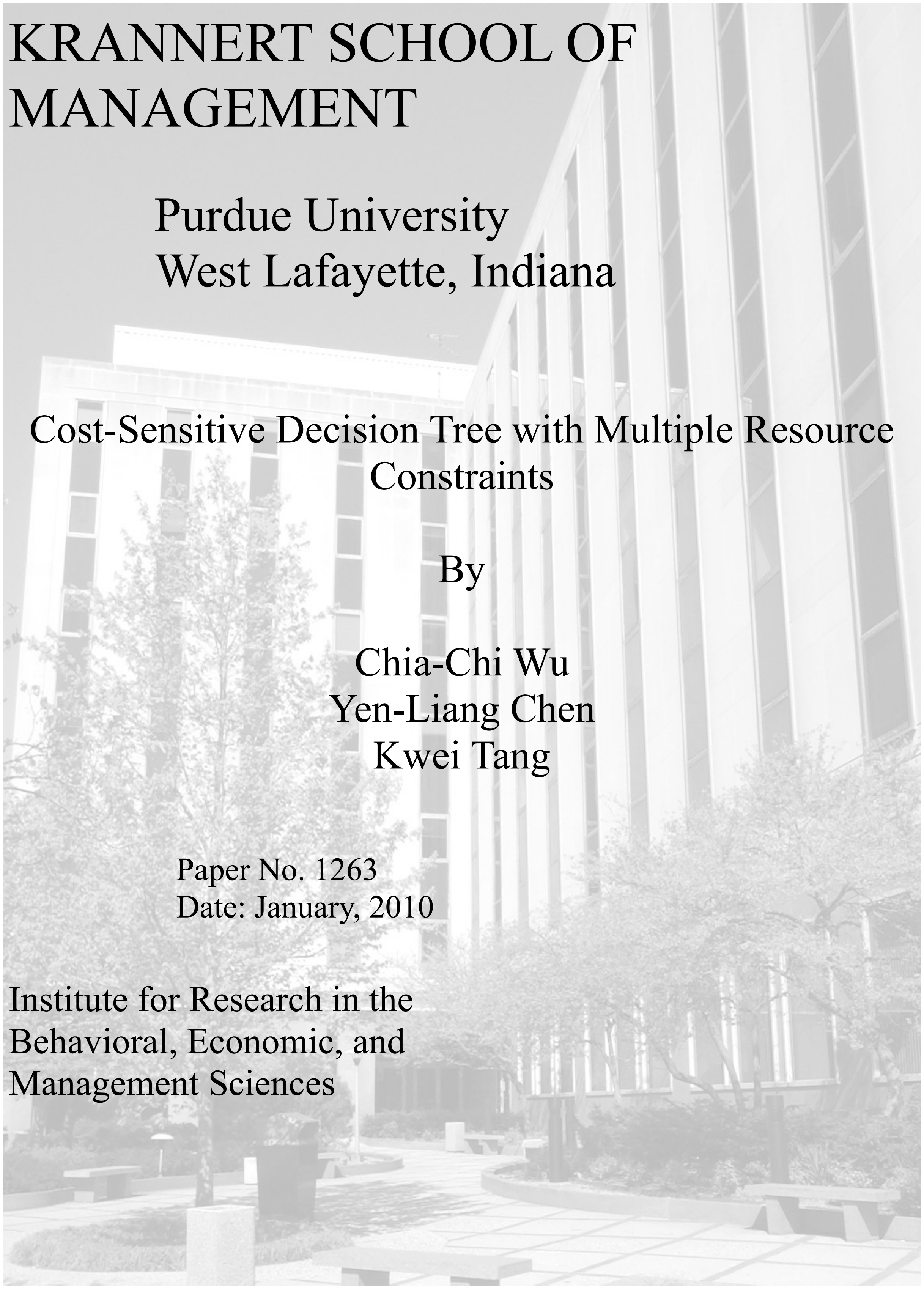


Cost-Sensitive Decision Tree with Multiple Resource Constraints

By

Chia-Chi Wu
Yen-Liang Chen
Kwei Tang

Paper No. 1263
Date: January, 2010

Cost-Sensitive Decision Tree with Multiple Resource Constraints

Chia-Chi WU, Yen-Liang CHEN, and Kwei TANG

Abstract—**Resource constraints are commonly found in classification tasks. For example, there could be a budget limit on implementation and a deadline for finishing the classification task. Applying the top-down approach for tree induction in this situation may have significant drawbacks. In particular, it is difficult, especially in an early stage of tree induction, to assess an attribute's contribution to improving the total implementation cost and its impact on attribute selection in later stages because of the deadline constraint. To address this problem, we propose an innovative algorithm, namely, the Cost-Sensitive Associative Tree (CAT) algorithm. Essentially, the algorithm first extracts and retains association classification rules from the training data which satisfy resource constraints, and then uses the rules to construct the final decision tree. The approach has advantages over the traditional top-down approach, first because only feasible classification rules are considered in the tree induction and, second, because their costs and resource use are known. In contrast, in the top-down approach, the information is not available for selecting splitting attributes. The experiment results show that the CAT algorithm significantly outperforms the top-down approach and adapts very well to available resources.**

Index Terms—**Cost-sensitive learning, mining methods and algorithms, decision trees**

I. Introduction

Resource constraints are often imposed on a classification task. In medical diagnosis and marketing campaigns, for example, it is common to have a deadline and budget for finishing the task. The objective

C.-C. Wu is with the Department of Information Management, National Central University, Chung-Li, Taiwan 320, Republic of China. E-mail: 944403006@cc.ncu.edu.tw. Tel: +886-3-4227151-66569. Fax: +886-3-4254604.

Y.-L. Chen is with the Department of Information Management, National Central University, Chung-Li, Taiwan 320, Republic of China. E-mail: ylchen@mgt.ncu.edu.tw. Tel: +886-3-4267266. Fax: +886-3-4254604.

K. Tang is with the Krannert Graduate School of Management, Purdue University, West Lafayette, IN 47907, USA. E-mail: ktang@purdue.edu. Tel: +1-765-494-4464, Fax: +1-765-494-9658

All correspondence should be addressed to Prof. Kwei Tang.

of this paper is to develop an algorithm for tree induction when costs and multiple resource constraints are explicitly considered in developing a decision tree. To our best knowledge, the problem has not been studied in the literature.

Decision trees are one of the most popular predictive models in data mining. The advantages of using a decision tree has been well documented, including its ability to handle high dimensional data, its simple structure in communicating with users, and the fact that its classification results are accurate in many application areas [9]. Many well-known tree induction algorithms have been developed, such as: ID3 [15], C4.5 [16]. CART [3], and GATree [13]. These algorithms are based on a top-down approach, which recursively selects attributes to partition the data to form a hierarchical tree structure with a goal of maximizing classification accuracy or minimizing classification error.

Recently, researchers have discussed economic factors in the process of developing decision trees, such as the costs of procuring, preparing, and storing training datasets and the computational costs of generating classification rules [23]. The research field, known "cost-sensitive decision trees," explicitly aims to minimizing the total cost of using a tree in classification [20]. Two cost components are often considered: the cost of using attributes in classification, and the cost or penalty incurred by misclassification. For example, in medical diagnosis, attributes may represent the results of diagnosis procedures. If an attribute appears on the decision tree, the cost of performing the diagnosis procedure is incurred for all cases going through the corresponding node [21]. Furthermore, the misclassification cost is the economical consequences associated with classification errors [1]. Studies have concentrated solely on attribute (test) costs [4][12][18], the misclassification cost [6][7][19][27], or the sum of the two costs [10][11][20][24][26].

Several researchers have considered one resource constraint for a cost-sensitive decision tree. Yang, Ling, Chai, and Pan [24] assumed that patients have a limited budget for performing medical tests and that the tree traveling stops when the limit is reached. Qin, Zhang, and Zhang [14] defined a general problem with one target and one resource component. A top-down algorithm is developed using the ratio of the gain in the target and the resource utilization as the splitting criteria. Chen, Wu, and Tang [4] considered a situation, where multiple targets exist for a classification task. Since targets may share common predictors, it may be beneficial to develop one decision tree with multiple targets, instead of separate trees for individual targets. A top-down approach was used to develop an algorithm for tree induction under a constraint limiting the cost of classifying an instance.

It is well known that the top-down approach is a greedy method, which selects attributes sequentially without back tracking. In other words, the method uses a heuristic attribute selection criterion, and a selected attribute cannot be removed from the tree structure in later stages. Both aspects may result in major drawbacks for using the approach in the situation under consideration. As applied in solving unconstrained problems, it is difficult to assess an attribute's contribution to improving the goal of the classification task during the tree-induction process. Furthermore, it is especially difficult to evaluate the impact an attribute selected in an early stage on attribute selection in later stages because of the resource constraints. For example, when a medical test is considered in medical diagnosis, the time for performing the test will limit other tests that can be used when a deadline has been specified for the diagnosis procedure. Note that when one constraint is considered, it is possible to use the cost reduction per unit resource used as a reasonable criterion for selecting a splitting attribute. When multiple constraints are considered, it is difficult to define a reasonable criterion. This further makes the top-down approach even less attractive or effective in this situation.

The drawbacks of the top-down approach motivate the development of an innovative algorithm, as illustrated in Fig. 1. Instead of deriving a decision tree directly from the training dataset, our algorithm consists of two steps. In the first step, we extract association classification rules from the training dataset first and retain those that satisfy the resource constraints. In the second step, we use the association classification rules obtained in the first step to develop a cost-sensitive decision tree.

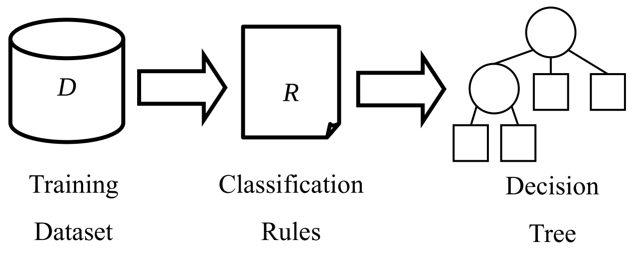


Fig. 1. The concept of our approach

The algorithm has two main advantages:

(1) Since only feasible classification rules are retained in the first step, we can focus on cost minimization without considering the resource constraints in step 2.

