PAPER Special Issue on Information Theory and Its Applications

Optimal Decision Tree Design Based on Information Theoretical Cost Bound*

Satoru OHTA† *and* Fumio KANAYA†, *Members*

SUMMARY Decision tree design is an important issue because decision trees have many applications in, for example, fault diagnosis and character recognition. This paper describes an algorithm designing a probabilistic decision tree, in which a test applied to a state produces different outcomes with non-zero probability. The algorithm strictly minimizes the tree cost, which is defined as the weighted sum of expected loss and expected test execution time. The lower bound of the cost is utilized to remove the excessive computing time needed to search for the exact optimum. This lower bound is derived from information theory and is transformed to an easily computable representation by introducing a network model. The result of a computational experiment confirms the time efficiency of the proposed method.

1. Introduction

A decision tree is a tree-like structure representing a sequential decision process in which several tests are successively executed to gather information about an unknown state. Depending on the outcome of these tests, decisions are made and an action is taken. Decision trees are categorized into two types: deterministic and probabilistic. With the former, a test applied to a state produces a fixed outcome with certainty, while a test yields different outcomes with non-zero probability in the latter. Each type of decision tree has been found useful for various applications such as fault diagnosis[1], computer programming[2], character recognition[3], or machine learning[4].

Because of its practical significance, considerable efforts have been focused on developing a method to construct an efficient decision tree. Some of these works have succeeded in achieving efficient heuristics that give near-optimal solutions by applying ideas from information theory[3]-[8]. However, in the design of a probabilistic decision tree, there are very few practical algorithms yielding a strict optimum for a certain criterion. This is due to difficulty in overcoming the explosion in computational time needed to search for the strict optimum. Nevertheless, it is worthwhile to develop such an algorithm from the viewpoint of not only theoretical interest but also practical necessity to evaluate the accuracy of approximate solutions obtained by heuristics.

Manuscript received February 22, 1991.
Manuscript revised May 16, 1991.

† The authors are with NTT Transmission Systems Laboratories, Yokosuka-shi, 238-03 Japan.

Branch-and-bound methods have provided efficient algorithms giving strict optimum solutions for various intractable optimization problems[9]. In this scheme, the space of all feasible solutions is divided into smaller subsets, and then a lower bound of the cost for each subset is calculated. If the bound is larger than a known feasible cost, the optimal solution is not included in the subset. Thus, such a subset can be excluded from further consideration. By repeating this process, the optimum is found by searching only a small part of the solution space. To achieve efficient computation with this scheme, a tight and easily computable lower bound is indispensable.

Lower bounds of some decision tree costs are closely related to information theory. It is known that the expected test execution time of a deterministic decision tree is lower bounded by the expected word length of the Huffman code[1],[7]. For a probabilistic decision tree, Refs. (8) and (10) have pointed out that the expected test execution time for a given value of expected loss is lower bounded by the rate-distortion function. However, there have been no studies clarifying whether these information theoretical bounds are applicable to the branch-and-bound methods used in constructing decision trees. Moreover, it is not clear whether there is a better lower bound than those currently recognized.

This paper describes an algorithm for efficiently designing a probabilistic decision tree that is strictly optimal in terms of the cost defined as the weighted sum of expected test execution time and expected loss. The proposed algorithm employs a branch-and-bound approach utilizing a lower bound of the cost. The bound is derived from an application of information theory and is tighter and easier to compute than the one derived from the rate-distortion function. The time efficiency of the proposed method is demonstrated by a computational experiment.

2. Preliminaries

Let $\mathcal{U}=\{u_1, \cdots, u_K\}$ be a finite set of unknown states and $\mathcal{A}=\{a_1, \cdots, a_I\}$ be a finite set of actions. Suppose that there are a given loss function $d: \mathcal{U} \times \mathcal{A}$

* This work was presented at ISITA'90 held in Honolulu, Hawaii, Nov. 27-30, 1990.

u_k / $Q(u_k)$	u_1 0.3		u_2 0.4		u_3 0.3	
	$P_t(0\|u_1)$	$P_t(1\|u_1)$	$P_t(0\|u_2)$	$P_t(1\|u_2)$	$P_t(0\|u_3)$	$P_t(1\|u_3)$
t_1	0.1	0.9	0.2	0.8	0.6	0.4
t_2	0.2	0.8	0.8	0.2	1.0	0.0
t_3	0.9	0.1	0.1	0.9	0.9	0.1

Fig. 1 Probabilistic decision table.

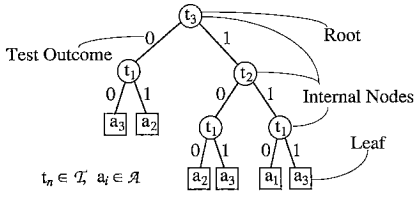


Fig. 2 Decision tree.

$\to [0, \infty)$ and a given finite set $\mathcal{T}=\{t_1, \cdots, t_N\}$ of tests to be applied to an unknown state. When a test $t \in \mathcal{U}$ is applied to a state $u \in \mathcal{U}$, one of b possible outcomes $0, \cdots, b-1$ can occur.

