrelevant values from a branch can be reduced greatly.

Theorem 4. Let Br and Br' be two branches through node P in the decision tree, where $Br = (Br[A_1, \dots, A_{j-1}], Br[P], Br[A_j, \dots, A_{k1}], C_{i1})$ and $Br' = (Br'[A_1, \dots, A_{k2}], C_{i2})$. Let $A = \{A_j, \dots, A_{k1}\}$ and A_1 be the same attributes in these two branches, where $A_1 \subseteq A$. Then, Br is in conflict with Br' with respect to A iff $Br[A_1] = Br'[A_1]$ and $C_{i1} \neq C_{i2}$.

Proof: Let $A = \{A_j, \dots, A_{k1}\}$ and A_1 be same attributes in these two branches, where $A_1 \subseteq A$. When $Br[A_1] \neq Br'[A_1]$, it implies that $\exists A_{k'}, Br[A_{k'}] \neq Br'[A_{k'}]$, where $A_{k'} \in A_1$. Therefore, by Definition 2, these two branches will never be in conflict with each other with respect to A . Moreover, when two branches belong to the same class, by Definition 2, these two branches will also never conflict with each other. Therefore, we only consider the case $Br[A_1] = Br'[A_1]$ and $C_{i1} \neq C_{i2}$. By Theorem 2 and Definition 2, to identify whether Br is in conflict with Br' , we need only to consider whether $Br[A] \rightarrow C_{i1}$ is a part of rule implied by Br' . However, when $Br[A_1] = Br'[A_1]$, $Br[A] \rightarrow C_{i2}$ must be a part of rule implied by Br' . Therefore, Br is in conflict with Br' with respect to A iff $Br[A_1] = Br'[A_1]$ and $C_{i1} \neq C_{i2}$. \square

We have provided an efficient algorithm to identify whether two branches are in conflict with each other. Since we do not have to consider the virtual branches of each branch in the decision tree by Theorem 3, the time complexity of identifying irrelevant values of a branch is reduced greatly. The time complexity of identifying whether two branches are in conflict with other is reduced to $O(k)$, where k is the number of nodes of the branch. For each node P , to identify whether the branch value of node P is an irrelevant value of a branch, we need only to consider the branches in the subtree of node P once by Theorem 2 and 3. Therefore, the time complexity of the worst case of identifying all irrelevant values of a branch by Theorems 2, 3, and 4 is reduced to $O(km)$, where m is the number of branches in the tree. As a result, the time complexity of the algorithm of generating a set of rules without irrelevant condition for a decision tree is $O(km^2)$. The corresponding algorithm is shown in Fig. 5. The correctness of the algorithm is proved by the following theorem.

Theorem 5. Algorithm RGFDT can remove all irrelevant values from a decision tree.

Proof: In the algorithm, we apply Theorem 2, 3, and 4 to remove all irrelevant values from a branch Br' . Since the correctness of Theorem 2, 3, and 4 has been proved, all irrelevant values of Br' can be identified and removed. Moreover, since we repeat the same process to all branches of the tree, the algorithm can remove all irrelevant values. \square

Algorithm RGFDT (Rules Generation From the Decision Tree)

Input : A decision tree;

Output: A set of simplified rules without irrelevant condition;

Let $Br = \{Br_1, \dots, Br_m\}$; /* the branches of the decision tree */

For each branch Br' in Br **Do**

```

{
  Let  $Br' = (Br'[A_1], \dots, Br'[A_{kI}], C_i)$ ; /* to remove irrelevant values from the  $Br'$  */
   $Br_I := Br - Br'$ ;
  For each node  $P$  in  $Br'$  Do /*  $Br'[P] = Br'[A_j]$ , and for  $j = kI$  down to  $1$  */
  { (1) By Theorem 2, we need only to consider the branches in the subtree of node  $P$ .
    Therefore, let  $Br'_I$  be the branches in the subtree of node  $P$ , where  $Br'_I \in Br_I$ .
    (2) Apply Theorem 4 to check whether  $Br'$  is in conflict with any branch in  $Br'_I$ .
      If  $Br'[P]$  is an irrelevant value,
      Then remove  $Br'[P]$  from  $Br'$ ;
    (3) By Theorem 3, all branches in  $Br'_I$  are useless for the following process to
    identify the other irrelevant values of  $Br'$ . Therefore, let  $Br_I := Br_I - Br'_I$ ; }
  Represented  $Br'$  by a rule;
}