(2) All feasible classification rules are considered in tree induction, whereas the top-down approach considers only those resulting from node splitting.

The remainder of this paper is organized as follows. In Section II, we formalize the problem under consideration, and, in Section III, present the proposed algorithm, namely the Cost-Sensitive Associative Tree (CAT) algorithm. Numerical evaluations of the algorithm are presented in Section IV. A summary and discussion are given in Section V.

II. Problem Definition

A decision tree is built with a training dataset that is usually represented as a relational table shown in Table I. We assume the dataset does not contain missing values. Let $A = \{A_1, A_2, \ldots, A_m\}$ be the set of all attributes in this table, and $C = \{c_1, c_2, \ldots, c_n\}$ the set of all classes. Each tuple in the table is a record consisting of attribute values and its corresponding class. We use d_k to denote the k^{th} record, $a_x(d_k)$ the value of A_x of d_k, and $c(d_k)$ the class of d_k.

TABLE I.
An Example of Training Data

ID	A_1	A_2	A_3	Class
1	A	A	A	T
2	A	B	B	F
3	A	A	A	T
4	B	B	B	T
5	B	A	A	F
6	B	A	B	F
7	C	A	A	F
8	C	B	B	F
9	B	A	A	T
10	A	A	A	T

An *item* is denoted by (A_x, a_{x_i}), where A_x is an attribute and a_{x_i} is a possible value of A_x. For example: (A_1, A) is an item. An *itemset* is a set of *item*s. An *itemset* is a set of *item*s. {(A_1, A), (A_3, B)} is an example of *itemset*. A record d_k can be represented as ($att(d_k)$, $c(d_k)$), which is a pair of an *itemset* and a class, where $att(d_k) = \{(A_1, a_1(d_k)), (A_2, a_2(d_k)), \ldots, (A_m, a_m(d_k))\}$, is the set of all attribute-value pairs of d_k. For example, the record with ID 1 can be represented as ({(A_1, A), (A_2, A), (A_3, A)}, T).

A decision tree *TR* is a directed acyclic graph. Fig. 2 is an example of a decision tree. We use a circle to represent an internal node and a square to represent a leaf node. We use n_i to denote a node in a decision tree, and n_0 is the root. An internal node n_i in a decision tree is associated with an attribute $s(n_i)$ which is used to split n_i, while a leaf node n_j in a decision tree is labeled with a class $l(n_j)$. For example, n_0 is an internal node with $s(n_0)$ = "A_1." n_{011} is a leaf node with $l(n_{011})$ = "T." $<n_i, n_j>$ is an edge that links node n_i with node n_j, where n_i is the parent node of n_j. $v<n_i, n_j>$, which is a possible value of $s(n_i)$, the value assigned on edge $<n_i, n_j>$. For example, $<n_0, n_{01}>$ is an edge that links node n_0 with n_{01}, and $v< n_0, n_{01}>$ = "A" is the value along this edge, which is a possible value of attribute A_1.

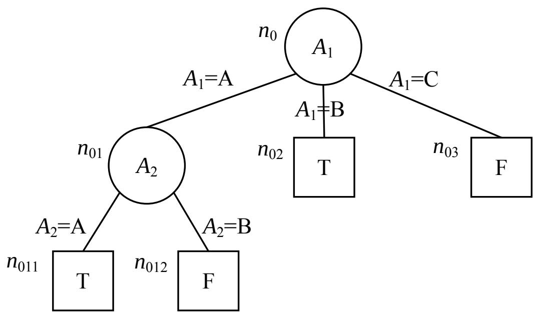
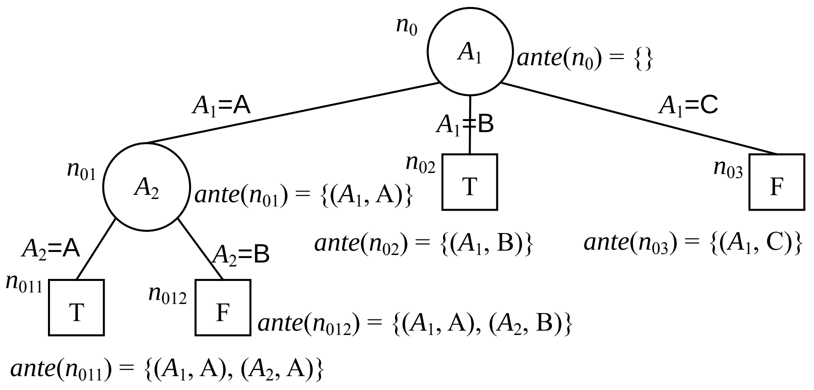


Fig. 2. An example of decision tree

We define $ante(n_i)$ as the antecedents of node n_i. Suppose the sequence of nodes between the root and n_i are $n_1, n_2, n_3, \ldots, n_m$, and the values on edges $<n_0, n_1>, <n_1, n_2>, \ldots, <n_m, n_i>$ are $v<n_0, n_1>, v<n_1, n_2>, \ldots, v<n_m, n_i>$ respectively, $ante(n_i) = \{(s(n_0), v<n_0, n_1>), (s(n_1), v<n_1, n_2>), \ldots, (s(n_m), <n_m, n_i>)\}$. Fig. 3 shows the antecedents of the nodes in our example decision tree. A record travels the decision tree according to the attributes in the internal nodes and the values on the edges. A record d_k can arrive n_i if and only if $ante(n_i) \subseteq att(d_k)$. We use D_{n_i} to denote a subset of *D*, where a record d_k belongs to D_{n_i} if and only if d_k can arrive node n_i. That is, $D_{n_i} = \{d_k \mid d_k \in D, \text{and } ante(n_i) \subseteq att(d_k)\}$.



n_0
A_1
$ante(n_0) = \{\}$
A_1=A
A_1=B
A_1=C
n_{01}
A_2
$ante(n_{01}) = \{(A_1, A)\}$
n_{02}
T
n_{03}
F
$ante(n_{02}) = \{(A_1, B)\}$
$ante(n_{03}) = \{(A_1, C)\}$
A_2=A
A_2=B
n_{011}
T
n_{012}
F
$ante(n_{012}) = \{(A_1, A), (A_2, B)\}$
$ante(n_{011}) = \{(A_1, A), (A_2, A)\}$

Fig. 3. Antecedents of nodes in decision tree

In most of the related studies, a misclassification cost is defined as the cost incurred by assigning a case to class i when it actually belongs to class j. Suppose there are m distinct classes, the misclassification costs can be represented as an $m \times m$ matrix. We use $MisCost(i, j)$ to denote the misclassification cost of assigning a case of class j to class i. Consider the example dataset in Table I. There are two classes, "T" and "F", and the misclassification cost matrix is a 2×2 matrix shown in Table II, where $MisCost(T, F) = 10$, $MisCost(F, T) = 20$, and $MisCost(T, T) = MisCost(F, F) = 0$.

TABLE II.
An Example of Misclassification Cost Matrix

Class	T	F
T	0	10
F	20	0

When we designate a node as a leaf node and assign a label, the misclassification cost at the node is determined. Let $NodeCost(n_i)$ denote the misclassification cost of assigning n_i as a leaf node, which is determined using the label that gives the lowest misclassification cost; i.e.,

$$NodeCost(n_i) = \min_{\forall l_j}\left(\sum_{\forall d_k \in D_{n_i}} MisCost(l_j, c(d_k))\right).$$

For example, in Fig. 2, n_0 can be labeled with "T" or "F." If we label n_0 with "T," the corresponding misclassification cost would be 50. In contrast, the cost would be 100 if "F" is assigned. In this case, we have $NodeCost(n_0) = 50$.

Measuring an attribute may consume several types of resources, such as cost and time. We use $ResCon_y(A_x)$ to denote the number of units of the yth resource needed to measure attribute A_x. For example, the resource consumptions of the attributes in Table I are listed in Table III.

TABLE III.
The Example of Resource Consumptions

Attribute	$ResCon_1(A_x)$	$ResCon_2(A_x)$
A_1	6	5
A_2	4	2
A_3	2	10

When a record arrives at a node from the root, we must accumulate the number of units of resources consumed to reach the node. We use $TotalCon_y(n_i)$ to denote the total consumption of the yth resource for moving a record from the root node to node n_i. Suppose A_1, A_2, …, and A_m are the attributes appearing in $ante(n_i)$, respectively, $TotalCon_y(n_i) = \sum_{x=1 \text{ to } m} ResCon_y(A_x)$. For example, in Fig. 1, we have $TotalCon_1(n_{02}) = 6$ and $TotalCon_2(n_{011}) = 7$.

We use $MaxResCon_y$ to denote the maximum consumption of the yth resource to complete the classification task for a record. For any leaf node n_i in a decision tree, the total consumption of any resource $TotalCon_y(n_i)$ cannot exceed $MaxResCon_y$.

Given a training dataset, our goal is to develop a cost-sensitive tree that minimizes the total misclassification cost under multiple resource constraints. The proposed algorithm is given in the next section.

III. ALGORITHM

There are three phases in the CAT algorithm:

Phase 1: Extract all classification rules satisfying the resource constraints from the training dataset

Phase 2: Build a decision tree from the classification rules extracted in phase 1

Phase 3: Adjust the result of phase 2 to produce the final decision tree

A classification rule *r* can be represented in form of: $condition(r) \rightarrow consequent(r)$. The condition of this rule is an *itemset*, and the consequent is a class. For example, $\{(A_1, \mathrm{C}), (A_2, \mathrm{B}), (A_3, \mathrm{A})\} \rightarrow \mathrm{T}$ is a classification rule. Semantically, the condition of this rule is a conjunction of *item*s. The rule above can be interpreted as $(A_1, \mathrm{C}) \wedge (A_2, \mathrm{B}) \wedge (A_3, \mathrm{A}) \rightarrow \mathrm{T}$. We use D_r to denote the set of records covered by rule *r*. That is, a record d_k belongs to D_r if and only if $condition(r) \subseteq att(d_k)$. For example, $D_{((A_1,A)\rightarrow \mathrm{T})} = \{d_1, d_2, d_3, d_{10}\} = D_{((A_1,A)\rightarrow \mathrm{F})}$. We define $cov(r) = |D_r|$, the *coverage* of rule *r*, which is the number of records in *D* covered by rule *r* or matching *condition*(*r*).