This paper deals with the problem of constructing a b-ary probabilistic decision tree from a given probabilistic decision table[5],[6]. The probabilistic decision table indicates probabilities $Q(u)$ and $P_t(x|u)$ for each $u \in \mathcal{U}$, $t \in \mathcal{T}$, and $x \in \{0, \cdots, b-1\}$, where $Q(u)$ is a prior probability of state u, and $P_t(x|u)$ is a conditional probability of observing x as an outcome of test t under state u. An example of such a table is shown in Fig. 1.

First, we make some assumptions as follows.

Assumptions: (1) $\mathcal{U}=\mathcal{A}$ and the loss function is Hamming metric; $d(u; a)=0$ if $u=a$ and $d(u; a)=1$ otherwise. (2) Outcomes of different tests are statistically independent. (3) Reapplication of the same test gives the same outcome; repeated execution of identical tests is useless.

These assumptions are often acceptable for practical applications, such as character recognition[3],[5].

An example of a decision tree is shown in Fig. 2. A sequential decision by a decision tree starts at a root node and ends at a leaf node. An internal node of the tree specifies a test, which is an element of $\mathcal{T}$. At the internal node, the specified test is executed for an unknown state. Depending on the test result, one of its b sons is selected as the next node to be reached. When one finally reaches a leaf by repeating this procedure, an appropriate action is taken so as to minimize expected loss, which is the sum of the loss function values weighted by the joint probabilities of state occurrence and leaf arrival. Thus, a leaf is labeled with an optimal action to be taken, while an internal node is labeled with a test as shown in Fig. 2.

3. Performance Measures

Most commonly used performance measures for decision trees are expected test execution time, expected test cost, and storage space[7],[11]. Among these, expected test execution time is used when a time-efficient decision tree is required. Expected loss is another reasonable measure for a probabilistic decision tree in which the action taken at a leaf is not necessarily the best one for the actually occurring state due to the probabilistic nature of test outcomes. In this study, the expected test execution time and the expected loss are employed as performance measures.

To decrease the expected loss, a large tree executing many tests is required, even though such a tree performs poorly from the viewpoint of test execution time. With this tradeoff, the construction of an efficient decision tree becomes an optimization problem with multiple objectives. To deal with this problem, we employ the parametric method, in which the objective cost is defined as the weighted sum of the performance measures. That is, the tree cost C is formulated as

$$C=W+\rho D, \tag{1}$$

where W is the expected test execution time, D is the expected loss, and ρ is the weight parameter ($\rho>0$). The weight parameter ρ is determined based on the requirements of the two measures: ρ is larger if the expected loss is required to be small.

In the multiobjective optimization problem, a common approach is to define the cost as the weighted sum of different objectives (for example, see Ref. (12)). Thus, costs similar to the one defined above are often utilized in the design of probabilistic decision trees[13],[14]. Another approach is to minimize one of two measures while restricting the other not to excess some specified value[3],[4]. However, the weighted sum scheme is more advantageous in that the optimization algorithm is simplified. This is because the next theorem can be utilized in optimization.

Theorem 1: Consider a tree T whose root node is not a leaf. Let $p_t(x)$ represent the probability of getting x ($0 \leqq x \leqq b-1$) as an outcome of test t executed at the root and let sons of the root be $v_0, \cdots, v_{b-1}$, where v_x corresponds to the outcome x. Also, suppose that C_x is the cost of the subtree rooted at v_x. Then, the tree cost C is expressed as follows:

$$C=1+\sum_{x=0}^{b-1} p_t(x)\, C_x. \tag{2}$$

Proof: Let W_x and D_x be the expected test execution time and the expected loss of the subtree rooted at v_x. At the root of T, a test is executed once, and more W_x tests are performed if outcome x is observed. Therefore, the expected test execution time of T, which is denoted by W, is given as follows:

$$W=1+\sum_{x=0}^{b-1} p_t(x)\, W_x. \qquad (3)$$

On the other hand, the expected loss of T, denoted by D, obviously satisfies

$$D=\sum_{x=0}^{b-1} p_t(x)\, D_x. \qquad (4)$$

From Eqs. (3) and (4),

$$C=W+\rho D=1+\sum_{x=0}^{b-1} p_t(x)\,(W_x+\rho D_x)$$

$$=1+\sum_{x=0}^{b-1} p_t(x)\, C_x. \qquad \square$$

4. Branch-and-Bound Approach

Let $C_{\mathrm{opt},t}$ denote the optimal tree cost when test t is executed at the root. *Theorem 1* shows that the tree cost is determined by its subtree costs $C_0, \cdots, C_{b-1}$, if a test is assigned to the root. Therefore, if $C_{\mathrm{opt},t}$ is achieved, each subtree cost must be minimized as well. Let $C_{\mathrm{opt},t}(x)$ represent the minimized subtree cost corresponding to test outcome $x\,(0\leqq x\leqq b-1)$. Since the root is either a leaf or an internal node associated with some test, the optimal tree cost C_{opt} satisfies

$$C_{\mathrm{opt}}=\min[C_{\mathrm{leaf}},\ C_{\mathrm{opt},t_1},\ \cdots,\ C_{\mathrm{opt},t_N}]$$

$$=\min\left[C_{\mathrm{leaf}},\ \min_{t\in\mathcal{T}}\left(1+\sum_{x=0}^{b-1} p_t(x)\, C_{\mathrm{opt},t}(x)\right)\right] \qquad (5)$$

where C_{leaf} is the cost of the tree consisting of only a leaf. This relation shows that the optimal tree is constructed by executing the procedure DPsearch shown in Fig. 3. This method is based on the dynamic programming technique.