```

Fig. 5. An efficient algorithm to convert a decision tree to a set of rules without irrelevant condition.

Since irrelevant conditions are removed from the original rules, the resultant rules are more general than those represented by the decision tree. As a side effect, like $GID3$ and $GID3^*$, the resultant rules are less likely to suffer from missing branches. Next, we introduce an example to demonstrate that our algorithm can solve the missing branches problem to increase the accuracy of the resultant rules.

Example 3. Let us consider the decision tree in Fig. 6. Let the testing data e be (d_2, a_1, b_1) . The $ID3$ tree fails to classify e . The missing branches problem is due to the fact that some of the reduced subsets at the non-leaf nodes do not necessarily contain examples of every possible value of the branching attribute [2]. After converting the decision tree to rules by our algorithm, this testing data, e , can be classified. The resultant rules are given as:

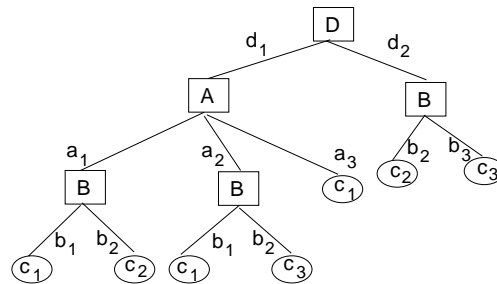


Fig. 6. A decision tree with irrelevant values.

1. $d_2 \wedge b_2 \rightarrow c_2$,
2. $d_2 \wedge b_3 \rightarrow c_3$,
3. $b_1 \rightarrow c_1$,
4. $d_1 \wedge a_3 \rightarrow c_1$,
5. $d_1 \wedge a_1 \wedge b_2 \rightarrow c_2$,
6. $d_1 \wedge a_2 \wedge b_2 \rightarrow c_3$.

□

Next, we introduce another example to demonstrate that our algorithm works correctly for the complex decision tree.

Example 4. Consider the decision tree in Fig. 7. This decision tree is produced by ID3 for the King-Knight-King-Rook chess end-game. For the same training set, the rule set produced by PRISM has 15 rules; therefore, J. Cendrowska claims that an expert system using the decision tree as its knowledge base would require significantly more tests to be performed [4]. However, when we apply our algorithm to this decision tree, the resultant rules are more simplified than those produced by PRISM. We remove the irrelevant value b_1 from rule 14 and 15 by our algorithm. The resultant rules produced by PRISM and our algorithm are listed in Table 1. Since the rule 16 is not included in the training set of PRISM, it will not generate this rule. Moreover, As pointed out in [4], both the decision tree, as shown in Fig. 7, and PRISM's rule set classify all 647 instances correctly. \square