The total consumption of resource *y* associated with rule *r* is denoted as $RuleCon_y(r)$. Suppose A_1, A_2, …, and A_m are the attributes appearing in *condition*(*r*), $RuleCon_y(r) = \sum_{x=1 \text{ to } m} ResCon_y(A_x)$. For example, $RuleCon_1((A_1,\mathrm{C}) \wedge (A_2,\mathrm{B}) \wedge (A_3,\mathrm{A}) \rightarrow \mathrm{T}) = 12$.

We use *RuleCost*(*r*) to denote the total misclassification cost of applying rule *r* to the records in D_r, which is determined by:

$$RuleCost(r) = \sum_{\forall d_k \in D_r} MisCost(consequent(r), c(d_k)).$$

We demonstrate our algorithm using the training dataset in Table I, the misclassification cost matrix in Table II, and the resource consumption matrix shown in Table III. Both thresholds of resource 1 and resource 2 are set to 10.

III.1 Phase 1: Extract classification rules

In this phase, we first extract all classification rules and then prune those rules in rule set R_{pruned}, where a rule r is in R_{pruned} if $cov(r)$ is smaller than a threshold *mincov* or $RuleCon_y(r) > MaxResCon_y$ for any resource. We prune rule $r \in R_{pruned}$ because each rule r corresponds to a node n_r in decision tree, where:

- $condition(r) = ante(n_r)$
- $consequent(r) = l(n_r)$, suppose that n_r is a leaf node
- $RuleCon_y(r) = TotalCon_y(n_r)$
- $Cov(r) = \left|D_{n_r}\right|$

When $cov(r) < mincov$, indicating only a small number of records support this rule, the rule is discarded to avoid a potential over-fitting problem. Furthermore, when $RuleCon_y(r) > MaxResCon_y$, the rule is not feasible because the resource constraint associated with the y^{th} resource is violated.

To extract the classification rules, we first transform the relational database D into transaction database $D_{Itemset}$ and then find all frequent patterns in $D_{Itemset}$. We use an apriori-like algorithm and find all the frequent itemsets in $D_{Itemset}$ which is shown in Table IV. For all records d_k belong to D, $att(d_k) \in D_{Itemset}$.

TABLE IV.
A $D_{Itemset}$ Transformed from Table I.

1	(A_1, A), (A_2, A), (A_3, A)
2	(A_1, A), (A_2, B), (A_3, B)
3	(A_1, A), (A_2, A), (A_3, A)
4	(A_1, B), (A_2, B), (A_3, B)
5	(A_1, B), (A_2, A), (A_3, A)
6	(A_1, B), (A_2, A), (A_3, B)
7	(A_1, C), (A_2, A), (A_3, A)
8	(A_1, C), (A_2, B), (A_3, B)
9	(A_1, B), (A_2, A), (A_3, A)
10	(A_1, A), (A_2, A), (A_3, A)

For each frequent itemset, we select the class that minimizes the misclassification cost as the consequent of the generated rule. For example, consider the frequent itemset (A_2, A). $RuleCost((A_2, \mathrm{A}) \rightarrow \mathrm{T}) = 30$, while $RuleCost((A_2, \mathrm{A}) \rightarrow \mathrm{F}) = 80$. Therefore, we generate Rule $(A_2, \mathrm{A}) \rightarrow \mathrm{T}$ from frequent itemset (A_2, A).

After phase 1, we obtain *R*, which is the pruned classification rules set extracted from the training dataset. The rules are ranked mainly in terms of the average misclassification cost. Given two rules r_i and r_j in *R*, r_i precedes r_j if

- $\dfrac{RuleCost(r_i)}{cov(r_i)} < \dfrac{RuleCost(r_j)}{cov(r_j)}$
- $\dfrac{RuleCost(r_i)}{cov(r_i)} = \dfrac{RuleCost(r_j)}{cov(r_j)} \wedge cov(r_i) > cov(r_j)$
- $\dfrac{RuleCost(r_i)}{cov(r_i)} = \dfrac{RuleCost(r_j)}{cov(r_j)} \wedge cov(r_i) = cov(r_j)$, but there are less items in the antecedent of r_i than those of r_j.
- All the criteria above are tied, but r_i was generated earlier than r_j**.**

After sorting, the rules extracted in our example are listed in Fig. 4.

{A_1 = A, A_2 = A}→T, *coverage*: 3, *misclassification cost*: 0, *average cost*: 0
{A_1 = C}→F, *coverage*: 2, *misclassification cost*: 0, *average cost*: 0
{A_1 = A, A_2 = B}→F, *coverage*: 1, *misclassification cost*: 0, *average cost*: 0
{A_1 = B, A_2 = B}→T, *coverage*: 1, *misclassification cost*: 0, *average cost*: 0
{A_1 = C, A_2 = A}→F, *coverage*: 1, *misclassification cost*: 0, *average cost*: 0
{A_1 = C, A_2 = B}→F, *coverage*: 1, *misclassification cost*: 0, *average cost*: 0
{A_1 = A}→T, *coverage*: 4, *misclassification cost*: 10, *average cost*: 2.5
{A_3 = A}→T, *coverage*: 6, *misclassification cost*: 20, *average cost*: 3.3
{A_2 = A}→T, *coverage*: 7, *misclassification cost*: 30, *average cost*: 4.3
{A_1 = B}→T, *coverage*: 4, *misclassification cost*: 20, *average cost*: 5
{A_3 = B}→F, *coverage*: 4, *misclassification cost*: 20, *average cost*: 5
{A_2 = B}→T, *coverage*: 3, *misclassification cost*: 20, *average cost*: 6.7
{A_1 = B, A_2 = A}→T, *coverage*: 3, *misclassification cost*: 20, *average cost*: 6.7

Fig. 4. The extracted rules

III.2 Phase 2: Build a decision tree with extracted rules

In this phase, we build a decision tree from the rules extracted in phase 1. An overview of this phase is shown in Fig. 5.

```
1. Starting with a single node, root
2. For each non-leaf node, $n_i$
   2.1. Make two estimates for each attribute.
   2.2. Select the best splitting attribute according to splitting criteria.
   2.3. If terminal condition is satisfied, stop splitting and assign $n_i$ as a leaf node.
        Else, split $n_i$ with the splitting attribute of $n_i$.
```

Fig. 5. An overview of phase 2

Phase 2 starts with a single root marked as n_0, the whole training dataset D and R, which is the set of all the rules extracted in phase 1.

If an attribute A_x was selected to split n_i, we use only the rules that contain A_x to classify the records. In other words, rules that exclude A_x have to be discarded. However, since we select the splitting attribute by estimation, the result may not be globally optimal. Some valuable rules may be discarded because of the estimate error. Therefore, when splitting a node with an attribute A_x, besides the original branches related to each value of A_x, we insert an additional branch marked "*unused*" to reserve the rules that exclude splitting attribute A_x. Fig. 6 shows the branches and corresponding rule sets after splitting the root with attribute A_1.

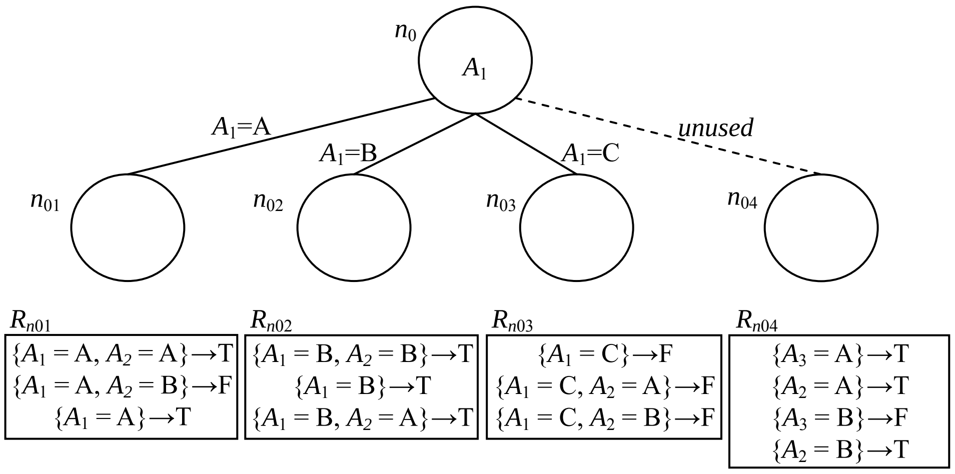


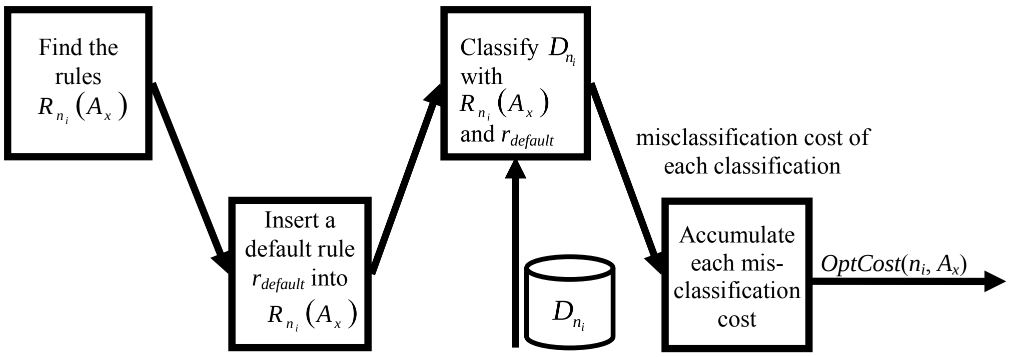
Fig. 6. Result of splitting the root node using attribute A_1

We use EXA_{n_i} to denote the set of attributes that are excluded from n_i. For example, $EXA_{n_{04}} = \{A_1\}$.

We use R_{n_i} to denote a subset of *R*, where rule r_j belongs to R_{n_i} if and only if $ante(n_i) \subseteq condition(r_j)$ and any attribute in EXA_{n_i} is not in *condition*(r_j). We use $R_{n_i}(A_x)$ to denote a subset of R_{n_i}. A rule r_j is in $R_{n_i}(A_x)$ if r_j is in R_{n_i} and A_x is in *condition*(r_j). For example, $R_{n_{04}}(A_3)$ contains two rules: {A_3=A}→T and {A_3=B}→F.