The procedure DPsearch splits the given problem into subproblems, in which $C_{\mathrm{opt},t}$ is calculated by recursive calls. However, it is very time consuming to solve the subproblem for every $t\in\mathcal{T}$. Branch-and-bound methods[9] can effectively eliminate this dis advantage. In the one used here, a lower bound of $C_{\mathrm{opt},t}$ is calculated, and the subproblem is not solved if the bound exceeds the cost of a known feasible solution. Exclusion of such a subproblem obviously does not affect the optimality of the solution. The algorithm is represented by the procedure B&B shown in Fig. 4, where $G(\boldsymbol{Q},\ \mathcal{T})$ denotes the lower bound of cost for a given state probability vector $\boldsymbol{Q}=\{Q(u_1), \cdots, Q(u_K)\}$ and a given test set $\mathcal{T}$. In this procedure, the lower bound of $C_{\mathrm{opt},t}$ is calculated from the relation,

$$C_{\mathrm{opt},t}\geqq 1+\sum_{x=0}^{b-1} p_t(x)\, G(\boldsymbol{Q}|x,\ \mathcal{T}-\{t\}),$$

where $\boldsymbol{Q}|x=\{Q(u_1|x), \cdots, Q(u_K|x)\}$. This relation is easily verified from Eq. (2).

To implement this algorithm, we need to find the lower bound $G(\boldsymbol{Q},\ \mathcal{T})$. For efficient execution, this bound must be quickly computable and also tight so as to exclude as many needless subproblems as possible.

```
Procedure DPsearch(Q, T, DT, C_opt)
INPUT: Q = {Q(u_1),..., Q(u_K)}, test set T;
OUTPUT: optimal tree DT, optimal cost C_opt
begin
DT:= A tree consisting of only a leaf; C_opt:= C_leaf;
for every t ∈ T do
  begin
  for x = 0 until b - 1 do DPsearch(Q|x, T - {t}, DT_x, C_opt,t(x));
  C_opt,t:= 1 + Σ_x p_t(x) C_opt,t(x);
  if C_opt,t < C_opt then
    begin
    DT:= A tree consisting of the root labeled with t
         and subtrees DT_0,...,DT_b-1;
    C_opt:= C_opt,t
    end
  end
end.
```

Fig. 3 Computing procedure based on the dynamic programming technique.

```
Procedure B&B(Q, T, DT, C_opt)
INPUT: Q = {Q(u_1),..., Q(u_K)}, test set T;
OUTPUT: optimal tree DT, optimal cost C_opt
begin
DT := A tree consisting of only a leaf; C_opt:= C_leaf;
for every t ∈ T do
  begin
  g:= 1 + Σ_x p_t(x) G(Q|x, T - {t});
  if g < C_opt then
    begin
    for x = 0 until b - 1 do B&B(Q|x, T - {t}, DT_x, C_opt,t(x));
    C_opt,t:= 1 + Σ_x p_t(x) C_opt,t(x);
    if C_opt,t < C_opt then
      begin
      DT:= A tree consisting of the root labeled with t
           and subtrees DT_0,...,DT_b-1;
      C_opt:= C_opt,t
      end
    end
  end
end.
```

Fig. 4 Computing procedure employing a branch-and-bound method.

5. Lower Bound of Tree Cost

In this section, we derive a lower bound of the tree cost that is advantageous for utilization in the branch-and-bound algorithm because it is easy to compute. For simplicity, we assume that states are denoted by 1, $\cdots$, K, and $Q(1)\geqq Q(2)\geqq\cdots\geqq Q(K)$ without loss of generality.

Lemma 1: Suppose that a decision tree T_1 has $V(a)$ $(0 \leqq V(a))$ leaves $l_{a,1}, \cdots, l_{a,V(a)}$ labeled with action $a \in \mathcal{A}$. Let $P(l_{a,v}|k)$ denote the probability that one reaches the leaf $l_{a,v}$ conditioned by the state $k \in \mathcal{U}$, and $\omega_{a,v}$ denote the unconditional probability of arrival to the leaf $l_{a,v}$ That is,

$$\omega_{a,v} = \sum_{k \in v} P(l_{a,v}|k)\, Q(k). \tag{6}$$

Then, the cost of the tree T_1, which is denoted by C, is bounded by

$$C \geqq -\sum_a \sum_v \omega_{a,v} \log_b \omega_{a,v} + \rho \sum_k \sum_a \sum_v d(k;\ a) \cdot P(l_{a,v}|k)\, Q(k). \tag{7}$$