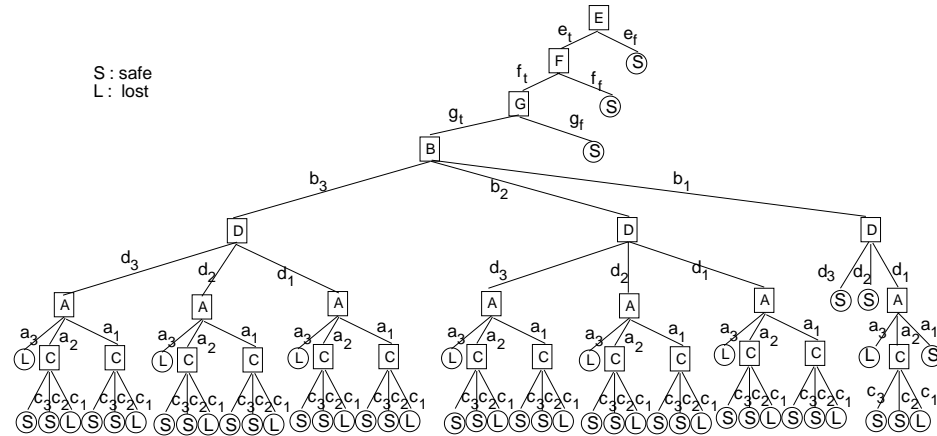


Fig. 7. The ID3 tree for the king-knight-king-rook chess end-game.

Table 1. The resultant rules produced by PRISM and our algorithm, RGFDT, respectively.

#	PRISM	Our algorithm
1	$e_f \rightarrow \text{safe}$	$e_f \rightarrow \text{safe}$
2	$f_f \rightarrow \text{safe}$	$f_f \rightarrow \text{safe}$
3	$g_f \rightarrow \text{safe}$	$g_f \rightarrow \text{safe}$
4	$b_1 \wedge d_2 \rightarrow \text{safe}$	$b_1 \wedge d_2 \rightarrow \text{safe}$
5	$b_1 \wedge d_3 \rightarrow \text{safe}$	$b_1 \wedge d_3 \rightarrow \text{safe}$
6	$a_1 \wedge c_2 \rightarrow \text{safe}$	$a_1 \wedge c_2 \rightarrow \text{safe}$
7	$a_1 \wedge c_3 \rightarrow \text{safe}$	$a_1 \wedge c_3 \rightarrow \text{safe}$
8	$a_2 \wedge c_2 \rightarrow \text{safe}$	$a_2 \wedge c_2 \rightarrow \text{safe}$
9	$a_2 \wedge c_3 \rightarrow \text{safe}$	$a_2 \wedge c_3 \rightarrow \text{safe}$
10	$e_f \wedge f_f \wedge g_f \wedge b_2 \wedge a_3 \rightarrow \text{lost}$	$e_f \wedge f_f \wedge g_f \wedge b_2 \wedge a_3 \rightarrow \text{lost}$
11	$e_f \wedge f_f \wedge g_f \wedge b_2 \wedge c_1 \rightarrow \text{lost}$	$e_f \wedge f_f \wedge g_f \wedge b_2 \wedge c_1 \rightarrow \text{lost}$
12	$e_f \wedge f_f \wedge g_f \wedge b_3 \wedge a_3 \rightarrow \text{lost}$	$e_f \wedge f_f \wedge g_f \wedge b_3 \wedge a_3 \rightarrow \text{lost}$
13	$e_f \wedge f_f \wedge g_f \wedge b_3 \wedge c_1 \rightarrow \text{lost}$	$e_f \wedge f_f \wedge g_f \wedge b_3 \wedge c_1 \rightarrow \text{lost}$
14	$e_f \wedge f_f \wedge g_f \wedge b_1 \wedge d_1 \wedge a_3 \rightarrow \text{lost}$	$e_f \wedge f_f \wedge g_f \wedge d_1 \wedge a_3 \rightarrow \text{lost}$
15	$e_f \wedge f_f \wedge g_f \wedge b_1 \wedge d_1 \wedge a_2 \wedge c_1 \rightarrow \text{lost}$	$e_f \wedge f_f \wedge g_f \wedge d_1 \wedge a_2 \wedge c_1 \rightarrow \text{lost}$
16		$b_1 \wedge a_1 \rightarrow \text{safe}$

6. EMPIRICAL EVALUATION RESULTS

In this section, we conduct experiments to compare the performance of ID3 with our algorithm for the ID3 tree on some real-world databases that are from the University of California, Irvine (UCI) [10]. The ID3 and our algorithm are implemented on a Pentium-200 PC with 64MB RAM, running on Windows/NT.