For each non-leaf node n_i, we use two measurements to estimate the effect of using an attribute A_x to split n_i. The first is *OptCost*(n_i, A_x), which is the misclassification cost of classifying the records in D_{n_i} with the rules in $R_{n_i}(A_x)$. Since not all rules in $R_{n_i}(A_x)$ will be retained in the final decision tree, *OptCost*(n_i, A_x) can be regarded as an optimistic estimate of using A_x in n_i. The second is *PesCost*(n_i, A_x), which is the sum of all misclassification costs of the child nodes generated by splitting n_i with A_x. *PesCost*(n_i, A_x) is obtained by supposing that no other attribute will be measured after A_x. Therefore *PesCost*(n_i, A_x) can be regarded as a pessimistic estimate of using A_x in n_i.

The process of measuring *OptCost*(n_i, A_x) is shown in Fig. 7.



Find the rules $R_{n_i}(A_x)$
Insert a default rule $r_{default}$ into $R_{n_i}(A_x)$
Classify D_{n_i} with $R_{n_i}(A_x)$ and $r_{default}$
D_{n_i}
misclassification cost of each classification
Accumulate each mis-classification cost
OptCost(n_i, A_x)

Fig. 7. The process of measuring *OptCost*(n_i, A_x)

We first insert a default rule $r_{default}$ into the bottom of $R_{n_i}(A_x)$, where the condition of $r_{default}$ is the antecedent of n_i, and the consequent of it is the class that minimizes the misclassification cost. Then, we classify the records in D_{n_i} with the rules in $R_{n_i}(A_x)$ and the default rule. For a record d_k in D_{n_i}, the first rule that covers the attribute values of d_k classifies it. The classification result may cause a misclassification cost. The accumulation of all the misclassification costs is the value of *OptCost*(n_i, A_x).

An example is shown in Fig. 8. For measuring *OptCost*(n_0, A_1), we use the rules in R_{n_0} with A_1 in their conditions and the default rule, $\{\} \rightarrow \mathrm{T}$, to classify the records belonging to D_{n_0}. The value of *OptCost*(n_0, A_1) is the accumulation of all misclassification costs of the classifications, and the result is

OptCost(n_0, A_1) = 20.

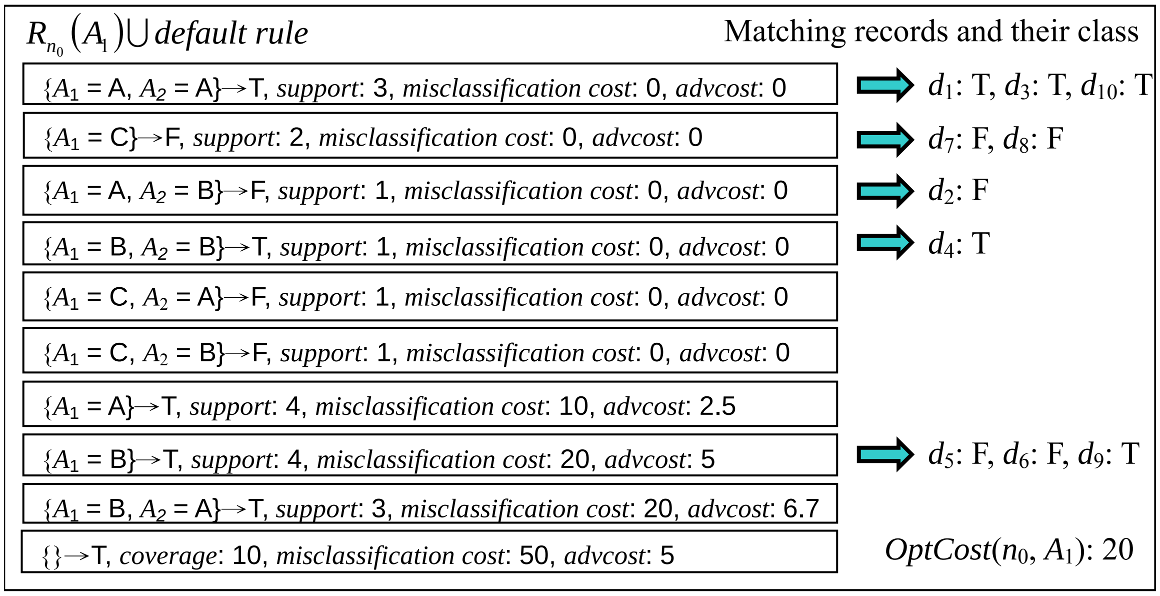
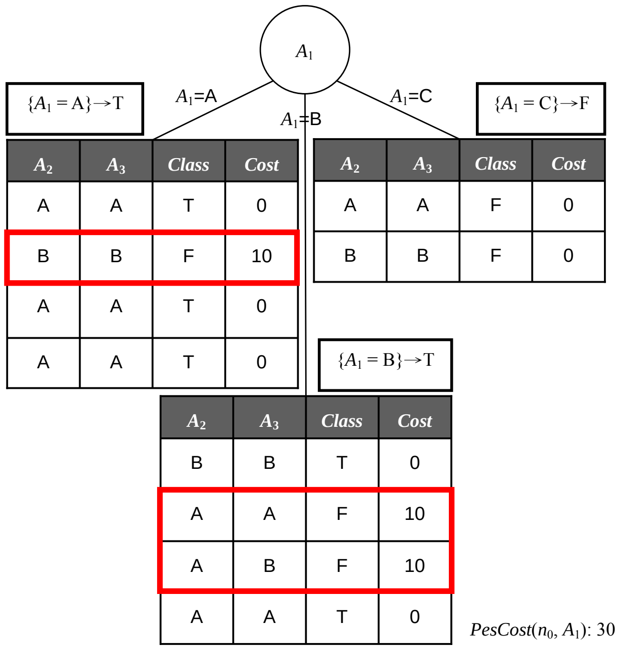

$R_{n_0}(A_1) \cup \textit{default rule}$
Matching records and their class
{A_1 = A, A_2 = A}→T, *support*: 3, *misclassification cost*: 0, *advcost*: 0
d_1: T, d_3: T, d_{10}: T
{A_1 = C}→F, *support*: 2, *misclassification cost*: 0, *advcost*: 0
d_7: F, d_8: F
{A_1 = A, A_2 = B}→F, *support*: 1, *misclassification cost*: 0, *advcost*: 0
d_2: F
{A_1 = B, A_2 = B}→T, *support*: 1, *misclassification cost*: 0, *advcost*: 0
d_4: T
{A_1 = C, A_2 = A}→F, *support*: 1, *misclassification cost*: 0, *advcost*: 0
{A_1 = C, A_2 = B}→F, *support*: 1, *misclassification cost*: 0, *advcost*: 0
{A_1 = A}→T, *support*: 4, *misclassification cost*: 10, *advcost*: 2.5
{A_1 = B}→T, *support*: 4, *misclassification cost*: 20, *advcost*: 5
d_5: F, d_6: F, d_9: T
{A_1 = B, A_2 = A}→T, *support*: 3, *misclassification cost*: 20, *advcost*: 6.7
{}→T, *coverage*: 10, *misclassification cost*: 50, *advcost*: 5
OptCost(n_0, A_1): 20

Fig. 8. The example of measuring *OptCost*(n_0, A_1)

The value of *PesCost*(n_i, A_x) is estimated by assuming that after splitting n_i with A_x all the immediate children will be leaf nodes. Suppose that the label of each leaf node is determined by the class with the minimal misclassification cost. Then, the value of *PesCost*(n_i, A_x) will be equal to the sum of all misclassification costs of the child nodes generated by splitting n_i with A_x. However, if a node has too few data or no data, we use the minimal misclassification cost label of its parent as its label.

Consider the example shown in Fig. 9, where three child nodes are generated by splitting n_0 with A_1. If no other attributes will be measured after A_1, there are only three rules: $\{A_1 = \mathrm{A}\} \rightarrow \mathrm{T}$, $\{A_1 = \mathrm{B}\} \rightarrow \mathrm{T}$, and $\{A_1 = \mathrm{C}\} \rightarrow \mathrm{F}$ that can be used to classify the records in D_{n_0}. The value of *PesCost*(n_0, A_1) is the accumulation of all misclassification costs of the classifications, and the result is *PesCost*(n_0, A_1) = 30.


A_1
A_1=A
A_1=B
A_1=C
$\{A_1 = \text{A}\}\rightarrow$T
$\{A_1 = \text{C}\}\rightarrow$F
$\{A_1 = \text{B}\}\rightarrow$T
$PesCost(n_0, A_1)$: 30

A_2	A_3	*Class*	*Cost*
A	A	T	0
B	B	F	10
A	A	T	0
A	A	T	0

A_2	A_3	*Class*	*Cost*
A	A	F	0
B	B	F	0

A_2	A_3	*Class*	*Cost*
B	B	T	0
A	A	F	10
A	B	F	10
A	A	T	0

Fig. 9. The example of measuring $PesCost(n_0, A_1)$

For each non-leaf node n_i, if there exists any attribute A_x where $OptCost(n_i, A_x) < NodeCost(n_i)$, select an attribute A_y which has the minimum $OptCost(n_i, A_y)$ to further split node n_i. If there is more than one attribute having the minimum *OptCost*, from them we select the attribute with the minimum *PesCost*. If $OptCost(n_i, A_x) \geq NodeCost(n_i)$ for all attributes A_x, assign n_i as a leaf, and label n_i with the class that minimizes $NodeCost(n_i)$. In our example, both $OptCost(n_0, A_1)$ and $OptCost(n_0, A_2)$ are 20, but $PesCost(n_0, A_1) = 30$, which is smaller than $PesCost(n_0, A_2) = 50$. Therefore, A_1 is selected to be the splitting attribute of the node n_0.

Suppose attribute A_x has m possible values. When n_i is split by A_x, there will be $(m+1)$ child nodes emanated from n_i, where m of them correspond to the values of A_x and the last one is connected to n_i by an edge marked “unused.” For the last node n_u, which is connected with the “unused” edge, $ante(n_u) = ante(n_i)$, $EXA_{n_u} = EXA_{n_i} \cup \{A_x\}$. For any other nodes n_j, $ante(n_j) = ante(n_i) \cup \{(A_x, v(n_i, n_j))\}$, and

$EXA_{n_j} = EXA_{n_i}$. In our example, there are three possible values of A_1. After splitting n_0 with A_1, there are four child nodes produced from n_0, as shown in Fig. 6. Among the four child nodes, $ante(n_{01})$ = $\{(A_1, \mathrm{A})\}$, and $EXA_{n_{01}} = \{\}$, while $ante(n_{04}) = \{\}$, and $EXA_{n_{04}} = \{A_1\}$.