Proof: A decision tree can be identified as a prefix-free code tree by considering a test results sequence to be a code word. Since the expected test execution time W is equivalent to the average word length of the code, the well known source coding theorem in information theory is applicable to find the lower bound of W. Therefore, W is not less than the entropy of the message, i.e., the first term in the right hand of Eq. (7). The second term is clearly the expected loss achieved by the tree. This immediately establishes Eq. (7). □

Now, consider $P(l_{a,v}|k)$'s as variables and minimize the right hand of Eq. (7) subject to

$$0 \leqq P(l_{a,v}|k) \leqq 1, \text{ and } \sum_{a,v} P(l_{a,v}|k) = 1,$$

$$\text{for } 1 \leqq k \leqq K.$$

Then, the minimum g is expressed by

$$g = \min_{P(l_{a,v}|k)} \left[-\sum_a \sum_v \omega_{a,v} \log_b \omega_{a,v} + \rho \sum_k \sum_a \sum_v d(k;\ a)\, P(l_{a,v}|k)\, Q(k) \right]. \tag{8}$$

Clearly, g is not larger than the cost of an arbitrary tree with $V(a)$ leaves for each action a.

Lemma 2: If the minimum of Eq. (8) is attained, then for any action $a \in \mathcal{A}$, the minimizing $P(l_{a,v}|k)$ must be such that $\omega_{a,v} > 0$ for at most one pair of (a, v) among $V(a)$ paris $(a, 1), \cdots, (a, V(a))$ associated with the action a.

Proof: Assume that the minimizing probability assignment $P_0(l_{a,v}|k)$ is such that $\omega_{a*,1} > 0$ and $\omega_{a*,2} > 0$ for two pairs $(a^*, 1)$ and $(a^*, 2)$ corresponding to a certain action $a^* \in \mathcal{A}$. Now, consider the new probability assignment $P'(l_{a,v}|k)$, which is identical to $P_0(l_{a,v}|k)$ except that

$$P'(l_{a*,1}|k) = P_0(l_{a*,1}|k) + P_0(l_{a*,2}|k), \text{ and}$$

$$P'(l_{a*,2}|k) = 0, \text{ for all } k. \tag{9}$$

Then, the second term of Eq. (8) obviously remains unchanged for $P'(l_{a,v}|k)$. On the other hand, it is easy to verify that $P'(l_{a,v}|k)$ gives a smaller value of the first term than $P_0(l_{a,v}|k)$ does. This is a contradiction and proves the lemma. □

The lemma asserts that only one leaf, say $l_{a,1}$, associated with each action a is necessary to achieve the minimum of Eq. (8) and other leaves $l_{a,2}, \cdots, l_{a,V(a)}$ are unnecessary if $V(a) \geqq 2$. Thus, a tree having two or more leaves labeled with action a yields the identical value of g as the smaller tree that has only one leaf for the action a and the same number of leaves for other actions.

Here, it should be noticed that there are two types of trees. A tree of the first type, denoted Type 1, has at least one leaf associated with each action. Contrarily, a tree of the second type, Type 2, has no leaves for some actions.

Lemma 3: Let g_1 and g_2 denote the minimums defined by Eq. (8) for a Type 1 tree and a Type 2 tree. Then,

$$g_1 \leqq g_2. \tag{10}$$

Proof: Let $P_{0,2}(l_{a,v}|k)$ represent the probability assignment giving g_2 and let $\mathcal{A}_2$ be the set of actions that have corresponding leaves in a Type 2 tree (thus, $\mathcal{A} \subset \mathcal{A}_2$). From *Lemma 2*, we can assume that

$$P_{0,2}(l_{a,1}|k) \geqq 0, \text{ and } P_{0,2}(l_{a,v}|k) = 0$$

for all $v \geqq 2$ and all $a \in \mathcal{A}_2$. However, if $P(l_{a,v}|k)$ for a Type 1 tree is set to be such that

$$P(l_{a,1}|k) = P_{0,2}(l_{a,1}|k) \text{ for } a \in \mathcal{A}_2, \text{ and}$$

$$P(l_{a,v}|k) = 0 \text{ for } v \geqq 2 \text{ or } a \notin \mathcal{A}_2$$

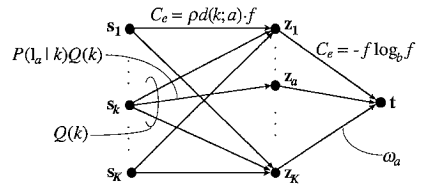
then the right hand of Eq. (7) obviously yields g_2. Since the minimization to obtain g_1 is done over probability assignments including $P(l_{a,v}|k)$ defined above, g_1 is not larger than g_2. □

Let $G_\rho(\boldsymbol{Q})$ denote g of Eq. (8) for a tree that has just one leaf associated with every action in $\mathcal{A}$. That is,

$$G_\rho(\boldsymbol{Q}) = \min_P \left[-\sum_a \omega_a \log_b \omega_a + \rho \sum_k \sum_a d(k;\ a) \cdot P(l_a|k)\, Q(k) \right], \tag{11}$$

where $l_{a,1}$ or $\omega_{a,1}$ is rewritten as l_a or ω_a. Evidently from Lemmas 2 and 3, $G_\rho(\boldsymbol{Q})$ is the lower bound of cost for an arbitrary tree with any number of leaves.