The data sets we used are described in Table 2. WBCD is a breast cancer database that was obtained from the university of Wisconsin Hospitals. Samples arrive periodically as Dr. Wolberg reports his clinical cases [9]. CAR is a car evaluation database, which was derived from a simple hierarchical decision model and originally developed for the demonstration of DEX [1]. BSWDD is a balance scale weight and distance database, which was generated to model psychological experiment result [16]. VOTE is voting records for each of the U. S. House of Representatives Congressmen on the 16 key votes identified by the Congressional Quarterly Inc. Washington, D.C. 1985 [15]. NURSERY database is derived from a hierarchical decision model originally developed to rank applications for nursery schools. NURSERY was used in 1980's when there was excessive enrollment at these schools in Ljubljana, Slovenia, and the rejected applications frequently needed an objective explanation.

There is random choice of one-third of each database as training set to induce a decision tree and the remaining data as a test set for its accuracy. Each database is executed for five runs and then the average of the resultant data will come up with the test data in Table 3. The data induces the running time, average error rates of the testing data, and the number of rules of ID3 and our algorithm RGFDT. We may discover that the time taken to remove irrelevant values only counts as a minor portion of that taken to build a decision tree. Since all irrelevant values are removed, the resultant rules are more general than those represented by the decision tree. As a side effect, the resultant rules are less likely to suffer from missing branches. Actually, its accuracy conspicuously increases and its error rate decreases at less 20%. Moreover, there is at least 10% decrease in the number of production rules.

Table 2. Details of the data sets in empirical evaluation.

Data Set	Examples		Attribute	Class
	Training set	Test set		
WBCD	233	450	10	2
CAR	576	1152	6	4
BSWDD	208	417	4	3
VOTE	135	300	16	2
NURSERY	300	700	8	4

Table 3. The empirical evaluation results.

Data Set	Running Time (second)		Error Rate (%)		Rules	
	ID3	RGFDT	ID3	RGFDT	ID3	RGFDT
WDBC	4.34375	0.15625	12.7	10	40	34.2
CAR	11.10156	1.8125	12.8	5.7	75.6	68.4
BSWDD	7.52344	1.47656	43.8	9.2	82.8	65
VOTE	2.31871	0.92012	6.3	2.7	25.6	22.4
NURSERY	8.78023	2.0123	50.4	34.2	61.8	50.6

7. CONCLUSIONS

To overcome the irrelevant values problem, we have proposed a new algorithm to solve it. We eliminate irrelevant values during converting the decision tree to rules by the semantics of the decision tree. Therefore, the irrelevant conditions will not appear in the resultant classification rules. The time complexity of the algorithm of generating a set of rules without irrelevant condition for a decision tree is $O(km^2)$, where k is the number of nodes of the branch and m is the number of branches in the tree.

As the empirical evaluation result indicated in section 6. The results present evidences to prove that our algorithm overcomes not only the irrelevant values problem but also some missing branches problem with negligible extra computational cost concerning that of constructing the decision tree by ID3. Since our algorithm depends only on the semantics of the decision tree, our algorithm can be integrated into any existing decision tree system easily.