Let n_h be the parent of n_i. A node n_i will stop splitting and become a leaf node if and only if:

- No attribute is selected by the splitting criterion.
- There exists only one rule r in R_{n_i}, where $condition(r) = ante(n_i)$. This may be caused by the fact that further splits at n_i will result in violation of at least one resource constraint, or by the fact that there are not sufficent records in all branches of n_i. In this case, we stop splitting at n_i and label n_i with the *consequent* of r.
- There exists no rule in R_{n_i}. This condition occurs because the number of records in some child nodes of n_h exceeds *mincov,* but in n_i it does not. In this case, n_i will be labeled with the class that minimizes the misclassification cost of n_h.
- There exist multiple rules in R_{n_i}, and $OptCost(n_h, s(n_h)) = PesCost(n_h, s(n_h))$. The equality of $OptCost(n_h, s(n_h))$ and $PesCost(n_h, s(n_h))$ indicates that the misclassification cost cannot be reduced after splitting n_h, the parent node of n_i. Therefore, we should stop splitting at n_i and label it with the class that minimizes the misclassification cost of n_i.

The output of phase 2 in our example is shown in Fig. 10.

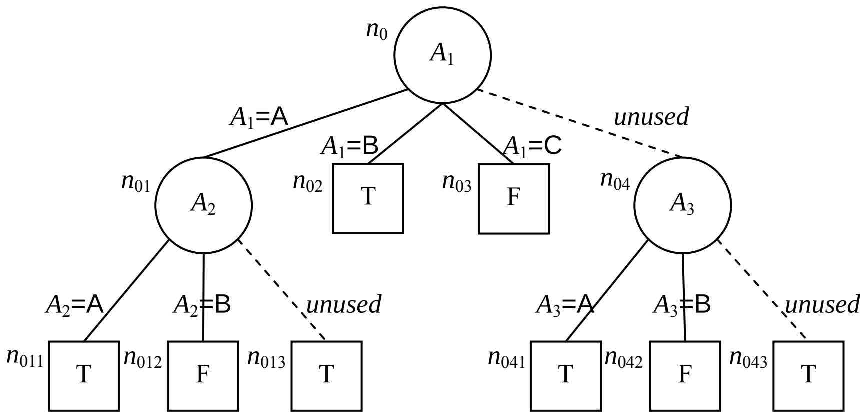


Fig. 10. The output of phase 2

III.3 Phase 3: Adjustment for producing the final decision tree

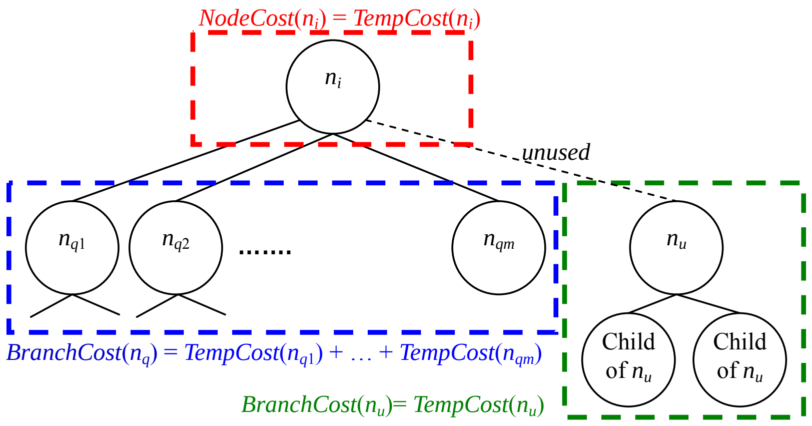
In phase 2, we build a tree from the classification rules extracted in phase 1. During the tree-induction process, we iteratively generate new branches from internal nodes. Although two measurements have been used to help evaluate whether an internal node should be further split or just stopped as a leaf node, it is difficult to make an accurate evaluation unless we have the final complete tree. Therefore, in phase 3, we will re-examine the entire tree carefully from bottom to top to determine whether there are any unnecessary branches. In other words, we hope to improve the tree by pruning unnecessary branches or nodes. The detailed procedure of phase 3 is shown in Fig. 11.

```
Algorithm Adjustment
Input: The decision tree $T_{p2}$ which is built in phase2
Output: The final decision tree

  for each internal node $n_{internal}$ in $T_{p2}$
     { $checked(n_{internal})$ = false; $TempCost(n_{internal}) = NodeCost(n_{internal})$;}
  for each leaf node $n_{leaf}$ in $T_{p2}$
     { $checked(n_{leaf})$ = true; $TempCost(n_{leaf}) = NodeCost(n_{leaf})$; }

  repeat until for all nodes $n_i$, $ckecked(n_i)$ = true
  {
      find a node $n_i$, where $checked(n_i)$ = false and all child nodes of $n_i$ are checked;

      /* suppose $n_i$ has $(m+1)$ child nodes, which are $n_{q1}, n_{q2}, \ldots, n_{qm}$, and $n_u$,
         where $n_u$ is linked to $n_i$ with an edge marked “unused”. */

      $BranchCost(n_q) = TempCost(n_{q1}) + TempCost(n_{q2}) + \ldots + TempCost(n_{qm})$;
      if( $TempCost(n_i) <= BranchCost(n_q)$ && $TempCost(n_i) <= TempCost(n_u)$ )
        { prune all branches emanated from $n_i$, $n_i$ becomes a leaf node;
          $checked(n_i)$ = true; }
      if( $BranchCost(n_q) < TempCost(n_i)$ && $BranchCost(n_q) <= TempCost(n_u)$ )
        { prune node $n_u$ and all branches from $n_u$;
         $TempCost(n_i) = BranchCost(n_q)$;
         $checked(n_i)$ = true;}
      if( $TempCost(n_u) < TempCost(n_i)$ && $TempCost(n_u) < BrabchCost(n_q)$ )
          replace $n_i$ with $n_u$;
  }
```

Fig. 11. The algorithm of phase 3

For each node n_i in the decision tree generated in phase 2, we use $checked(n_i)$ to indicate whether n_i is checked. For a checked node n_i, $TempCost(n_i)$ is used to store the misclassification cost of the subtree with the root node n_i. Initially, for all nodes n_i, $TempCost(n_i)$ is set to be $NodeCost(n_i)$ and is updated as we produce the final tree. Furthermore, $checked(n_i)$ is set to be “false” for all internal nodes, and $checked(n_i)$ “true” for all leaf nodes.

We adjust the tree from the bottom to the top by repeatedly processing all unchecked nodes, whose child nodes have been checked. Consider a node n_i as shown in Fig. 12. Suppose there are $(m+1)$ child nodes of n_i, where n_{q1}, n_{q2}, …, and n_{qm} correspond to the values of $s(n_i)$, and n_u is linked to n_i with an edge marked “unused.” Additionally, all the $(m+1)$ child nodes of n_i are checked.



$NodeCost(n_i) = TempCost(n_i)$
n_i
unused
n_{q1}
n_{q2}
....
n_{qm}
n_u
Child of n_u
Child of n_u
$BranchCost(n_q) = TempCost(n_{q1}) + \ldots + TempCost(n_{qm})$
$BranchCost(n_u) = TempCost(n_u)$

Fig. 12. A node being processed in phase 3

We compare three costs:

- *NodeCost*(n_i): the misclassification cost of assigning n_i as a leaf node, which is equal to *TempCost*(n_i).
- *BranchCost*(n_q): the total misclassification cost of classifying the data in D_{n_i} using the subtrees with the root nodes: n_{q1}, n_{q2}, …, and n_{qm}. The value of *BranchCost*(n_q) is obtained by summing *TempCost*(n_{q1}), *TempCost*(n_{q2}), …, and *TempCost*(n_{qm}).
- *BranchCost*(n_u), the misclassification cost resulting from classifying the data in D_{n_i} using the subtree with the root node n_u. That is, attribute $s(n_i)$ will not be used. Instead, classification will be performed using $s(n_u)$.

There are three cases about the comparison result. These three cases correspond to the following three situations:

Case 1: $NodeCost(n_i) \leq BranchCost(n_q)$ and $NodeCost(n_i) \leq BranchCost(n_u)$. In this case, further splitting cannot reduce the misclassification cost of the records in node n_i. Therefore, we prune all branches generated from n_i, and n_i becomes a leaf node.

Case 2: $BranchCost(n_q)<NodeCost(n_i)$ and $BranchCost(n_q) \leq BranchCost(n_u)$. In this case, it is beneficial to use $s(n_i)$ to classify the data in D_{n_i}. Therefore, we prune n_u and its sub-branches**.**

Case 3: $BranchCost(n_u)<BranchCost(n_q)$ and $BranchCost(n_u)<NodeCost(n_i)$. In this case, it is not beneficial to use $s(n_i)$ to classify the data in D_{n_i}. Therefore, we replace n_i with n_u.

Fig. 13 displays the adjustments in the three cases. For example, consider the node n_{01} in the figure 10, whose splitting attribute is A_2. There are three child nodes of n_{01}. Two of them are n_{011} and n_{012}, which related to "A_2=A" and "A_2=B," respectively, and the last child node n_{013} is linked to n_{01} with an edge marked "unused." After computation, we have $NodeCost(n_{01}) = 10$, $NodeCost(n_{011}) + NodeCost(n_{012}) = 0$, and $NodeCost(n_{013}) = 10$. Obviously, $NodeCost(n_{011}) + NodeCost(n_{012})$ is the smallest one. Therefore, we prune n_{013}.

In Cases 1 and 2, $checked(n_i)$ will be set to be true. We only need to update $TempCost(n_i)$ in Case 2 by setting it equal to $BranchCost(n_q)$. In our example, the values of $TempCost(n_{01})$ and $checked(n_{01})$ become "0" and "true," respectively. After this phase, the final tree of our example is shown in Fig. 2. Note that , in Case 3, since the sub-tree with the root node n_i is replaced by that with the root node n_u, it is not necessary to update $TempCost(n_i)$.