It is not easy to see what values of $P(l_a|k)$'s give the minimum in Eq. (11). To represent $G_\rho(\boldsymbol{Q})$ in an easily computabe form, it is convenient to introduce the model network shown in Fig. 5. In this network, there are K source nodes $s_1, \cdots, s_K$ corresponding to states $1, \cdots, K$, K intermediate nodes $z_1, \cdots, z_K$ corresponding to leaves $l_1, \cdots, l_K$, and one sink node t. A source node s_k $(1 \leqq k \leqq K)$ sends a flow involving the total amount of $Q(k)$ to the sink t. Then, the amount of the flow sent from s_k to z_a $(1 \leqq a \leqq K)$ is obviously represented by $P(l_a|k)\ Q(k)$ and the flow amount on



C_e: Edge Cost, f: Flow Amount on an Edge

Fig. 5 Model network associated with the lower bound $G_p(\boldsymbol{Q})$.

the edge between z_a and t is equal to ω_a. The vector representing all these flows is termed a flow pattern. The edge cost C_e is defined as follows:

$$C_e=\begin{cases}\rho d(k;\ a)\cdot f, & \text{for the edge between } s_k \text{ and } z_a,\\ -f\log_b f, & \text{for the edge between } z_a \text{ and } t,\end{cases}$$

where f is the amount of the flow on the edge. The flow pattern is said to be optimal if the network cost is minimized. Then, the minimization of Eq. (11) is clearly equivalent to optimization of the flow pattern. This problem is a minimum concave cost flow problem[15].

Lemma 4: If the flow pattern is optimal, then the flow from s_k to t follows only one route.

Proof: The lemma is clearly valid from the characteristics of the minimum concave cost flow problem described in Ref. (15). □

Since there are too many $(=K^K)$ flow patterns satisfying Lemma 4, it is impossible to search the optimum from them all. To efficiently find the lower bound, we need to clarify more specific conditons which are to be satisfied by the optimal flow pattern.

Definition: a *loss-generating flow* is a flow on the edge between s_k and z_a such that $k\neq a$ (thus, $d(k;\ a)=1$), while a *loss-free flow* passes the edge between s_k and z_k.

Lemma 5: If the flow pattern is optimal, then every loss-generating flow traverses the identical intermediate node.

Proof: Assume that two loss-generating flows, termed r_i and r_j, pass different intermediate nodes z_i and z_j $(1\leqq i,\ j\leqq K)$ in the optimal flow pattern. Let f_i and f_j denote the amount of these flows. Let the amount of the flow on the edge between z_i and t be F_i+f_i and that on the edge between z_j and t be F_j+f_j. We can assume $F_i\geqq F_j\geqq 0$ without loss of generality. Then, consider a new flow pattern in which the flow r_j is rearranged so as to traverse the node z_i. This reassignment does not increase the cost portion associated with edges from s_k's to z_a's because the cost coefficient of the edge involving r_j does not increase. It is also verified that the cost decreases on edges between z_a's and t. Let the cost change on these edges be Δ. Then,

$$\begin{aligned}\Delta=&-(F_i+f_i+f_j)\log_b(F_i+f_i+f_j)-F_j\log_b F_j\\&+(F_i+f_i)\log_b(F_i+f_i)\\&+(F_j+f_j)\log_b(F_j+f_j)\\<&-\frac{f_j(F_i-F_j+f_i)}{(F_i+f_i+f_j)\ln b}<0.\end{aligned}$$

Consequently, the new flow pattern gives lower cost. This is a contradiction. □

Lemma 6: There exists an optimal flow pattern in which every loss-generating flow traverses intermediate node z_1.

Proof: First, we show that there is an optimal flow pattern in which the flow sent out from s_1 is a loss-free flow. Suppose that this flow is a loss-generating flow traversing the node z_i $(2\leqq i\leqq K)$ when the cost is minimized. From *Lemma 5*, every loss-generating flow runs into z_i, and thus no flow exists on the edge from z_1 to t. Here, rearrange every flow passing z_i such that it traverses z_1. The cost increment by this rearrangement is at most $\rho(Q(i)-Q(1))$. Since $Q(i)\leqq Q(1)$, the rearranged flow pattern yields another optimal solution. Thus, we can achieve an optimal flow pattern in which the flow from s_1 is a loss-free flow.

Then, assume that every loss-generating flow traverses the node z_i, $(2\leqq i\leqq K)$ in this flow pattern. Here, rearrange all the loss-generating flows so as to pass z_1. This obviously does not change the cost generated on edges between s_k's and z_a's. Also, it is proved by using $Q(1)\geqq Q(i)$ and primary calculations that the cost of edges between z_a's and t does not increase. Therefore, the minimum cost is achieved by the rearranged flow pattern in which every loss-generating flow passes z_1. □

Lemma 7: If the flow sent from s_k $(2\leqq k\leqq K)$ is a loss-generating flow passing the intermediate node z_1 in an optimal flow pattern, then the flow sent from $s_{k'}$ such that $Q(k')<Q(k)$ is also a loss-generating flow $(k<k'\leqq K)$.