REFERENCES

1. M. Bohanec and V. Rajkovic, "Expert system for the decision making," *Sistemica*, Vol. 1, No. 1, 1990, pp. 145-157.
2. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth, 1984.
3. Y. Cai, N. Gercone, and J. Han, "An attribute-oriented approach for learning classification rules from relational databases," in *Proceedings of Conference on Data Engineering*, 1990, pp. 281-288.
4. J. Cendrowska, "PRISM: an algorithm for inducing modular rules," *International Journal of Man-Machine Studies*, Vol. 27, No. 4, 1988, 349-370.
5. J. Cheng, U. M. Fayyad, K. B. Irani, and Z. Qian, "Improved decision trees: a generalized version of ID3," in *Proceedings of the Fifth International Conference on Machine Learning*, 1988, pp. 100-108.
6. U. M. Fayyad and K. B. Irani, "A machine learning algorithm (GID3*) for automated knowledge acquisition improvements and extensions," *General Motor Research Report CS-634*, Warren, MI: GM research labs, 1991.
7. U. M. Fayyad, "Branching on attribute values in decision tree generalization," in *Proceedings of Twelfth National Conference on Artificial Intelligence, AAAI-94*, 1994, pp. 104-110.
8. M. Kamber, L. Winstone, W. Gong, S. Cheng, and J. Han, "Generalization and decision tree induction: Efficient classification in data mining," in *Proceedings of Seventh International Workshop on Research Issue in Data Engineering*, 1997, pp. 111-121.
9. O. L. Mangasarian and W. H. Wolberg, "Cancer diagnosis via linear programming," *SIAM News*, Vol. 23, No. 25, 1990, pp. 1-18.
10. P. M. Murphy and D. W. Aha, "UCI repository of machine learning databases," <http://www.ics.edu/~mlaern>.
11. J. R. Quinlan, "Learning efficient classification procedures and their application to chess end games," in *Machine Learning: An Artificial Intelligence Approach*, Michalski, Carbonell and Mitchell. eds. Morgan Kaufmann, 1983, pp. 463-482.
12. J. R. Quinlan, "Induction of decision tree," *Machine Learning*, Vol. 1, No. 1, 1986, pp. 81-106.
13. J. R. Quinlan, "Simplifying decision trees," *International Journal of Man-Machine Studies*, Vol. 27, No. 3, 1987, pp. 221-234.

14. J. R. Quinlan, *C4.5: Program for Machine Learning*, Morgan Kaufmann, 1993.
15. J. C. Schlimmer and D. Fisher, "A case study of incremental concept induction," in *Proceedings of Fifth International Conference on Artificial Intelligence*, 1986, pp. 496-501.
16. T. Shultz, D. Mareschal, and W. Schmit, "Modeling cognitive development on balance scale phenomena," *Machine Learning*, Vol. 16, No. 1, 1994, pp. 59-88.
17. P. E. Utgoff, "ID5: an incremental ID3," in *Proceedings of Fifth International Conference on Machine Learning*, 1988, pp. 107-120.
18. P. E. Utgoff, "ID5: Incremental induction of decision trees," *Machine Learning*, Vol. 4, No. 2, 1989, pp. 161-186.

Ding-An Chiang (蔣定安) is a professor of the Department of Information Engineering at Tamkang University. He received B.S. degree in Hydraulic Engineering from Chung Yuan Christian University, Taiwan, in 1981, and the M.S. and Ph.D. degrees in Computer Science from the University of Southwestern Louisiana in 1986 and 1990, respectively. His current research interests include fuzzy relational databases and data mining.

Wei Chen (陳偉) received B. S. degree of mathematics at Soochow University in 1979 and M. S. degree of mathematics at Tamkang University in 1981. In 1998, he received Ph.D. degree of Information Engineering at Tamkang University. He is dean in National Taipei College of Nursing, Taipei, Taiwan and his research interests are in database, machine learning, multimedia, and data mining.

Yi-Fan Wang (王亦凡) received his B.S. degree in computer science, M.S. and Ph.D. degrees in information engineering, all from Tamkang University in Taipei, Taiwan, in 1991, 1995 and 1998, respectively. He is currently an associate professor in Chang Gung Institute of Nursing. His main research interests include artificial intelligence, computer graphic, knowledge discovery in databases. He is also a member of the IEEE and the ACM.

Lain-Jinn Hwang (黃連進) received B.S. and M.S. degrees in computer science and management information system from Tamkang University, Taipei, Taiwan, Republic of China, in 1977 and 1979, respectively. He is completing the Ph.D. degree in computer science and information engineering at TKU. His research interests include image/video processing/compression, computer architecture.