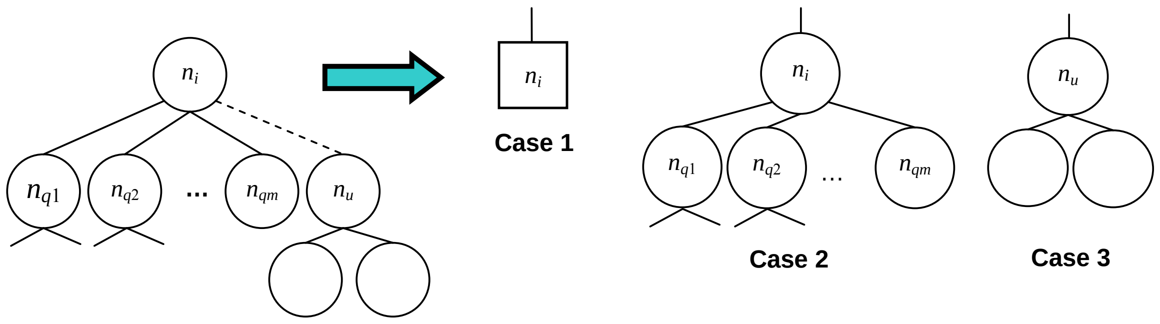


Fig. 13. Adjustments of three cases

IV. Performance Evaluation

In this section, we perform experiments to evaluate the proposed algorithm, CAT. Twelve datasets obtained from the UCI Machine Learning repository [2][5] are used in the experiments. A description of these datasets is given in Table V. The numerical attributes in the datasets are discretized using the minimal entropy method in the data preparation stage [8]. Furthermore, we removed all records containing missing values

TABLE V.
Datasets We Adapted in Experiments

Dataset	Number of attributes	Number of examples	Class distribution
Acute Inflammations(D1)	6	120	no: 61 yes: 59
Acute Inflammations(D2)	6	120	no:70 yes: 50
Car	6	1728	unacc: 1210 acc: 384 good: 69 v-good: 65
Contraceptive Method Choice	9	1473	1: 629 2: 333 3: 511
Credit Approval	15	690	+: 307 -: 383
Heart Disease	13	303	0: 164 1: 139
Nursery	8	12960	not_recom: 4320 recommend: 2 very_recom: 328 priority: 4266 spec_prior: 4044
Pima Indians Diabetes	8	768	Healthy: 500 Diabetes: 268
Thyroid Disease	20	3772	Hypothroid: 93 Hyperthyroid: 191 Normal: 3488
Tic-Tac-Toe	9	958	positive:626 Negative:332
Voting	16	435	Democrat: 267 Republican: 168
Zoo	16	101	1: 41 2: 20 3: 5 4: 1 5: 4 6: 8 7: 10

We compare the CAT algorithm, labeled as M4, with four additional approaches, M0 through M3 described in Table VI, in the experiments. Note that M0 is used as a baseline, which represents an approach without using a classification model.

TABLE VI.
Five Methods in the Experiment

Methods	Description
M0	Classify an instance to the class, which minimizes the misclassification cost.
M1	A top-down approach, which selects the splitting attribute with the maximum reduction in the misclassification cost. Resource constraints are not considered.
M2	A top-down approach, which selects the splitting attribute with the smallest sum of the total misclassification cost and the total resource usage.
M3	A top-down approach, which selects the splitting attribute with the largest ratio of the reduction in the total misclassification cost and the total resource usage.
M4	CAT

Most of the original datasets do not contain cost information. For our experiments, the misclassification costs are determined by the distribution of the class labels. Following the method in [11] and [26], we use 200 as the baseline for the per-case cost of misclassification and make adjustments so that the cost is higher for misclassifying rare classes. For example, in the Pima Indians Diabetes dataset, the ratio of the Healthy label to the Diabetes level is roughly 2 to 1. Therefore, we set the costs of misclassification for this dataset as $MisCost\ (1, 0) = 200$ and $MisCost\ (0, 1) = 400$.

We ran a cross validation in each experiment. Datasets were randomly split into ten parts. Ten pairs of the training and test datasets were generated, where the test dataset involved one partition and the training dataset involved the other nine partitions. We built decision trees with the training datasets and evaluate the trees with the test datasets. The final result from each experiment is the average of the results obtained with the ten pairs of training and test datasets.

In the remainder of this section, we evaluate the performance of the methods under a single resource constraint (Section IV.1), two resource constraints (Section IV.2), and multiple resource constraints (Section IV.3).

IV.1 Performance evaluation under a single constrained resource

We first evaluate the four methods under one resource constraint. In order to determine a possible upper bound on *MaxResCon* for a dataset, we first obtain a decision tree by M1 without any resource constraints and obtain the maximal resource consumption, denoted by *MaxTotalCon*, to arrive a leaf node. To observe how the scarcity of resource impacts performance, we obtain the results by gradually reducing *MaxResCon* from *MaxTotalCon* to 90%×*MaxTotalCon*, 80%×*MaxTotalCon*, …, and 20%×*MaxTotalCon*. Furthermore, we randomly set the resource consumption of each attribute between 1 and 100, and the minimal coverage threshold *mincov* 5%.

The results are presented in Table VII, where the columns list the levels of available resources, the rows show the datasets and the four approaches, and the cell show values that are the ratio of the misclassification cost of the corresponding approach and resource level to that of M0. The results are analyzed by a three-factor ANOVA model, where the three-way interaction is not used, because no replications were performed under each treatment level. It was found that the three main effects, Dataset, Method, and Percentage, are statistically significant at the 5% level, and the two-way interactions are also significant, except that between Dataset and Method. The results of Tukey's multiple comparisons indicate that CAT significantly outperforms the other three approaches, and that M2 is significantly better than M1 and M3. These results are expected because our method benefits from selecting splitting attributes according to all possible sequences of attributes, whereas the traditional top-down approach considers only the result of splitting a present node.

TABLE VII.
EXPERIMENT RESULTS FOR EACH METHOD IN DIFFERENT DATASETS

Dataset	Method	*MaxResCon* (*form* 100%×*MaxTotalCon* to 20%×*MaxTotalCon*)								
		100%	90%	80%	70%	60%	50%	40%	30%	20%
Acute Inflammations (D1)	M1	0.1475	0.1639	0.1639	0.1639	0.1639	0.1639	0.1639	0.3443	0.4098
	M2	0.1639	0.1639	0.1639	0.1639	0.1639	0.1639	0.1639	0.3279	0.3279
	M3	0.1639	0.1639	0.1639	0.1639	0.1639	0.1639	0.1639	0.1639	0.3279
	M4	0.0000	0.0000	0.0000	0.0000	0.0000	0.1639	0.1639	0.1639	0.3279
Acute Inflammations (D2)	M1	0.0000	0.1429	0.1429	0.1429	0.1429	0.1429	1.0000	1.0000	1.0000
	M2	0.0000	0.1429	0.1429	0.1429	0.1429	0.1429	0.4200	0.7714	0.7714
	M3	0.0000	0.1429	0.1429	0.1429	0.1429	0.1429	0.7543	0.7714	0.7714
	M4	0.0000	0.0000	0.0000	0.0000	0.1429	0.1429	0.4200	0.7714	0.7714
Car	M1	0.4999	0.4999	0.4999	0.5768	0.5768	0.5768	0.5768	0.6656	0.6752
	M2	0.5321	0.5321	0.5321	0.5321	0.5321	0.5321	0.5321	0.5068	0.6752
	M3	0.6816	0.6816	0.6816	0.6816	0.6816	0.6816	0.6816	0.6816	0.8015
	M4	0.4999	0.4999	0.4999	0.5280	0.5280	0.5086	0.5086	0.5068	0.6752
Contraceptive Method Choice	M1	0.8394	0.8394	0.8394	0.8394	0.8340	0.8340	0.8517	0.8606	0.8695
	M2	0.8780	0.8780	0.8780	0.8780	0.8780	0.8780	0.8869	0.8869	0.8737
	M3	0.8699	0.8699	0.8699	0.8699	0.8699	0.8699	0.8737	0.8737	0.8737
	M4	0.7506	0.7641	0.7618	0.7552	0.7548	0.7263	0.7811	0.8039	0.8069
Credit Approval	M1	0.2600	0.2600	0.2600	0.2600	0.2600	1.0000	1.0000	1.0000	1.0000
	M2	0.4790	0.4790	0.5995	0.5995	0.6081	0.6342	0.6342	0.6752	0.6752
	M3	0.4005	0.4005	0.4005	0.4005	0.5916	0.7133	0.6860	0.7099	0.7099
	M4	0.2895	0.2782	0.2691	0.2600	0.2600	0.5540	0.5614	0.6752	0.6752
Heart Disease	M1	0.5500	0.5500	0.5500	0.5500	0.5500	0.5500	0.5500	0.6090	0.6090
	M2	0.4718	0.4718	0.4718	0.4718	0.4718	0.4718	0.4718	0.4718	0.5090
	M3	0.4885	0.4885	0.4885	0.4885	0.4885	0.4885	0.4885	0.4885	0.5090
	M4	0.4910	0.4231	0.3615	0.3782	0.4474	0.4359	0.4641	0.4295	0.4872
Nursery	M1	0.2970	0.2970	0.2970	0.2970	0.2970	0.4686	0.4686	1.0000	1.0000
	M2	0.3248	0.3248	0.3248	0.3248	0.3248	0.3979	0.4247	0.7671	0.8290
	M3	0.6982	0.6982	0.6982	0.6982	0.6982	0.6982	0.6982	0.7671	0.8290
	M4	0.2970	0.2970	0.2970	0.2970	0.2970	0.3932	0.4247	0.6982	0.7860
Pima Indians Diabetes	M1	0.5720	0.5720	0.5720	0.5640	0.5460	0.5460	0.5460	0.5760	0.5760
	M2	0.5580	0.5580	0.5580	0.5380	0.5380	0.5500	0.5460	0.5860	0.6060
	M3	0.5660	0.5660	0.5660	0.5460	0.5460	0.5460	0.5460	0.5860	0.6060
	M4	0.5480	0.5480	0.5800	0.5620	0.5140	0.5640	0.5440	0.5300	0.5760
Thyroid Disease	M1	0.5078	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	M2	0.5078	0.6765	0.6765	0.6765	0.6765	0.6765	0.6765	0.9802	0.9903
	M3	0.8118	0.8118	0.8141	0.8118	0.8118	0.8118	0.8118	0.9802	0.9903
	M4	0.5078	0.6241	0.6241	0.6765	0.6765	0.6765	0.6765	0.9994	0.9877
Tic-Tac-Toe	M1	0.6725	0.6725	0.6725	0.6725	0.6565	0.6565	0.6677	1.0000	1.0000
	M2	0.8035	0.8035	0.8035	0.8035	0.8035	0.8035	0.8035	0.8083	0.8083
	M3	0.6645	0.6645	0.6645	0.6645	0.6645	0.6837	0.7013	0.7971	0.8115
	M4	0.5304	0.5112	0.5687	0.6358	0.6166	0.6374	0.6629	0.7812	0.7875
Voting	M1	0.0585	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	M2	0.2621	0.2621	0.2621	0.2621	0.2903	0.2903	0.2903	0.2903	0.2903
	M3	0.3891	0.3427	0.3569	0.3367	0.3992	0.3992	0.3931	0.3972	0.3629
	M4	0.0585	0.2540	0.3044	0.3024	0.3327	0.2722	0.2621	0.2903	0.2903
Zoo	M1	0.2283	0.2283	0.2283	0.2283	0.2862	0.4239	0.4239	0.4239	1.0000
	M2	0.3297	0.3297	0.3297	0.3297	0.3297	0.3297	0.3297	0.4239	0.1957
	M3	0.1848	0.1848	0.1848	0.1848	0.1848	0.1848	0.1848	0.1848	0.2065
	M4	0.0725	0.0725	0.1558	0.1558	0.1413	0.1667	0.1522	0.1920	0.1993