Proof: Assume that the flow sent from $s_{k'}$ is a loss-free flow and $Q(k')\leqq Q(k)$ when the flow sent from s_k is a loss-generating flow in an optimal flow pattern. Then, rearrange the flows as follows:

- Split the flow sent from s_k into two parts. One part is routed via z_k and its amount is $Q(k')$. The other is routed on z_1 and its amount is $Q(k)-Q(k')$.
- Move the flow sent from $s_{k'}$ so as to pass z_1.

By performing this rearrangement, the total amount of loss-generating flows does not vary, nor does cost change on edges between z_a's and t. Therefore, the reassigned flow pattern gives the same cost as the original flow pattern. However, the rearranged flow does not satisfy the necessary condition of Lemma 4. This is a contradiction. □

Lemmas 6 and 7 show that the minimum network cost is found from particular flow patterns in which flows from $s_1, \cdots, s_m$ are loss-free flows and those from $s_{m+1}, \cdots, s_K$ are loss-generating flows traversing z_1 as shown in Fig. 6. Because there are only K such flow assignments, the minimum is very easy to compute. By

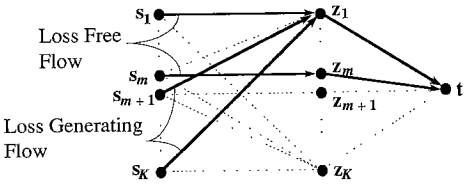


Fig. 6 Optimal flow pattern yielding $G_\rho(\boldsymbol{Q})$.

observing the network costs corresponding to these K patterns, the next theorem is obtained.

Theorem 2: Let the cost of an arbitrary tree be C for a given $\boldsymbol{Q}=\{Q(1), \cdots, Q(K)\}$ and ρ. Then, C is lower bounded by

$$C \geqq G_\rho(\boldsymbol{Q}),$$

where

$$G_\rho(\boldsymbol{Q}) = \min_{1 \leq m \leq K} \left[-\sum_{k=1}^{m} Q_m(k) \log_b Q_m(k) + \rho \sum_{k=m+1}^{K} Q(k) \right], \tag{12}$$

and

$$Q_m(k) = \begin{cases} Q(1)+Q(m+1)+\cdots+Q(K), \\ \qquad \text{for } k=1, \\ Q(k), \text{ for } 2 \leqq k \leqq m. \end{cases}$$

Another lower bound can be obtained by using the rate-distortion function. However, this bound is not as tight as $G_\rho(\boldsymbol{Q})$ (see Appendix 1). Additionally, the computation of $G_\rho(\boldsymbol{Q})$ is extremely easy in comparison with that of the rate-distortion function, as shown in Appendix 2.

By using $G_\rho(\boldsymbol{Q})$ and Eq.(5), another lower bound denoted by $G_\rho^*(\boldsymbol{Q}, \mathcal{T})$ is derived as follows:

$$G_\rho^*(\boldsymbol{Q}, \mathcal{T}) = \min \left\{ \rho(1-Q(1)), \min_{t \in \mathcal{T}} \left[1 + \sum_{x=0}^{b-1} p_t(x)\, G_\rho(\boldsymbol{Q}|x) \right] \right\}. \tag{13}$$

Despite increased complexity in computation, this lower bound $G_\rho^*(\boldsymbol{Q}, \mathcal{T})$ is advantageous in giving a tighter bound than $G_\rho(\boldsymbol{Q})$.

6. Evaluation

A computational expeiment was carried out to evaluate the effectiveness of the branch-and-bound algorithm employing the lower bound of the cost derived in Sect. 5. The algorithm was programmed in Turbo-Pascal on an NEC PC-9801UX41 computer. The lower bound $G_\rho^*(\boldsymbol{Q}, \mathcal{T})$ defined by Eq.(13) was

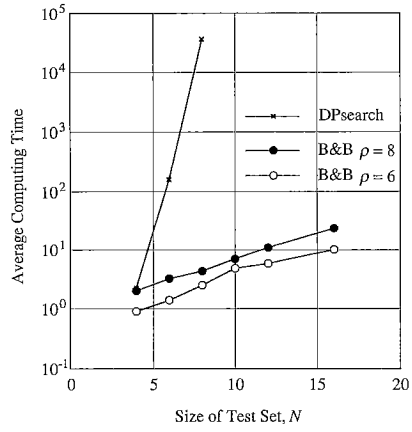


Fig. 7 Experimental result. B&B denotes the proposed branch-and-bound method and DP search denotes the dynamic programming method without utilizing the lower bound.

used. For comparison, we also coded the dynamic programming method without utilizing the lower bound. Randomly generated problems were fed to both of the programs and computing time was measured.