IV.2 Performance evaluation under two resource constraints

In the evaluation reported in this section, we apply two constraints to the classification task, one for the attribute costs, and the other for the completion time. The Heart Disease dataset is used in the evaluation. The dataset was made available by Robert Detrano of the Cleveland Clinic Foundation. The attribute costs and times given in Table VIII are obtained from the literature [17][20]. In addition, the misclassification costs are *MisCost*(1, 0) = 600 and *MisCost*(0, 1) = 1,000 [17].

TABLE VIII.
THE ATTRIBUTES IN THE HEART DISEASE DATASET

Attribute/Test	Test Fee (CN$)	Test Time (Minutes)	Description
num	-	-	diagnosis of a heart disease 0: less than 50% artery narrowing; 1: more than 50% artery narrowing
age	1.00	1	in years
sex	1.00	1	sex (1 = male; 0 = female)
cp	1.00	1	chest pain type 1: typical angina; 2: atypical angina; 3: non-anginal pain; 4: asymptomatic
trestbps	1.00	1	resting blood pressure (in mm Hg on admission to the hospital)
chol	7.27	240	serum cholestoral in mg/dl
fbs	5.20	240	fasting blood sugar > 120 mg/dl
restecg	15.50	30	resting electrocardiographic results
thalach	102.90	60	maximum heart rate
exang	87.30	60	exercise induced angina (1 = yes; 0 = no)
oldpeak	87.30	60	ST depression induced by exercise relative to rest
slope	87.30	60	the slope of the peak exercise ST segment
ca	100.90	60	number of major vessels (0-3) colored by flourosopy
thal	102.90	60	3 = normal; 6 = fixed defect; 7 = reversible defect

As in Section IV.1, the four methods are compared under different limits for the two constraints. We first obtain the decision tree for M1 and evaluate the resource use, $MaxTotalCon_1$ and $MaxTotalCon_2$, respectively, for time and cost. Three levels – 75%, 50%, and 25% of $MaxTotalCon_1$ – are used for the time constraint, whereas the limit for the cost constraint remains at $MaxTotalCon_2$. The cost comparisons between selected methods and M0 are reported in Fig. 14.

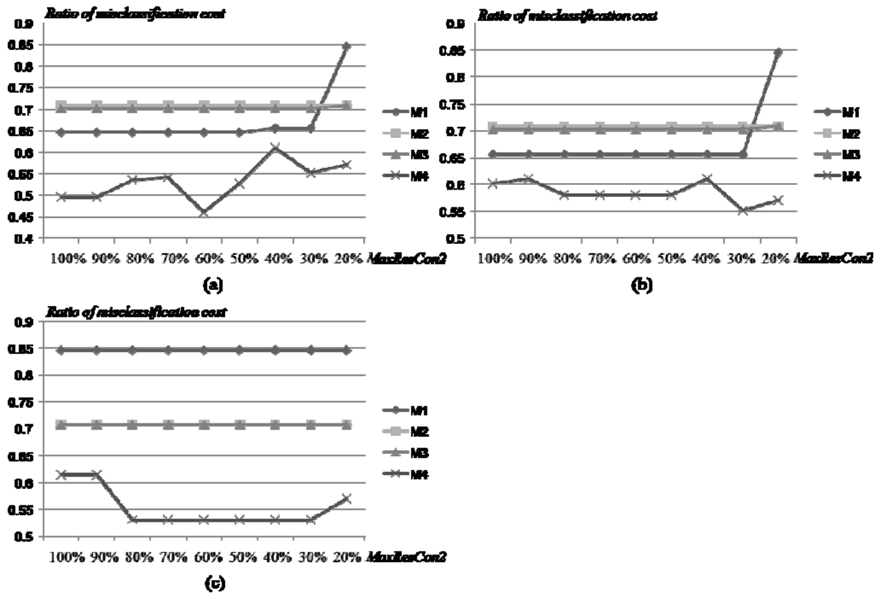


Fig. 14. The ratio of misclassification cost of each method for datasets Heart Disease under (a)$MaxResCon_1$ = 75%×$MaxTotalCon_1$, (b) $MaxResCon_1$ = 50%×$MaxTotalCon_1$, and (c)$MaxResCon_1$ = 25%×$MaxTotalCon_1$

From the results shown in Fig.14, the proposed method, CAT, is clearly the overall best performer, especially under tight resource constraints. This finding empirically demonstrates that CAT is very effective in reducing the misclassification cost under limited resources. Furthermore, the performance of M2 and M3 are generally not affected by resource constraints. These results suggest that a top-down approach is unable to adapt to the availability of resources, especially when resources are abundant.

IV.3 Performance evaluation under multiple resource constraints

We use all the datasets listed in Table V in this section. Since the cost and time information for most of these datasets are not available, we generate the values randomly in our evaluation. Five resource constraints are considered with the consumption rate of each resource by each attribute was set randomly between 1 and 100. For each constrained resource Res_y, the resource constraint $MaxResCon_y$ was set randomly among 75%×$MaxTotalCon_y$, 50%×$MaxTotalCon_y$, and

25%×$MaxTotalCon_y$.

The results are shown in Table IX, where the columns represent the number of constrained resources, the rows give the various alternatives, and the cell values show the ratio of the misclassification cost resulting from this method and the misclassification cost resulting from M0.

TABLE IX
EXPERIMENT RESULTS OF DIFFERENT NUMBER OF CONSTRAINED RESOURCE

DataSet	Method	*Number of Constrained Resource*					DataSet	Method	*Number of Constrained Resource*				
		1	2	3	4	5			1	2	3	4	5
Acute Inflammations (D1)	M1	0.1639	0.1639	0.1639	1.0000	1.0000	Nursery	M1	0.4686	0.4686	1.0000	1.0000	1.0000
	M2	0.1639	0.1639	0.1639	0.8361	0.8361		M2	0.4797	0.7689	0.8404	0.8404	0.8404
	M3	0.1639	0.1639	0.1639	0.4754	0.8361		M3	0.5102	0.6932	0.8404	0.8404	0.7860
	M4	0.0000	0.1639	0.1639	0.4754	0.8197		M4	0.3633	0.4686	0.7860	0.7860	0.7860
Acute Inflammations (D2)	M1	0.2000	1.0000	1.0000	1.0000	1.0000	Pima Indians Diabetes	M1	0.5720	0.5900	0.5900	0.5900	0.5900
	M2	0.2000	0.8000	0.8000	1.0000	1.0000		M2	0.5340	0.6900	0.7300	0.6900	0.8120
	M3	0.0000	0.8000	0.4000	1.0000	1.0000		M3	0.5340	0.6900	0.7280	0.5900	0.5900
	M4	0.0000	0.4200	0.4000	1.0000	1.0000		M4	0.5280	0.5900	0.5900	0.5900	0.5900
Car	M1	0.5983	0.6752	0.6752	0.6752	0.6752	Thyroid Disease	M1	1.0000	1.0000	1.0000	1.0000	1.0000
	M2	0.5321	0.7013	0.7460	0.7820	0.7820		M2	0.6937	0.8769	0.6937	1.0000	1.0000
	M3	0.5399	0.7013	0.6752	0.7820	0.6752		M3	0.8769	0.8769	0.6937	1.0000	1.0000
	M4	0.5327	0.7074	0.6752	0.6752	0.6752		M4	0.6937	0.6937	0.6937	1.0000	1.0000
Contraceptive Method Choice	M1	0.8429	0.8429	0.8695	0.8695	0.8695	Tic-Tac-Toe	M1	0.6725	0.6725	0.6725	0.6805	0.6805
	M2	0.8672	0.8610	0.8707	0.9004	0.8695		M2	0.6581	0.6693	0.6326	0.8227	0.6486
	M3	0.8683	0.8544	0.8436	0.8695	0.8695		M3	0.6757	0.6677	0.6470	0.6486	0.6597
	M4	0.7483	0.7471	0.7737	0.7795	0.8695		M4	0.6022	0.5879	0.6278	0.6645	0.6294
Credit Approval	M1	0.2600	0.2600	1.0000	1.0000	1.0000	Voting	M1	1.0000	1.0000	1.0000	1.0000	1.0000
	M2	0.3430	0.5529	0.7799	0.7799	0.5563		M2	0.2742	0.3548	0.3065	0.3065	1.0000
	M3	0.2600	0.2600	0.7309	0.5563	0.5563		M3	0.3548	0.3548	0.3065	0.3065	1.0000
	M4	0.2691	0.2600	0.5563	0.5563	0.5563		M4	0.2601	0.2984	0.3065	0.3065	1.0000
Heart Disease	M1	0.5500	0.5500	0.5449	0.5449	0.5974	Zoo	M1	0.2283	0.2283	0.2283	0.2283	0.4239
	M2	0.5346	0.6038	0.5590	0.5603	0.6538		M2	0.2935	0.2935	0.2862	0.2862	0.3188
	M3	0.5372	0.4577	0.5321	0.5551	0.6218		M3	0.2971	0.2935	0.2862	0.2862	0.3188
	M4	0.4577	0.4551	0.4551	0.4744	0.4962		M4	0.0870	0.1594	0.2174	0.1159	0.2609

As shown in Table IX, in a majority of the cases, as the number of constrained resources increases, the number of attributes that can be used decreases and the misclassification cost increases. In some cases, the performances of the four methods are roughly the same. This phenomenon happens when the depth of the tree is short because only a small number of attributes can be used as the resource constraints become tight. As a result, the variation in selected attributes is small among the four methods. However, CAT, still achieved a better or at least comparable performance, compared with the other three methods. The results from the statistical analysis using a three-factor ANOVA model show

CAT significantly outperforms the other three methods in the average cost. However, M2 and M3 are not significantly different in their performances.