To examine the behavior of the algorithm as a function of test set size, we generated 6 classes of problems (10 problems in each class) with 4, 6, 8, 10, 12, 16 tests. The size of state sets was taken to be 4 and coefficient b was set to be 2. Prior state probabilities $Q(u)$'s were chosen randomly from uniform distribution over the interval [0, 1] and then normalized for their sum to be 1. To select the value of conditional test outcome probabilities $P_t(0|k)$ and $P_t(1|k)$, we first chose one of two intervals [0, 0.1] and [0.9, 1] with probability of 0.5, and then drew $P_t(1|k)$ randomly from uniform distribution over the chosen interval. $P_t(0|k)$ was set to be $1-P_t(1|k)$. The weight parameter ρ was taken to be 6 or 8.

Figure 7 shows the experimental result. The average computing time is much less for the branch-and-bound algorithm in the range of $6 \leqq N$ than for the method without using the bound. Moreover, time increases much more slowly with increasing test size with the proposed method. This characteristic implies that the proposed method is applicable to considerably large problems. This result provides sufficient evidence for us to claim the effectiveness of the described lower bound in accelerating computation.

Among strict optimum algorithms, the method in Ref.(16) is relatively time efficient[11]. This method is applicable for the cost used in this paper, as seen from Ref.(13). While the algorithm is based on the dynamic programming technique, it differs from the procedure shown in Fig. 3 in the order of constructing subtrees. The time of the method is shown to be proportional to $N(b+1)^{N}$[11]. Meanwhile, the average

computing time of the proposed method seems to be proportional to 1.2^N or 1.3^N. Although this computing time is exponential for N, it is much less than $N(b+1)^N$. This infers that the proposed method is a very efficient algorithm yielding a strict optimum.

7. Conclusion

This paper described a branch-and-bound algorithm for constructing a probabilistic decision tree with the strictly minimum cost. The algorithm features the lower bound of the cost, which was derived based on the word length bound of a prefix-free code. The bound was transformed to an easily computable representation by introducing a model network. The result of a computational experiment showed that the proposed algorithm is much faster than the dynamic programming method without employing the bound. Also, the computing time with the proposed method increases more slowly than the method reported in Refs. (13) and (16).

The decision tree cost was defined as the weighted sum of expected test execution time and expected loss in this paper. While this tree cost is often a reasonable measure for efficiency, other cost functions (storage and expected loss, for example) may be more adequate in some applications. The extension of the proposed method to other relevant cost functions is left for further study. Comparison between the proposed method and an algorithm yielding approximate solutions is another remaining problem.

References

(1) Brulé J. D., Johnson R. A. and Kletsky E. J.: "Diagnosis of equipment failures", IRE Trans. Reliability and Quality Control, **RQC-9**, pp. 23-34 (April 1960).

(2) Reinwald L. T. and Soland R. M.: "Conversion of limited—entry decision tables to optimal computer programs I: Minimum average processing time", J. the Assoc. Comp. Mach., **13**, pp. 339-358 (July 1966).

(3) Casey R. G. and Nagy G.: "Decision tree design using a probabilistic model", IEEE Trans. Inf. Theory, **IT-30**, 1, pp. 93-99 (Jan. 1984).

(4) Quinlan J. R.: "Learning efficient classification procedures and their applications to chess-end games", Machine Learning: An Artificial Intelligence Approach, eds. R. S. Michaelski, J. G. Carbonell and T. M. Mitchell, Palo Alto, CA: Tioga, pp. 463-482 (1983).

(5) Khuri S., Hartmann C. R. P. and Varshney P. K.: "Application of information theory to the construction of efficient decision diagrams", preprint.

(6) Khuri S., Hartmann C. R. P. and Varshney P. K.: "Applying information theory to the construction of decision diagrams", Abstract Proc. Int. Symp. on Information Theory, p. 132 (June 1988).

(7) Hartmann C. R. P., Varshney P. K., Mehrotra K. G. and Gerberich C. L.: "Application of information theory to the construction of efficient decision trees", IEEE Trans. Inf. Theory, **IT-28**, 4, pp. 565-577 (July 1982).

(8) Goodman R. M. and Smyth P.: "Decision tree design from a communication theory standpoint", IEEE Trans. Inf. Theory, **IT-34**, 5, pp. 979-994 (Sept. 1988).

(9) Lawler E. L. and Wood D. E.: "Branch-and-bound methods: A survey", Oper. Res., **14**, pp. 699-719 (July 1966).

(10) Chou P. A. and Gray R. M.: "On decision trees for pattern recognition", Abstract Proc. Int. Symp. on Inf. Theory, p. 69 (Oct. 1986).

(11) Moret B. M. E.: "Decision rtrees and diagrams", Computing Surveys, **14**, 4, pp. 593-623 (Dec. 1982).

(12) Sakawa M. and Yano H.: "Generalized scalarization methods for multiobjective optimization problems: Hyperplane methods", Trans. IEICE, **J71-A**, 3, pp. 791-797 (March 1988).

(13) Peters R. J.: "Zero order and nonzero order decision rules in medical diagnosis", IBM J. Res. and Develop., **21**, pp. 449-460 (Sept. 1977).