From the results of the experiment presented in this section, we conclude that the proposed algorithm, CAT, outperforms the top-down approach and produces satisfactory results in all the situations under consideration.

In order to study the time complexity of CAT, we use the Nursery dataset, which has 12,960 records and 8 attributes, to show the effect of the number of records on its computation time. We also use the Zoo dataset, which has 16 attributes and 101 records, to show the effect of the number of attributes. The results are reported in Figure 15, where the total computation time is plotted against the number of attributes or records.

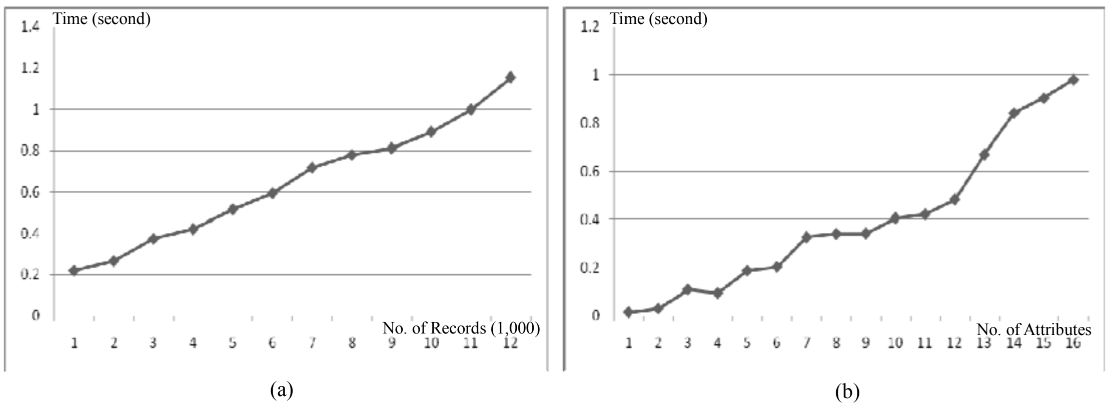


Fig. 15. The computation time of (a) different data size of Nursery dataset (b) different number of attribute of zoo dataset

The figure shows the total computation time increases linearly as the number of attributes or records increases, suggesting the proposed method performs satisfactorily. It is evident that phase 1 consumes a large portion of the computation time. Note that, in phase 1, the Apriori algorithm is modified to generate association rules, which satisfy the *mincov* threshold. We follow the same approach to develop an Apriori-like approach to work simultaneously for resource constraints. This combination effectively reduces the computation time by systematically pruning un-desired rules with respect to *mincov* and resources in early stages. The results suggest this approach works efficiently.

V. CONCLUSION AND FUTURE STUDIES

Cost-sensitive classification has been a popular research topic in recent years. In this paper, we study a problem of developing a cost-sensitive decision tree under multiple resource constraints. A new algorithm is developed to address the drawbacks of the traditional top-down approach to tree induction. The algorithm consists of two steps. In the first step, we extract association classification rules from the training dataset first and retain those that satisfy the resource constraints. In the second step, we use the association classification rules obtained in the first step to develop a cost-sensitive decision tree. Because only feasible classification rules are extracted and retained in step 1, it is not necessary to consider resource constraints in step 2. The results of an extensive numerical experiment show that the proposed algorithm produces satisfactory results under different tightness levels of resource constraints and outperforms the top-down approach.

There are several possible extensions to this study. First, we allow only attributes as predictors in the proposed algorithm. An implicit assumption is that we would have to perform discretization to allow using numerical variables in the algorithm. An extension allowing direct use of numerical variables would broaden the applicability of the algorithm. Furthermore, we assume no missing values in developing the proposed algorithm. The internal node strategy, proposed by Ling, et.al [10] and adopted by Zhang, et al. [26], may be incorporated in the proposed algorithm to handle missing data.

Zadrozny, Langford, and Abe [25] developed a novel method for generating cost-sensitive trees using existing non-cost based tree induction algorithms. They developed a conversion method by incorporating cost factors into existing classification algorithms or by subsampling the training data. They did not consider resource constraints. However, their method may be extended to constrained problems. Finally, as Verhein and Chawla [22] have demonstrated, many existing methods may not work well for imbalanced data. They proposed an attribute selection criterion called the class correlation ratio and showed it was more effective than support and confidence in generating associative classifiers. It would be an interesting research issue on whether the class correlation ratio or a similar measure could be used in the first phase of our algorithm when encountering an imbalanced dataset. These extensions would represent a line of possible research issues in contrast to the popular top-down approach in tree induction.

REFERENCES

[1] A. Arnt and S. Zilberstein, "Learning Policies for Sequential Time and Cost Sensitive Classification," *Proc. 1st International Workshop on Utility-Based Data Mining*, pp. 39-45, 2005.

[2] A. Asuncion and D.J. Newman, "UCI Machine Learning Repository," http://www.ics.uci.edu/~mlearn/MLRepository.html, 2007

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, CA, Wadsworth, Belmont, 1984.

[4] Y. L. Chen, C. C. Wu, and K. Tang, "Building a Cost-Constrained Decision Tree with Multiple Condition Attributes," *Information Sciences*, vol. 179, No. 7, pp. 967-979, 2009.

[5] J. Czerniak, H. Zarzycki, "Application of rough sets in the presumptive diagnosis of urinary system diseases," *Proc. ACS'2002 9th International Conference*, pp. 41-51, 2003.

[6] P. Domingos, “MetaCost: A General Method for Making Classifiers Cost-Sensitive,” *Proc. Fifth International Conference on Knowledge Discovery and Data Mining*, pp. 155-164, 1999.

[7] C. Elkan, “The Foundations of Cost-Sensitive Learning,” *Proc. Seventeenth International Joint Conference on Artificial Intelligence*, Seattle, Morgan Kaufmann, pp. 973-978, 2001.

[8] U.M. Fayyad and K.B. Irani, “Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning,” *Proc. 13th Int’l Joint Conf. Artificial Intelligence (IJCAI93)*, pp. 1022-1027, 1993.

[9] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2006.

[10] C.X. Ling, Q. Yang, J. Wang, and S. Zhang, “Decision Trees with Minimal Costs,” *Proc. 2004 International Conference on Machine Learning*, pp. 69, 2004.

[11] C. X. Ling, V. S. Sheng, and Q. Yang, “Test Strategies for Cost-Sensitive Decision Tree,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, No. 8, pp. 1055-1067, 2006.

[12] M. Núñez, “The Use of Background Knowledge in Decision Tree Induction.” *Machine Learning*, vol. 6, pp. 231-250, 1991.

[13] A. Papagelis and D. Kalles, "GATree: Genetically Evolved Decision Trees," *Proc. 12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI’00)* , pp. 203-206, 2000.

[14] Z. Qin, S. Zhang, and C. Zhang, “Cost-Sensitive Decision Trees with Multiple Cost Scales.” *Australian Joint Conference on Artificial Intelligence*, Cairns , Australia, vol. 3339, pp. 380-390, 2004.

[15] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol.1, pp. 81-106, 1986.

[16] J. R. Quinlan, *C4.5: Programs for Machine Learning*, San Mateo, Morgan Kaufmann, 1993.

[17] V. S. Sheng and C. X. Ling, "Feature Value Acquisition in Testing: A Sequential Batch Test Algorithm," *Proc. 23nd International Conference on Machine Learning*, pp. 809-816, 2006.

[18] M. Tan, “Cost-Sensitive Learning of Classification Knowledge and Its Applications in Robotics,” *Machine Learning*, vol. 13, pp. 7-33, 1993.

[19] K. M. Ting, “An Instance-Weighting Method to Induce Cost-Sensitive Trees,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 3, pp. 659-665, 2002.

[20] P. D. Turney, “Cost-Sensitive Classification: Empirical Evaluation of A Hybrid Genetic Decision Tree Induction Algorithm,” *Journal of Artificial Intelligence Research*, vol. 2, pp. 369-409, 1995.

[21] P. D. Turney, “Types of Cost in Inductive Concept Learning,” *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning*, pp. 15-21, 2000.

[22] F. Verhein and S. Chawla, "Using Significant, Positively Associated and Relatively Class Correlated Rules for Associative Classification of Imbalanced Datasets," *Proc. Seventh IEEE International Conference on Data Mining*, pp.679-684, 2007

[23] G. M. Weiss and F. Provost, "Learning When Training Data are Costly: The Effect of Class Distribution on Tree Induction," *Journal of Artificial Intelligence Research*, vol. 19, pp. 315-354, 2003.

[24] Q. Yang, C. Ling, X. Chai, and R. Pan, “Test-Cost Sensitive Classification on Data with Missing Values,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 5, pp. 626-638, 2006.

[25] B. Zadrozny, J. Langford, and N. Abe, "Cost-Sensitive Learning by Cost-Proportionate Example Weighting," *Proc. Third IEEE International Conference on Data Mining*, pp. 435-442, 2003.

[26] S. Zhang, Z. Qin, C. X. Ling, and S. Sheng, “Missing is Useful”: Mining Values in Cost-Sensitive Decision Trees,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no, 12, pp. 1689-1693, 2005.

[27] H. Zhao, “A Multi-Objective Genetic Programming Approach to Developing Pareto Optimal Decision Trees,” *Decision Support System*, vol. 43, no. 3, pp. 809-826, 2007.