(14) Chou P. A., Lookabaugh T. and Gray R. M.: "Optimal pruning with applications to tree-structured source coding and modeling", IEEE Trans. Inf. Theory, **IT-35**, 2, pp. 299-315 (March 1989).

(15) Zangwill W. I.: "Minimum concave cost flows in certain networks", Management Science, **14**, 7, pp. 429-450 (March 1968).

(16) Garey M. R.: "Optimal binary identification procedures", SIAMJ. Appl. Math., **23**, 2, pp. 173-186 (Sept. 1972).

Appendix 1

The lower bound of the tree cost can be derived from the rate-distortion function. Let $R(d)$ be the rate-distortion function for a given state space and the Hamming metric loss function. For the expected loss d, a decision tree must gather $R(d)$ (nats) of the information about the state. Meanwhile, theinformation obtained by executing a test once is at most $\ln b$ (nats). Thus, the expected test execution time is not less than $R(d)/(\ln b)$ for the loss d. Therefore,

$$C \geqq \min_d \left[\frac{R(d)}{\ln b} + \rho d \right].$$

The next theorem shows the relation between this bound and $G_\rho(\boldsymbol{Q})$.

Theorem A1: For a given state space, the tree cost C and the lower bound $G_\rho(\boldsymbol{Q})$ satisfy

$$C \geqq G_\rho(\boldsymbol{Q}) \geqq \min_d \left[\frac{R(d)}{\ln b} + \rho d \right].$$

Proof: Let M be the value of m that gives the minimum in Eq. (12). Then, $P(l_j|k)$'s giving the minimum in Eq. (11) are as follows. (i) For $1 \leqq k \leqq K-M$, $P(l_j|k)=1$ if $j=k$, and $P(l_j|k)=0$ otherwise. (ii) For $K-M<k \leqq K$, $P(l_j|k)=1$ if $j=1$, and $P(l_j|k)=0$ otherwise.

Let I_M be the amount of the information obtained by the tree satisfying this probability assignment. It is easy to show that

$$I_M = -\sum_{k=1}^{m} Q_m(k) \ln Q_m(k).$$

Let D_M be the expected loss of this tree. Then,

$$C \geqq G_\rho(\boldsymbol{Q})$$

$$= I_M/(\ln b) + \rho D_M \geqq \min_d \left[\frac{R(d)}{\ln b} + \rho d\right]. \quad \square$$

Thus, the lower bound $G_\rho(\boldsymbol{Q})$ is tighter than the one derived from the rate-distortion function.

Appendix 2

This shows the computig scheme of $G_\rho(\boldsymbol{Q})$. By rewriting Eq. (12),

$$G_\rho(\boldsymbol{Q}) = \min_{1 \leq m \leq K} g_m, \qquad (A \cdot 1)$$

where

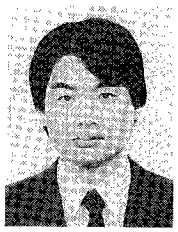
$$g_m = -\sum_{k=1}^{m} Q_m(k) \log_b Q_m(k) + \rho \sum_{k=m+1}^{K} Q(k).$$

The quantity g_m satisfies the next recurrence.

$$g_1 = \rho \sum_{k=2}^{K} Q(k),$$

$$g_{m+1} = g_m - Q_{m+1}(1) \log Q_{m+1}(1) - Q(m+1) \log Q(m+1) + Q_m(1) \log Q_m(1) - \rho Q(m+1).$$

By using this, g_m's are computed in $O(K)$ time and the minimization in Eq. (A·1) takes also $O(K)$ time. Thus, the computing time is determined by the sorting of $Q(u)$'s, which uses $O(K \log K)$ time.



Satoru Ohta was born in Yokosuka, Japan, in 1958. He received the B. E. and M. E. degree in electronics engineering from Tokyo Institute of Technology, Tokyo, Japan, in 1981 and 1983, respectively. In 1983, he joined the Yokosuka Electrical Communications Laboratories, NTT. During 1983-1989, he worked on research and development of cross-connect systems, transmission network synthesis, and broadband ISDN. Since 1989, he has been engaged in research on information theory at NTT Transmission Systems Laboratories. His current research interests include combinatorial optimization problems and applications of information theory. He is a member of the IEEE and the Society of Information Theory and Its Applications. He received the Excellent Paper Award in 1991 from IEICE.

Fumio Kanaya received the B. S., M. S., and Dr. Eng. degrees in electrical engineering from The University of Tokyo, Tokyo, in 1963, 1965, and 1976, respectively. In 1965 he joined the Electrical Communication Laboratories, NTT and has been engaged in research and development works on various communication systems including digital transmission and coding of images. Since July 1987, as Research Fellow he has superintended Kanaya Research Laboratory at NTT Transmission Systems Laboratories, Kanagawa. His curret research interests are in information theory, and its application to communication theory and statistical decision theory. Dr. Kanaya is a member of IEEE, SITA and JNNS. He received the 1970, and 1990 Paper Awards from IEICE Japan.